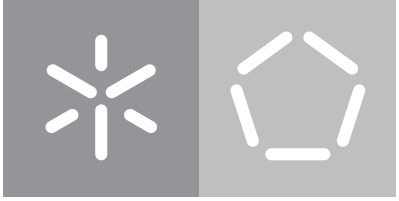
Universidade do Minho
Escola de Engenharia

Ricardo Barros Pereira

Development of a Self-diagnosis Tests System for Integration in a Cyber-physical System

July, 2021



Universidade do Minho
Escola de Engenharia

Ricardo Barros Pereira

Development of a Self-diagnosis Tests System for Integration in a Cyber-physical System

Master's Dissertation
Master's in Informatics Engineering

Work supervised by
Professor Doctor José Carlos Ramalho
Professor Doctor Miguel Abrunhosa de Brito

July, 2021

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the Universidade do Minho.

Acknowledgements

I would like to express my thanks to some special people who were very important during this year and throughout my academic journey.

First of all, I want to thank my advisors. To Professor Miguel Abrunhosa de Brito, for guiding me with excellence, for all his availability, and all the knowledge and motivation he gave me throughout this year. To Professor José Carlos Ramalho, for all the knowledge, experience and motivation that he provided me, not only as advisor of this dissertation but also as a teacher on several occasions throughout my academic journey. To Professor Pedro Rangel Henriques for his brilliant knowledge and ideas on domain-specific languages.

To Centro de Computação Gráfica, for all the good work environment that it provided me and for all the help during the beginning of this work, until the pandemic situation sent us all to our homes and we had to say goodbye.

To my family, for always being by my side and for helping me whenever I need to.

To my closest friends, for always being present and also for the experiences we had during this journey. A special thanks to my friend Hugo Carvalho for being my great example in my academic journey and for always motivating, advising and helping me to make the right choices.

To my dear girlfriend, Océane, for always being by my side, for all the support and all the patience she had with me while I was doing my dissertation.

Finally, I would like to dedicate this work and my entire academic journey to my parents, António and Isabel, for the importance and significance that this achievement has for them. Special thanks to them for always supporting me and transmitting the best values.

My sincere thanks to all of you!
Ricardo Pereira

This work is a result of the project POCI-01-0247-FEDER-040130, supported by Operational Program for Competitiveness and Internationalization (COMPETE 2020), under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF).

Resumo

Hoje, a CONTROLAR fornece para a Bosch a Intelligent Functional Test System Machine, um sistema ciber-físico desenvolvido para realizar diferentes níveis de testes funcionais em dispositivos e componentes electrónicos. A Bosch utiliza-a para testar o correto funcionamento dos auto-rádios produzidos. Durante este processo, os auto-rádios são submetidos a vários testes e o problema surge quando a máquina detecta erros em vários auto-rádios consecutivos e não é possível saber se a própria máquina está com problemas, pois não possui nenhum módulo que permita saber se está a funcionar corretamente ou não.

A origem deste trabalho surge da necessidade de encontrar uma solução que resolva o problema enunciado, mas também, inovadora e com contribuições para o mundo da investigação em sistemas ciber-físicos e sistemas de testes de autodiagnóstico. A solução é integrar um sistema de autodiagnóstico na máquina que possa testar o seu funcionamento para que a Bosch possa ter certeza se o problema está na máquina ou nos auto-rádios. Como a máquina é um sistema ciber-físico, permite a integração de um sistema de software que possa gerir a execução de testes, sendo capaz de detectar falhas nas máquinas.

O trabalho aqui apresentado aborda o problema criando um novo sistema de testes de autodiagnóstico que garantirá a confiabilidade e integridade do sistema ciber-físico. Em detalhe, esta dissertação começa por expôr um estudo sobre o estado da arte atual de sistemas ciber-físicos, automação de testes, metodologia de teste keyword-driven e mais alguns conceitos relacionados a linguagens específicas de domínio que serão relevantes para a solução final. São apresentadas a especificação e análise do sistema, a fim de definir bem os seus componentes. Uma nova arquitetura modular e extensível é proposta para sistemas de testes de autodiagnóstico, bem como uma arquitetura para estendê-lo e integrá-lo num sistema ciber-físico. Foi proposto um novo sistema de testes de autodiagnóstico que aplica a arquitetura proposta provando que é possível realizar o autodiagnóstico em tempo real do sistema ciber-físico e permitindo a integração de qualquer tipo de teste. Para validar o sistema, foram realizados 28 casos de teste, abrangendo todas as suas funcionalidades. Os resultados mostram que todos os casos de teste passaram e, portanto, o sistema cumpre todos os objetivos propostos.

Palavras-chave: Sistemas Ciber-físicos, Auto-diagnóstico, Automação de Testes, Aplicação Web

Abstract

Nowadays, CONTROLAR supplies with Bosch the Intelligent Functional Test System Machine, a cyber-physical system developed to perform different levels of functional tests on electronic devices and components. Bosch uses it to test the correct functioning of the produced car radios. During this process, the car radios are subjected to several tests and the problem arises when the machine detects errors in several consecutive car radios and it is not possible to know if the machine itself has any problems, as it does not have any module that allows knowing whether it's working correctly or not.

The origin of this work arises from the need to find a solution that solves the referred problem, but also, innovative and with contributions to the world of research in cyber-physical systems and self-diagnosis tests systems. The solution is to integrate a self-diagnosis system into the machine that can test its functionality so that when these car radio failures appear, Bosch can be sure whether the problem is with the machine or the car radio. As the machine is a cyber-physical system, it allows the integration of a software system to control and manage all its actions. Therefore, it is necessary to develop a system to manage the tests and their execution, being able to detect internal failures in the machines.

The work presented here addresses the problem by creating a new self-diagnosis tests system that will guarantee the reliability and integrity of the cyber-physical system. In detail, this dissertation begins by exposing a study on the current state of the art of cyber-physical systems, test automation, keyword-driven test methodology and some more concepts related to domain-specific languages that will be relevant to the final solution. The specification and analysis of the system are presented, to define well its components. A new modular and extensible architecture is proposed for self-diagnosis test systems, as well as a methodology for extending and integrate it into a cyber-physical system. A new self-diagnosis tests system has been proposed that applies the proposed architecture proving that it is possible to carry out the self-diagnosis in real-time of the cyber-physical system and allowing the integration of any type of test. To validate the implementation of the system, 28 test cases were carried out to cover all its functionalities. The results show that all test cases passed and, therefore, the system meets all the proposed objectives.

Keywords: Cyber-physical systems, Self-diagnosis, Test automation, Web Application

Contents

List of Figures

List of Tables

Listings

Acronyms

ANTLR Another Tool for Language Recognition

API Application Programming Interface

CPS Cyber-physical system

CRUD Create, Read, Update, and Delete

CSS Cascading Style Sheets

DOM Document Object Model

DSL Domain-Specific Language

DUT Device Under Test

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

JSON JavaScript Object Notation

JWT JSON Web Token

KDT Keyword-Driven Testing

MERN MongoDB, Express, React, Node

MVC Model-View-Controller

ODM Object Data Modeling

REST Representational State Transfer

TSIM Test System Intelligent Machines

URI Uniform Resource Identifier

URL Uniform Resource Locator

WWW World Wide Web

XML Extensible Markup Language

Chapter 1

Introduction

This research work is inserted in the context of a project called Test System Intelligent Machines (TSIM) and this project comes out from the union of skills and knowledge of the Consortium between CONTROLAR, University of Minho [4] and Centro de Computação Gráfica [7]. This arises from the challenge launched by the partnership between Bosch and the University of Minho through the Bosch Supplier Club Initiative Program.

Nowadays, CONTROLAR is a company dedicated to the development of hardware and software for the industry, with a great vocation in the automotive electrical components industry and great know-how in industrial automation, development of functional and quality test systems for electronic devices. One of its business areas is the Test of systems and has as base the development and integration of functional test systems, monitoring data for the validation of electrical characteristics and quality tests, using data collection plates, generators, and specific equipment for the acquisition and analysis of signal and data. Within this business unit, Controlar has been working with several leading manufacturers worldwide, such as the multinational Bosch.

The objective of this project is to guarantee the creation of value in CONTROLAR's products (as a current Bosch supplier) and its adaptation to the reality of Bosch 4.0 Industry and the requirements of the automotive sector. The objective is to do this through the development of tools that make CONTROLAR's automatic test equipment more flexible, efficient and intelligent.

One of the products that CONTROLAR supplies to Bosch today is the Intelligent Functional Test System Machine [10]. This machine is a CPS and was developed to perform different levels of functional tests on electronic devices and components. Bosch uses this machine at the end of the production line to guarantee the correct functionality of the produced car radios.

1.1 Problem

During the process of habitual use of this machine, the car radios are subjected to various tests and the machine approves the car radios whose functionalities worked correctly compared to the tests carried out. The problem appears when the machine detects errors in several car radios in a row, which could be an indicator that the production line would have failed in one of its segments, thus causing possible errors in all car radios in that line. Obviously, a scenario in which all the car radios come with errors means that something has failed in production, and for a company like Bosch this is unacceptable as it will cause unexpected delays and all the consequences that result. Besides, it makes it necessary to fix all car radios wich brings increased costs.

When Bosch faces this problem it tends to attribute the problem to the machine and not to its products and, correctly or not, tries to find out what the problem is with the machine. Another additional problem is the fact that the machine does not have any module that allows understanding if it is working correctly or not so the only way they have to know it is by changing physical connections and internal properties of the machine construction to try to understand if something is a malfunction. But most of the time this happens, those looking for errors in the machine usually end up always causing damage to the machine itself without realizing it, as they do not know all the details of its construction and start to change many properties until they make the machine completely unusable and without finding the problem. The most likely result of these actions is the need to repair the machine or even replace it, which obviously causes financial losses for CONTROLAR and Bosch itself.

1.2 Motivation and Objectives

The origin of this work arises from the need to find a solution that can solve the problem stated, but also that it is an innovative solution with contributions to the world of research in CPS and self-diagnosis tests systems. The solution is to integrate a self-diagnosis system in the machine that can test the functionality of the machine itself so that when these car radios failures appear, Bosch can be sure if the problem is really on the machine or in the production line of the car radios. As the machine is a CPS, it allows the integration of a software system that can control and manage all of its actions. Therefore, it is necessary to develop a testing system to manage the tests and their execution, being able to detect internal machine failures. But before we can develop a system, it is necessary to find the correct architecture for a testing system. This architecture should be as suitable as possible to the problem we are facing, but also as generic as possible regarding what a testing system is.

The main objective of this dissertation is to develop a self-diagnosis tests system for integration in a CPS, which is capable of performing the self-diagnosis of CPS in real-time, thus guaranteeing its integrity. The steps and objectives for each phase of this work are as follows:

- Analysis and specification of the system;

- Develop and propose a new architecture for test management and configuration;
- Develop and propose a new architecture for the self-diagnosis tests system;
- Propose an architecture to integrate the self-diagnosis tests system in a CPS;
- Development of a self-diagnosis tests system;
- Validation of the self-diagnosis tests system;
- Integration of the self-diagnosis tests system with the CPS.

The work/research methodology will be as follows:

- Study of the problem;
- Survey of the state of the art;
- Study on the technologies to be used in development;
- Development of the system architecture;
- Development, validation and integration of the system.

1.3 Contributions

As previously mentioned, it was expected not only to find a solution to the problem presented but also that the solution found was innovative and had contributions to the current state of the art. Therefore, this dissertation presents the following contributions:

- A modular and extensible architecture for self-diagnosis tests systems that combines the keyword-driven testing methodology with a domain-specific language for managing and configuring the tests of the system. This contribution is important because this architecture can be applied to any self-diagnosis tests system without restricting any type of test, physical or logical, and can also be extended to CPS.
- An architecture to extend and integrate the self-diagnosis tests system into a CPS. This contribution is important because this architecture proves the modularity of the architecture of the self-diagnosis tests system, demonstrating how we can extend it in a CPS.
- A self-diagnosis tests system ready to be integrated into a CPS. This contribution allows validating the proposed architectures and the usefulness of the system developed in the self-diagnosis of CONTROLAR's machines.

As a result of this work, two research papers were produced, published and presented:

- *"Architecture based on keyword driven testing with domain specific language for a testing system"* was published at the *"32nd IFIP International Conference on Testing Software and Systems"* (ICTSS 2020)
- *"Development of Self-diagnosis Tests System Using a DSL for Creating New Test Suites for Integration in a Cyber-physical System"* was published at the *"10th Symposium on Languages, Applications and Technologies"* (SLATE 2021)

1.4 Dissertation Structure

This document is organized as follows: Chapter 2 analyzes the state of the art in CPS and self-diagnosis systems to understand whether strategies or architectures already exist for these two systems to be combined into one. This chapter also examines the state of the art in test automation and KDT methodology, in addition to analyzing and comparing some current tools that use this tests methodology. A few more concepts that will also be important for this work are also introduced in this chapter. Chapter 3 analyzes and specifies the requirements and structure of the system and defines the technologies used for its implementation. Chapter 4 explains the architecture defined for the system, where this process was divided into 3 steps, starting with the definition of an architecture for the management and configuration of the system tests, which will then be integrated into the architecture of the self-diagnosis tests system, and finally, the CPS architecture is composed, integrating all its components and the two previously defined architectures. Chapter 5 describes the implementation of the system, explaining each of the components involved, and presents a validation of the system carried out through several of the test cases performed. Finally, chapter 6 contains a conclusion of this work, which summarizes all the work done and also presents future work.

Chapter 2

State of the art

In this chapter, a review of the state of the art in CPS and test automation will be presented in Sections 2.1 and 2.2, respectively. In Section 2.3, is presented the KDT Framework and some current tools that are a demonstration of its use, as well as the advantages and disadvantages of using them. In Section 2.4, a brief introduction is made to the concept of DSL and, more specifically, how to apply this concept with the Another Tool for Language Recognition (ANTLR).

2.1 Cyber-physical systems

CPS are integrations of computing, network, and physical processes. Embedded computers and networks monitor and control physical processes. The economic and social potential of such systems is huge, and major investments are being made worldwide to develop the technology. CPS integrates the dynamics of physical processes with software and the network, providing abstractions and modelling, design, and analysis techniques for the integrated whole. But to realize the full potential of CPS, the main abstractions of computing need to be rethought and improved [32].

According to the state-of-the-art, the CPS provides the necessary technology to improve the realization and automation corresponding to a complex system on a large scale. Currently, CPS requires solutions that support it at the device, system, infrastructure, and application level. This is a challenge that includes an engineering approach and a fusion of communication, information, and automation technologies [22, 35, 50]. To take advantage of the full potential of CPS, abstractions of computing need to be rethought and improved. But to improve the effective orchestration of software and physical processes, semantic models are needed to reflect the relevant properties to both [32]. Traditional control systems are adapted to each case, requiring a very expensive and time-consuming effort to develop, maintain, or reconfigure. The current challenge is to develop innovative, agile, and reconfigurable architectures for control systems, using emerging technologies and paradigms that can provide the answer to these requirements [34]. It is

still necessary to maintain a balance between these requirements and the notion of lightweight and safe solutions [39]. Despite the great development of CPS, the development of software for it remains a large and growing area, with a rich body of knowledge [57].

Concerning testing, the communication infrastructure is essential. Most tests focus on communication-oriented research, privacy and security of the communication infrastructure. For the tests created to be as appropriate as possible, research and investigation must focus on specific needs. The interconnection of the tests would provide an excellent opportunity to combine tests with different resources. This will provide an opportunity to reduce operating and investment costs. Therefore, tests must be designed to provide a remote interface [8]. Verification and composition testing methods must also be adapted to the CPS. Creating an automated or semi-automatic method to assess the results of system tests is a challenge in CPS testing that also deserves attention [5]. The CPS test remains a challenging field of research due to the increasing heterogeneity, scale, and complexity. While CPS bring some advances to existing testing theory and technology, there are still many limitations to the broader industrial application of CPS testing [60]. The challenge will be to automate the maximum number of tasks in this process and get the most out of CPS.

2.2 Test Automation

The importance of testing automation is directly related to the quality of the final product. The execution of all functional tests before delivery guarantees the lowest incidence of errors in the post-delivery of the final product. As such, software developers/creators are required that their projects maintain a certain quality standard during all phases of development until the launch of a new product. Therefore, testing at the end of each stage no longer works in a professional environment. This is because the occurrence/discovery of unforeseen obstacles can significantly delay the development of the software. In recent years, it has been found that the software development market has increased its competitiveness, due to the modernization of the technologies involved and due to the maturity of the capacity to develop software. Thus, the range of information technology solutions, to meet the needs of consumer organizations, has increased considerably, which ends up making it difficult for users to choose when purchasing a product. In this competitive scenario, consumer organizations, when opting for software, are increasingly relying on quality criteria. One of the pillars for ensuring this quality of the software product is the testing process [6].

In the current software market, the concern for creating quality and error-free products has led companies to look for models and processes that guarantee quality to satisfy the needs of their customers. Unsuccessful projects, with expired deadlines and defective products, lead to customer dissatisfaction, high maintenance costs and compromise the company's image. The main objective of a software test is to define the implementation of this software that meets all the specifications and expectations defined and expected by the customer, that is, the objective is to "verify" if what was specified in the requirements phase is what really was developed. When verifying that the implemented software meets all specifications and expectations defined and expected by the customer, it is also looked for errors in the software. The

software test must be seen as a part of its quality process. Test automation is not limited to just performing the tests but above all being aware of when and where the tests need to be carried out, thus leaving the test team more time to plan more effective tests with better quality accuracy instead of worrying about scheduling them. Thus, automation results in the mechanization of the entire process of monitoring and managing the needs for testing and evaluation associated with software development [17].

2.3 Keyword-Driven Testing

KDT is a type of functional automation testing methodology that is also known as table-oriented testing or action-based testing. In KDT, we use a table format, usually a spreadsheet, to define keywords or action words that represent the content of the tests in a simple way. But it also allows the use of a keyword to represent part of the test case and in this way make the creation of the test case simpler, since we can reuse the keywords and the whole process they represent in different test cases. It allows novice or non-technical users to write tests more abstractly and it has a high degree of reusability. The industrial control software has been having an enormous increase in complexity as technology has developed and requires a systematic testing approach to enable efficient and effective testing in the event of changes. KDT has been proving that it is a valuable test method to support these test requirements [60]. Recent results from other researchers have shown that the design of the KDT test is complex with several levels of abstraction and that this design favours reuse, which has the potential to reduce necessary changes during evolution [26]. Besides, keywords change at a relatively low rate, indicating that after creating a keyword, only localized and refined changes are made. However, the same results also showed that KDT techniques require tools to support keyword selection, refactoring, and test repair [18].

2.3.1 Current Tools

Here are presented some of the most well-known test automation tools that implement the keyword-driven test framework, as well as their characteristics and comparisons between them. Finally, an analysis is made of the advantages and disadvantages of using this type of tools.

- Selenium: is an easy-to-use tool and is available for free under the Apache 2.0 license. It is an automated tool and is used to test existing applications on the web. This tool supports different browsers, namely the main ones (Google Chrome, Internet Explorer, Mozilla Firefox, etc.) [49]. In addition to being easy to use and being an intuitive tool, it does not require advanced technological or programming knowledge, and it also adapts to different programming languages [20]. It does not need installation since it works as an add-on to the browser. One of the disadvantages of this tool comes from the fact that it does not have its own repository for the objects, implying greater attention and care in its use [48].

- QuickTest Professional: belonging to the Hewlett Packard quality center, it belongs to the functional testing category. It has a wider range of supported platforms since, in addition to working as web testing, it works on the operating system itself, requiring a license for its use. It has some privations since it has its own IDE being exclusive to it, just accept the VBScript language and finally, it only works in the browser associated with Microsoft, Internet Explorer [56]. For a good understanding and use of this test tool it is necessary to have a prior knowledge of basic programming. QTP has its own object repository, but it nevertheless has many limitations when it comes to interacting with a cloud [31]. It has a commercial cost, having a proprietary license.

- TestComplete: is an automatic test tool that supports various types of applications from the web (Internet Explorer, Mozilla Firefox and Google Chrome), mobile and desktop to Java, .NET and WPF technologies [52]. This software monitors the most used keyword-driven operations and facilitates access to them, however for this to be possible the program must contain the Keyword test editor functionality [28]. Finally it is considered a very versatile tool as it supports a wide variety of programming languages from JavaScript, Python and Vbscript to C and C ++, however it only supports the Microsoft Windows operating system and presents a high-value commercial license [51].

- SilkTest: is a robust automation tool for testing web and corporate applications [14]. It is a functional testing tool as well as a regression testing tool that is economical and tests the aforementioned applications efficiently and intuitively. This tool saves time, tests scripts effectively and tests several platforms and also several browsers [36]. This software supports Windows only, but it nevertheless supports both the Windows-specific browser, Internet Explorer, as well as Mozilla Firefox. Unlike QTP, it supports several programming languages, from Java, 4Test, VBScript, C and VB.net. It also has a proprietary license at moderate cost.

- RobotFramework: has an open-source testing structure that uses the keyword-driven approach, that is, keyword-driven test for their automation, thus becoming an automatic testing tool [16]. In addition to the keyword-driven, this program has the ability to use behaviour-driven and data-driven test, however, it does not have support from popular IDE's which makes it difficult to debug the tests. It is also considered easy to install and use as it does not depend on extensive knowledge of high-level tests and provides several tools in several libraries that support the creation and execution of test cases [25]. It is a free tool under the Apache 2.0 license that was developed for Windows and Linux only and supports the most popular browsers, such as Internet Explorer, Mozilla Firefox and Google Chrome.

- Ranorex: is also an automated tool that tests web applications, with the main disadvantage of not being free. It is the best software within those mentioned, for easy installation for non-programmers. This tool can be used to identify objects, such as automatic object synchronization, use of the Xpath expression, features not provided by Selenium, for example. Also, it offers resources for reuse [25]. It is only existing software for Microsoft Windows, supporting different browsers: Google Chrome,

Safari, Microsoft Edge, Internet Explorer and Mozilla Firefox, without any kind of free license. In addition to the support of web applications, it also offers possibilities to work in applications of the operating system, as well as in terms of mobile applications, existing for Android and iOS. Like QTP, it has its own IDE, having the advantage of not being limited to a single programming language, and here, there is the possibility to program in C, VB.Net or Python. In addition to being an easy to use tool and very intuitive in its interaction, it has the ability to create tests without the use of any type of programming and supports the execution of several tests in parallel. Finally, it presents a shareable object repository [47].

Tool	**Operating System**	**Browser**	**License**	**Application Support**	**Allowed Languages**	**Cost**
Selenium	All	All	Apache 2.0	Web Application	Java, C#, Ruby, Python, PHP, Perl, ...	Free
QuickTest Professional	Microsoft Windows	Internet Explorer	Owner	Web and Desktop Application	VBScript	Commercial (Free Trial)
TestComplete	Microsoft Windows	All	Owner	Web, Mobile and Desktop Application	VBScript, Java, C++, Python	Commercial (High)
SilkTest	Microsoft Windows	Internet Explorer, Firefox	Owner	Web Application	Java, 4Test, VB, C#	Commercial
RobotFramework	Microsoft Windows, Linux	All	Apache 2.0	Web and Desktop Application	Java, Python, .Net, Pearl, PHP	Free
Ranorex	Microsoft Windows	All	Owner	Web and Desktop Application	C#, VB, Python	Commercial (Free trial)

Table 2.1: Comparison of analyzed tools

After analyzing and observing the characteristics of each of these tools, the following advantages and disadvantages are presented as a conclusion of their use:

2.3.1.1 Advantages

- Fast execution of test cases;
- Software testing in less time;
- All manual testing problems are solved by automated testing;

- Repeating test cases are handled in an easy way.

2.3.1.2 Disadvantages

- Sometimes some knowledge of programming and skill is needed to use these tools;
- Some tools have high purchase prices;
- Maintenance is a complicated task and can be expensive;
- They do not have general use, that is, some tools have key limitations in relation to browsers or operating systems.

2.4 Domain-Specific Language

DSL is a language meant to be used in the context of a particular domain. A domain could be a business context or an application context. A DSL does not attempt to please all. Instead, it is created for a limited sphere of applicability and use, but it's powerful enough to represent and address the problems and solutions in that sphere [19]. A DSL can be used to generate source code from a keyword. However, code generation from a DSL is not considered mandatory, as its primary purpose is knowledge. However, when it is used, code generation is a serious advantage in engineering. DSL will never be a solution to all software engineering problems [37], but their application is currently unduly limited by the lack of knowledge available to DSL developers, so further exploration of this area is needed [29]. Other researchers used DSL in CPS and left their testimony of how the specification language hides the details of the implementation. The specifications are automatically enriched with the implementation through reusable mapping rules. These rules are implemented by the developers and specify the execution order of the modules and how the input/output variables are implemented [9]. This allows the reuse of software components (e.g. modules or classes) and improves software productivity and quality [30].

2.4.1 ANTLR

ANTLR is a parser generator, a tool that helps you to create parsers [42]. A parser takes a piece of text and transforms it into an organized structure, a parse tree, also known as an Abstract Syntax Tree (AST) [55]. AST is like a story describing the content of the code, or also as its logical representation, created by putting together the various pieces [43]. Figure 2.1 shows the parsing process.

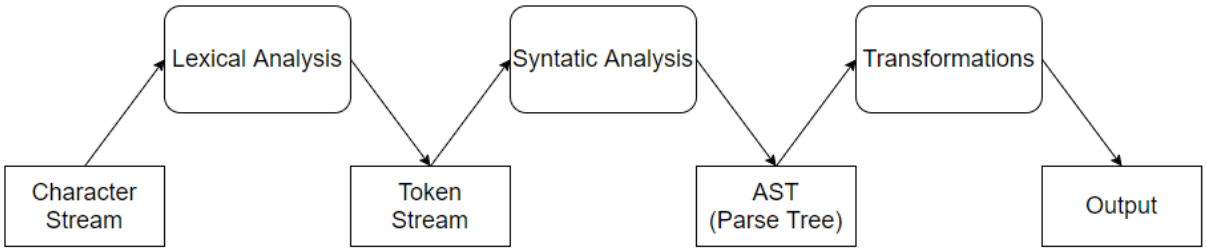


Figure 2.1: Block diagram of a standard Language Processor

The Parsing process shown in Figure 1 goes through three major phases explained below:

- Lexical Analysis:
  - It is performed by a component usually called Lexer, Scanner, Lexical Analyser or Tokenizer;
  - The Lexer reads and divides the input (character or byte stream) into tokens applying lexical rules;
  - Lexical rules are defined using regular expressions and aim to identify terminal symbols and specify tokens;
  - In the end, the Lexer generates a token stream as output.
  - Figure 2.2 shows the illustration of this process:

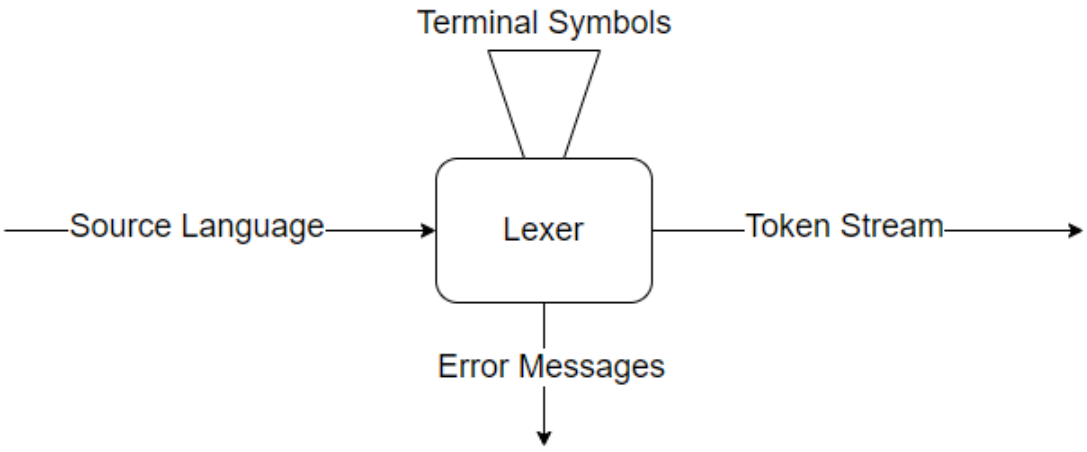


Figure 2.2: Block diagram of a Lexical Analyzer

- Syntactic Analysis:
  - It is performed by a component usually called Parser, Syntatic Analyser or Grammar;
  - The parser gives the token stream a structure by checking token order against structural rules;
  - These Structural rules define the order and structure of token combination;
  - In the end, the Parser generates a parse tree as output.

- Figure 2.3 shows the illustration of this process:

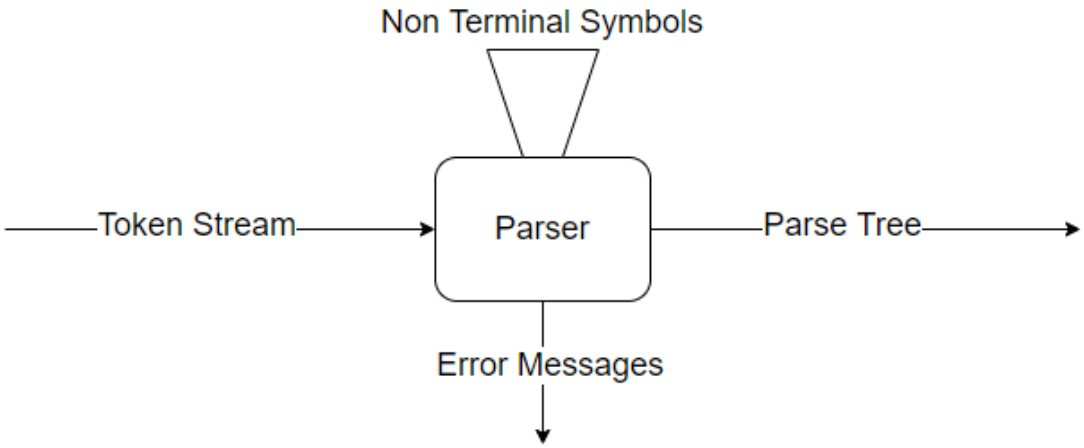


Figure 2.3: Block diagram of a Syntactic Analyzer

- Transformations:
  - It is performed by a component usually called Transformer or Walker and it follows the pattern Visitor or Listener;
  - The Transformer traverses the parse tree in order to produce some output;
  - The traversal defines an action for each node of the parse tree;
  - The action can output text (string) or any other complex object.
  - Figure 2.4 shows the illustration of this process:

Figure 2.4: Block diagram of a Transformer

ANTLR is a parser generator that uses ALL(*). It parses the input dynamically at runtime and uses a top-down parser left to right by constructing a Leftmost derivation of the input and looking any number of ahead tokens when selecting among alternative rules [44]. The Visitor pattern let us decide how to traverse the tree and wich nodes we will visit. It also allows us to define how many times we visit a node [41].

2.5 REST

REST is an architectural style for providing standards between computer systems on the web, making it easier for systems to communicate with each other[2]. It is a protocol developed specifically to create web

services, which aims to be platform and language independent, using Hypertext Transfer Protocol (HTTP) for communication between servers. This protocol is based on 6 fundamental principles[11, 12]:

- Client-Server Architecture - there is a separation of the different responsibilities of the system in modules, such as, for example, the separation between the system interface and the service modules;
- Stateless - all requests are processed independently, the server is unaware of the status of the client-side application;
- Cache - consists of storing data on the client-side, to reduce the need for requests to the server, optimizing the system;
- Uniform Interface - this architecture focuses on simplicity, accessibility, interoperability and the ability to discover resources;
- Layered System - the use of different layers allows the system to protect certain components, making them not all accessible in public;
- Code-On-Demand - which allows the server to provide the code transfer capability that can be used on the client.

2.6 Discussion

In this chapter, the state of the art of this dissertation was presented. A review of the current state of research development in CPS, as well as exposing the proven importance of test automation. The concepts of the KDT framework were presented since it will be a key element in the development of this work as well as some tools that are examples of its use today. The topic DSL was also addressed, which will also be another key element of this work. Within the domain of DSL, it was explained how ANTLR works internally. Finally, the REST protocol was presented, as well as the reasons for being, today, one of the most used protocols in the construction of web services.

This state-of-the-art review revealed that, although there are several tools dedicated to the management and execution of tests, none of them are intended for testing or, more specifically, for self-diagnosis of CPS. Therefore, the goal of this work will be, taking advantage of those that are the best characteristics of the automatic test systems, to develop a system aimed at the self-diagnosis of the CPS itself.

That said, having the necessary theoretical knowledge already, the next chapters will focus more on the process of modelling, developing and implementing the system. Starting with the explanation of the requirements, where all the clauses that the system must comply with will be mapped, followed by a specification and analysis of the technologies to be used as well as the system architecture, ending with the implementation chapter, where all the work will be explained.

Chapter 3

Analysis and Specification

The purpose of this chapter is to provide a detailed analysis of the requirements that the system must satisfy, which were also accompanied by the requirements of CONTROLAR since the system will be developed for application in its CPS. After the analysis and specification of the requirements, the presentation of the system structure and the technologies to be used for the development of the system follows.

3.1 Requirements

The main objective of this work is the construction of a system that allows the self-diagnosis of the CPS through the execution of tests on it. Therefore, this system must be also able to manage the tests performed and also the results obtained. Thus, system requirements include:

- General requirements
  - Unlimited number of errors (scalable solution) - The system must be able to define an unlimited number of errors and be scalable to be able to update in the future with new situations that appear;
  - Modular tests at different levels - The system must be able to execute several test levels (physical and logical) in a modular way;
  - Ability to self-assess (frequently) - The test equipment can run the self-diagnosis system periodically or on request;
  - Communication and assured integration of electronic tests with the system - The system will have to communicate bi-directionally with the Controller / Server of the electronic tests to receive inputs related to the test sequencing for diagnostics of the machine and send the test results back;

  - Universal - The system should work on any operating system without compromising its reliability;
  - Intuitive interface - The system must have an interface that allows the user, in a simple and intuitive way, to use it unequivocally.

- Specific requirements
  - Multiple selection of tests to run - The user selects the test set he wants to run;
  - Test execution request - The tests selected by the user are performed;
  - Running a test suite - Trigger the execution of a set of tests performed by the Server / Controller of the electronic tests through the communication established between this and the system;
  - Run remote and local tests - The system should allow the user to perform local tests (machine) or remote tests (car radio);
  - Schedule a test suite to run - The user can schedule the time to run a set of selected tests;
  - Presentation of test results - After the execution of the tests and the communication of their results to the system, by the Controller / Server of the electronic tests, the results obtained for each of the tests performed are presented to the user;
  - Consultation of documentation for each test - The user can consult the documentation and information (metadata) associated with each of the tests in the system;
  - Consult the history of tests performed - The history of the test reports performed must be available to the user for consultation;
  - Automatic update of tests available on the system - The system should allow adding, updating and removing of tests available on the system automatically;
  - Provide a specific language for creating new test packages - The system should allow using a specific language for programming test packages;
  - Test package creation - The user must be able to create new test runs, specifying those runs using the language provided by the system (this will be available as a file for execution, planned or on request, in the same way as the tests available in the system).
  - Add new test packages to the system - The user can add new test packages to the system;
  - Update test packages in the system - The user can update test packages existing in the system;
  - Remove test packages from the system - The user can remove test packages from the system;
  - Export executions reports to CSV file - The user can export the data related to the execution reports to a CSV file, specifying the start and end dates of these reports or exporting all available ones;

  - Perform system backups - The user can perform backup copies of the system at any time, being able to select and specify the data he wants to copy;
  - Restore system backups - The user can restore a system backup at any time, correcting possible failures or corrupted data;

- User types and permissions
  - Industrial operator - The industrial operator is the most basic type of user in the system. He is responsible for realizing the tests to the car radios and the machine in the industrial day-to-day. This user will only have access to the so-called execution mode, in which he will only be able to execute tests and packages. In this mode, the user must not have access to any other functionality so that it is not possible to compromise the system;
  - Test Manager / Administrator - The test manager or administrator, as the word suggests, will be responsible for managing the system and all of its settings, in addition to any tests that the system will have available for execution. This user has access to all system resources. This user has access to all the features of the system and will be someone with more technical capabilities, so he has a more sophisticated environment, but maintaining the graphic simplicity that characterizes an industrial production environment.

3.2 System structure

After elaborating the system requirements, this section aims to present an overview of the system structure. In the requirements analysis, the solution chosen for the structure of this system is a client-server architecture, based on the REST model. The client-server model aims at the division of tasks to reduce the system load. The server will offer a series of services to a specific user, that is, an Application Programming Interface (API), and will perform the tasks requested by the user and return the data. On the other hand, the client is responsible for requesting a specific service from the server, through messages. This is the most used model in the web community since it is possible to run several clients without compromising the operation of the system.

The system will then be composed of the web client and the server. The web client will communicate with the server via HTTP requests and, in turn, the server will respond via HTTP responses. The server will be responsible for connecting to the database, making the necessary transactions or queries and returning the results.

Figure 3.1 represents an illustration of the system structure.

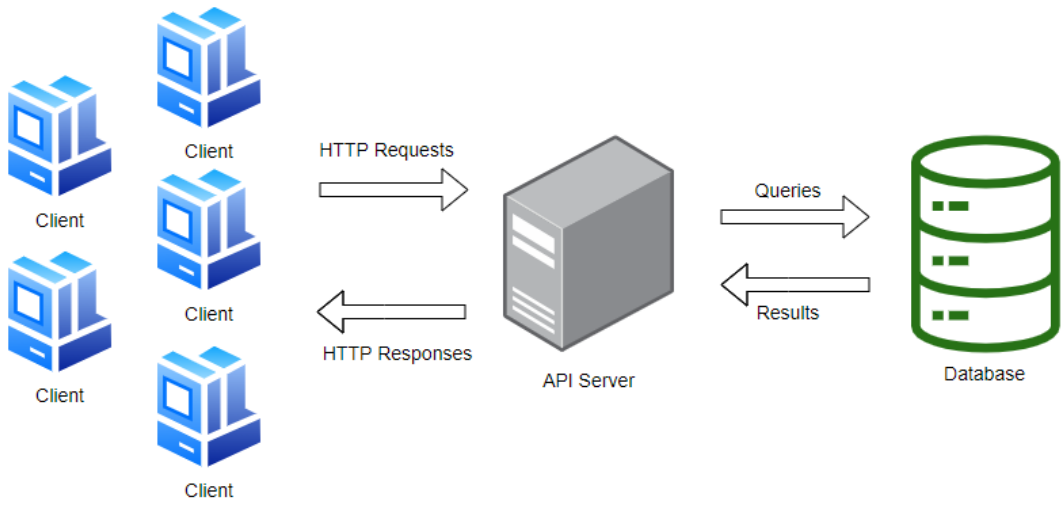


Figure 3.1: System structure - REST model

3.3 Technologies to use

After analyzing the requirements and the structure of the system, this section describes the chosen technologies for the system development and implementation. As we saw in the previous section, the system can be divided into two main components. The component that represents the Client-side will be called Frontend and the component that represents the Server-side will be called Backend.

3.3.1 Backend technology

This component of the system will include the API Server and the database that will be used to ensure the consistency of the system data. Once it was decided that the system should be implemented as a web application, it makes sense to use a platform that works in the web language, JavaScript [45]. For this, Node.js [15] was chosen, which consists of a development platform based on Google JavaScript Engine V8, which allows the JavaScript language to be used on a web application server.

The main objective of application development in Node.js is to be able to create fast and scalable applications that, at the same time, perform well with low memory consumption [54]. For that to happen, the platform is based on 3 concepts:

- Non-blocking I/O;
- Asynchronous event-driven programming;
- Single thread.

Unlike other development frameworks, Node.js does not rely on multithreading to run simultaneous processes, operating only on a single thread, using an event-based approach and non-blocking input and

output. The use of non-blocking I/O operations prevents the application from being blocked when calling any operation. This means that the server never waits for data to be returned. These types of attributes make Node.js lightweight and efficient compared to other structures that use I/O blocking operations [33].

In figure 3.2, it is possible to observe the processing of operations in Node.js:

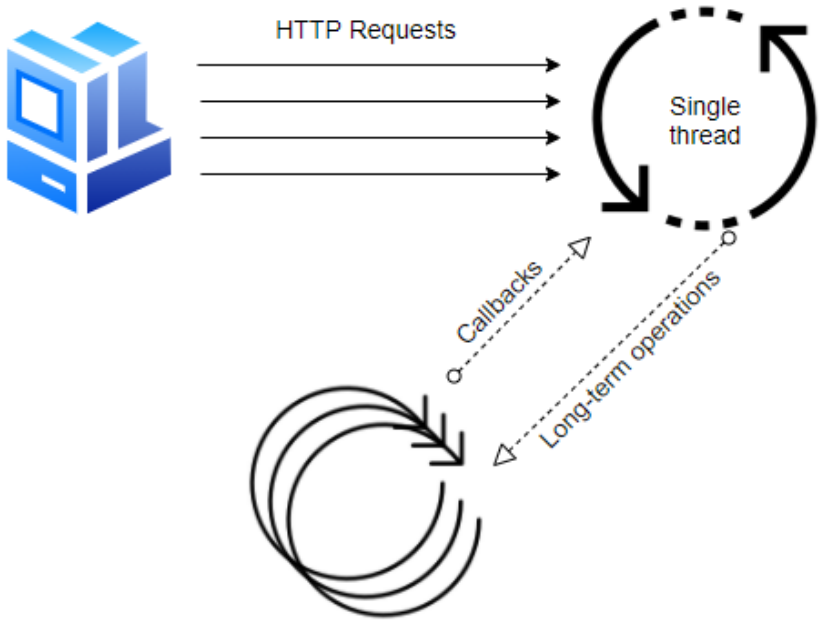


Figure 3.2: Processing single-thread operations in Node.js

As we can see in figure 3.2, several I / O operations can occur in parallel and the respective callback of the event will be invoked as soon as the operation is finished. For this to happen, an event loop is required, which is a mechanism that will perform two tasks in a continuous cycle: event detection and triggering event handlers. The event loop is just a thread running within a process, so we can draw two conclusions: only one event handler will be running at the time and any event handler will run until the end, without being interrupted [1].

To maximize the potential of Node.js, we use the Express Framework, which provides a robust set of features for web applications and will make the system more flexible. This framework allows the creation of an API in a very fast and simple way, having a multitude of middleware methods at your disposal [13]. In summary, this framework helps us to maximize our API developed in Node.js.

For data consistency, the MongoDB database was chosen. MongoDB is a document database, which means that it stores data in JavaScript Object Notation (JSON) type documents. As the data will always be traded as JSON documents, the use of MongoDB will maintain this consistency in a more natural, expressive and powerful way than any other model [53]. The way to ensure the best communication between the API and the database will be through Mongoose, which is an Object Data Modeling (ODM) library for MongoDB and Node.js [21]. It manages relationships between data, provides schema validation and is used to translate between objects in code and the representation of those objects in MongoDB.

In figure 3.3, we can see this mapping of objects between Node.js and MongoDB through Mongoose.

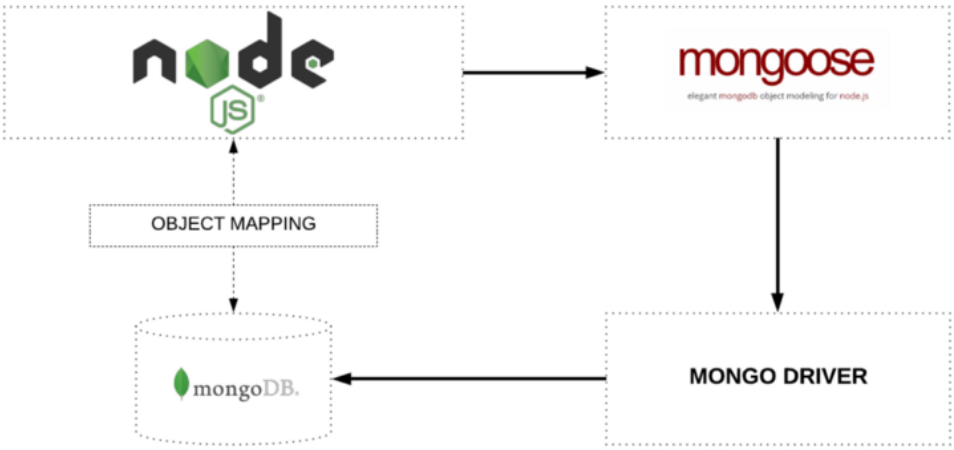


Figure 3.3: Object Mapping between Node.js and MongoDB managed via Mongoose

3.3.2 Frontend technology

For choosing the Frontend technology, obviously, there are many options, many of which are based on JavaScript and all are intended to assist in the process of building user interfaces. For this system, the tool chosen was React.js [23] due to its ability to interact with applications developed with the technologies chosen for Backend [46].

In React.js it is easy to create an interactive UI because it projects simple visualizations for each application state, efficiently updates and renders the right components when their data changes. These declarative views make the code more predictable and easier to debug [3]. React.js allows the creation of encapsulated components that manage their own state and through the composition of these components, it is possible to create more complex interfaces with less complexity in the code.

For the creation of components, it is still necessary to understand the basics of 5 more technologies with which we form the components in React.js:

- HTML - is a web page development language and is the main markup language on the WWW. Through HTML it is possible to create web pages, which are essentially HTML documents that are interpreted by browsers [58]. Its syntax is quite simple, based on tags, as in Extensible Markup Language (XML), where each tag will represent an element of the page, as can be seen in figure 3.4.

```
<html>
    <head>
        <title>My Page</title>
    </head>
    <body>
        <h1>My Heading</h1>
        <p>My paragraph.</p>
    </body>
</html>
```

Figure 3.4: HTML simple structure example

- Cascading Style Sheets (CSS) - is a language used to describe the appearance of HTML elements [59], that is, through CSS it is possible to manipulate the various HTML elements so that the appearance of the application is as expected. It is an easy-to-use technology, simpler than similar ones, with agile design and no license to use.

- JavaScript - is an interpreted and object-oriented programming language [38]. Initially, it was created to be executed only on the client-side, that is, in browsers, but nowadays it is also used on the server-side, as in Node.js. The use of this technology will be relevant both on the Node.js server-side and on the React.js client-side, as it is through it that the logic of the components will be developed.

- JSON - is a format for exchanging data. Its structure is based on the key-value relationship, that is, for each value represented, a key is assigned [27]. This format is easily interpreted by humans and machines. In this case, this technology will be particularly important since it will be used to exchange data both between the server and the client and between the server and the database.

- HTTP - is a communication protocol that aims to exchange data between the client and the server. In this protocol, the React.js server sends requests, HTTP Requests, to the Node.js server, which after executing the tasks, responds with HTTP responses with the data.

Since the component's logic is written in JavaScript instead of models, we can easily pass data through the application and maintain the state outside the DOM [24]. The React virtual DOM allows the implementation of some intelligent alternative solutions that guarantee the quick rendering of the components, which is necessary since the data has to be presented immediately and effectively. In these situations, React finds an ideal way to update the UI and all you need to do is provide the data flow through the API [40]. Basically, the React virtual DOM acts as an intermediate step whenever there are changes in the web page and allows the renderings to be faster and more efficient, making the pages highly dynamic.

We can see the difference between Real DOM and React Virtual DOM in figure 3.5.

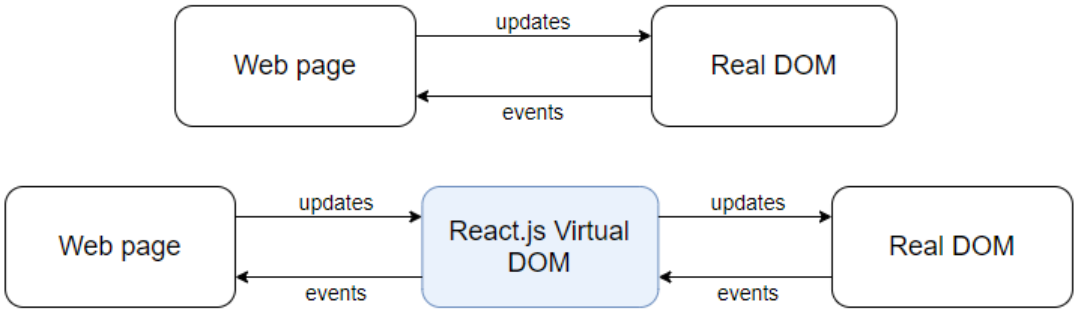


Figure 3.5: Real DOM and React Virtual DOM

3.4 Discussion

As mentioned, the system analysis and specification were carried out in this chapter. The system requirements have been divided into three parts to allow a better interpretation in the development of the system. A basic system structure was defined, separating it into two, the API Server, responsible for accessing the data and implementing the logic, and the Client, responsible for the user interface. Finally, the technologies chosen for the development of the system were presented and explained. For the server-side, MongoDB, Express and Node.js were chosen and for the client-side, React.js was chosen. These technologies, due to their recurring use together in the web community, are also known as MongoDB, Express, React, Node (MERN) Stack.

Having all the objectives well defined and the necessary theoretical knowledge about the structure and technologies to be used in the construction of the system, in the next chapter the final abstraction of the entire system will be presented in the form of general and detailed architecture for it.

Chapter 4

Architecture

In this chapter, the general architecture of the CPS will be presented. However, as this architecture encompasses several diversified components, its modelling was divided into three phases. In the first phase, the part of the architecture that concerns the management and configuration of the system tests is explained. The architecture of the system to be developed is explained below. In the end, the general architecture of the CPS is presented.

4.1 Test Management and Configuration Architecture

The architecture that we propose in this section aims to automate and facilitate the process of creating new tests. In this architecture, we use the KDT methodology in conjunction with a DSL. To fully understand the architecture and how its components interconnect, it will be explained first how KDT and DSL are applied and only later how they are integrated into the same architecture.

4.1.1 Keyword-Driven Testing Methodology

KDT will be used to abstract low-level code scripts, associating each script with a keyword that will represent it most descriptively and explicitly possible. In this way, the user does not need to know the details of the script implementation, but only what it does. We will also associate each keyword with metadata related to the corresponding test, which will be stored in a database. Figure 4.1 represents the approach taken in using KDT. The names given to the tests in the figure are only fictitious names to show that the names given to the keywords must be as descriptive as possible.

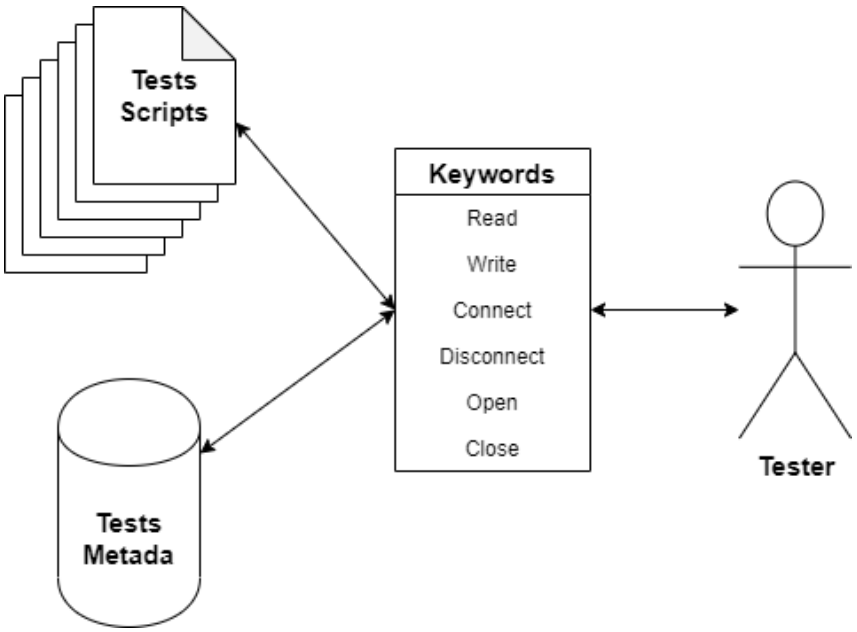


Figure 4.1: KDT Approach

In Figure 4.1, we see a stack of scripts that represent the test scripts that already exist in the system, in this case, primitive tests that focus only on testing a feature or set of features, as long as they can be well-identified only by a word that can serve as a keyword. We see also the representation of a database that will be where all the information and metadata about the tests existing in the system will be stored, that is, the same ones that are represented in the stack of scripts. Connected to the database and the stack of scripts, we see a table with keywords in which each keyword represents all the information related to a test. This table is the most relevant element in the figure because it is where we can relate all the information from the stack of scripts and the database, and this is done with just one word that concedes testers with little programming knowledge to interpret what each test does or means. Finally, we have the link between the Tester and the keywords table that demonstrates the Tester will only have access to the keywords, without needing to know any details of implementation.

4.1.2 Domain-Specific Language

This use of KDT alone does not bring great advantages, as we still need someone to design a test execution flow according to its purpose. This is where the importance of DSL comes in, as it allows to define a friendly language for testers, without the need for very sophisticated programming knowledge. The proposed language is extremely simple but allows the creation of new scripts with new execution flows and logical rules applied. This is achieved only using the keywords defined by the KDT and some terminal symbols defined in the DSL. Table 4.1 shows the symbols of the defined DSL terminal and what they represent.

Symbol	Description
keyword	Catches the keywords in the script
->	Catches the "next"symbol, which means that after that symbol the next block to be executed arrives
(	Catches the opening parenthesis
)	Catches the closing parenthesis
?	Catches the conditional expressions from the script
:	Catches the next block of code to be executed when a condition is false
&	Catches the logical operator that means intersection
|	Catches the logical operator that means union
;	Catches the end of the script

Table 4.1: DSL Symbols Description

4.1.3 Proposed Architecture

To achieve the full potential of the integration between KDT and DSL, a final abstraction of all these processes is necessary. Figure 4.2 presents the architecture that guarantees to abstract the entire complex process of creating new tests for the system, thus giving the possibility to users less endowed with programming knowledge to be able to build new tests.

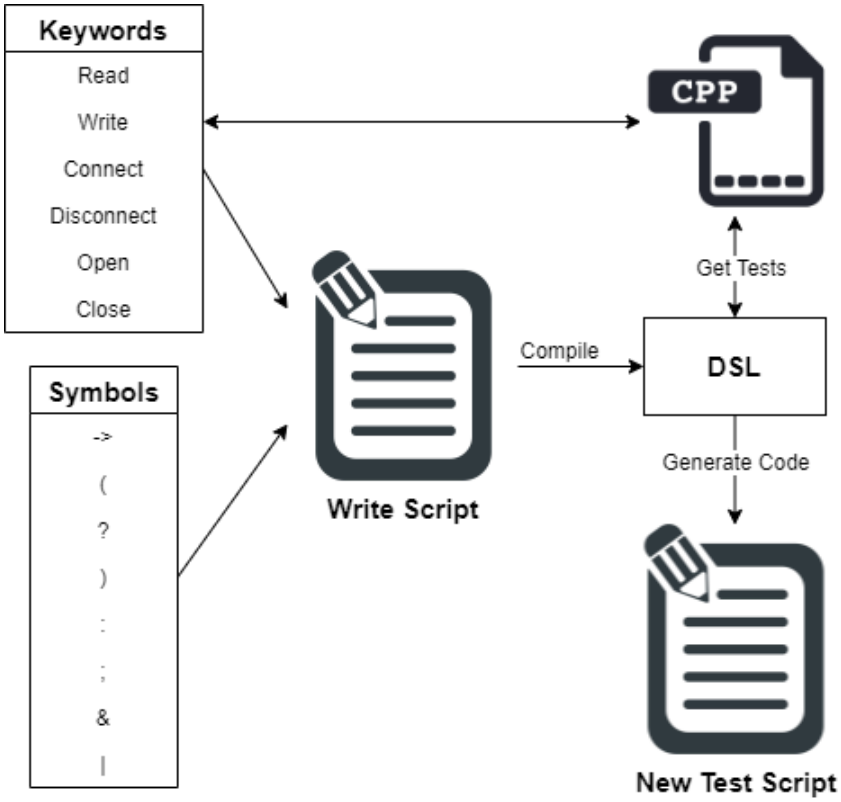


Figure 4.2: Proposed architecture for KDT with DSL

The two tables illustrated in Figure 4.2, `Keywords` and `Symbols` represent the elements that can be used to form new test scripts. The elements present in the `Keywords` table are the keywords that correspond to the tests defined and available in the system to be used in the creation of new tests. It is

also possible to verify the connection between the existing tests programmed in lower-level languages, such as C++, with the `Keywords` table. The elements in the `Symbols` table contain the terminal symbols of the defined DSL, that is, the only symbols recognized by the DSL. These allow giving logic and organization to the new tests of the system. Therefore, it is possible for the Tester, with the elements available in these two tables, to write the new test script and this is what is represented with the connections between the `Write Script` element and the tables.

As soon as a new test script is written, the DSL will analyze it, using a Lexer and Parser, and verify that it is syntactically and lexically well written. This step is represented in the `Compile` connection. If the script complies with the defined rules, the DSL will compile that script and generate the code for a new test. But, for that, it needs to have access to the code of the tests that were used through the keywords and that is what is represented with the connection `Get Tests`. At the end of this process, the DSL will be able to generate the code for a new test. This is represented in the `Generate Code` link. From that moment, the new test is available for execution in the system.

4.1.4 Example of Application

In this section, a complete example of creating a new test with this architecture is presented to demonstrate its simplicity and efficiency. In this example, it is assumed that the scope of the tests will be the same as shown in the `Keywords` table in figure 4.2 and the symbols that we can use are those shown in Table 4.1. The first step is to compose the script with the keywords and available symbols. In this example, we will use the following script:

```
( Connect & Open ) ? Read -> Write -> Close : Disconnect ;
```

Here the scripts corresponding to the keywords `Connect` and `Open` will be executed and if both return a true value the execution will follow to the block just after `?`. If any of the scripts return a false value, the next block of execution will be the one after the `:` symbol. The block after `?` will execute the three scripts corresponding to the keywords `Read`, `Write` and `Close` sequentially in the order they are specified in the script. The block after `:` will execute only the script corresponding to the keyword `Disconnect`.

Once the script is written, it will be analyzed by Lexer that will verify that all the elements that are in the script are part of the language. In this case, all symbols will be recognized successfully and then it is the time for Parser to continue with his analysis and check that all the rules of phrase formation are respected. After these checks, if the script is written correctly, it will be compiled and generated the new source code for the new test script. The source code of the new test will be based on the source code of the tests that were used with the keywords but adding the logic applied with the symbols used in the script.

4.2 Self-diagnosis Tests System Architecture

In this section, the architecture for the self-diagnosis tests system will be presented. To obtain a complete understanding of this architecture, it was decided to divide it into 3 tiers, Frontend, Backend and Database. In each tier, the internal components will be presented and explained.

4.2.1 Frontend

In this subsection, the Frontend tier of the architecture is presented and explained. We can see this in figure 4.3.

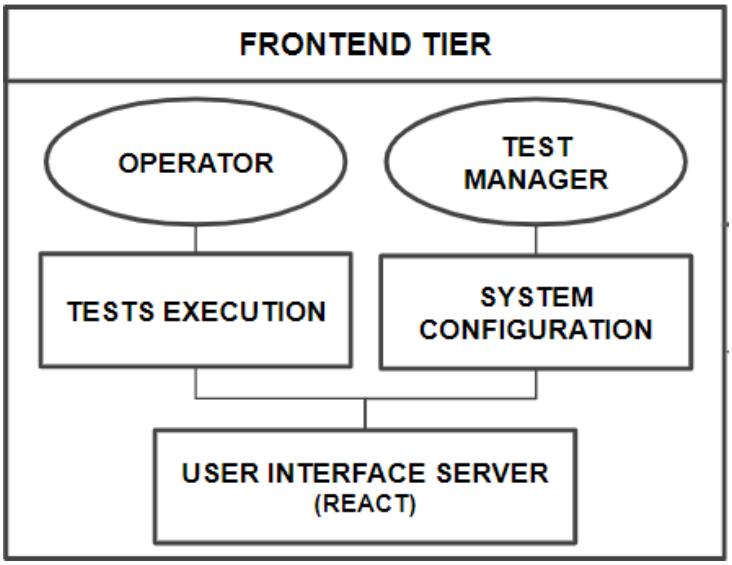


Figure 4.3: Frontend tier of the architecture

In this tier, we have two first elements, `OPERATOR` and `TEST MANAGER`, which represent the two types of users that the system has. Therefore, according to the permissions of each one, this tier makes available to each user the respective interface that will give access to the realization of the functions of each one in the system.

The two elements below in the tier, `TESTS EXECUTION` and `SYSTEM CONFIGURATION`, represent the different interfaces that each user will have access to. In this case, the `OPERATOR` type user will have access to the system `TESTS EXECUTION` mode and the `TEST MANAGER` type user will have access to the `SYSTEM CONFIGURATION` mode. The differences between these two users are listed and explained in section 3.1 and the interfaces to which they correspond must conform to those specifications.

The last element of this tier, `USER INTERFACE SERVER`, represents the logic of the Client. It is in charge of implementing any logic that exists in this tier, such as, for example, providing an adequate interface for the type of user that must comply with it or even the manipulation of data in the formation of web pages. It is also this server that establishes the connection to the Backend tier, making HTTP requests to request data or actions, receiving and validating data that arrives through HTTP responses.

4.2.2 Backend

In this subsection, the Backend and Database tiers of the architecture are presented and explained. We can see this in the figure 4.4.

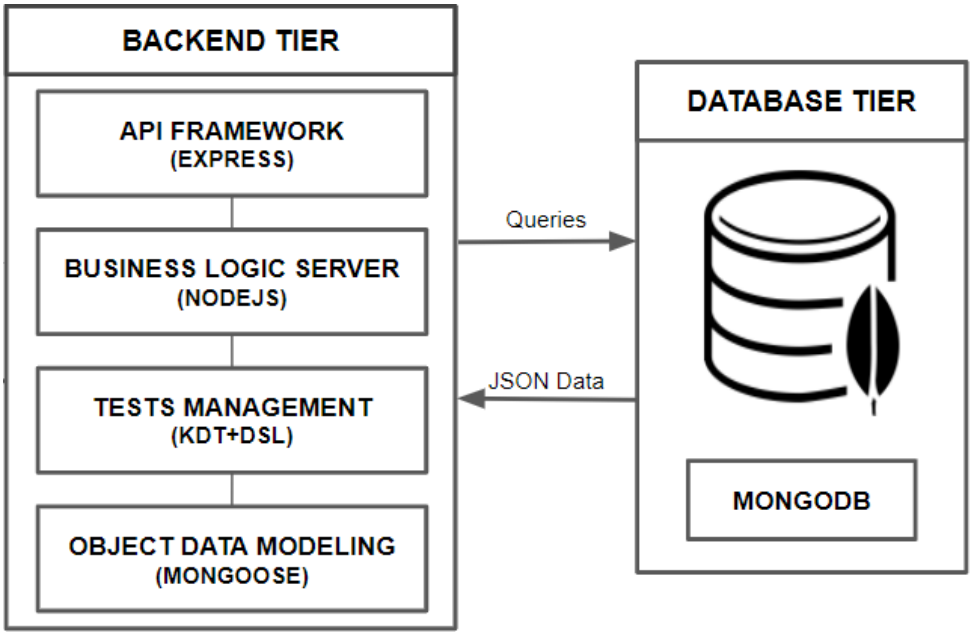


Figure 4.4: Backend and Database tiers of the architecture

The Backend tier, unlike what was done in the Frontend tier, will be analyzed from the bottom up, as it will be understood more intuitively. In this tier, we start by looking at two elements in parallel. The `OBJECT DATA MODELING` element represents the module responsible for establishing the connection between this tier and the Database tier, that is, it is this module that performs the queries and receives data from the database. Element `TESTS MANAGEMENT` is responsible for the acquisition and management of the primitive tests of the system and the configuration of new test suites for the system, using the KDT methodology and a DSL. This component represents the previous architecture, presented and explained in section 4.1, which is now integrated into this tier of the system. Above, we see the `BUSINESS LOGIC SERVER` element that represents the Server that implements all the logic of this tier. This component is responsible for executing the tests and for the internal organization of all other components of this tier. Manages all data arriving at the system, guaranteeing its integrity, and also provides the routes or services through which this tier responds to Clients requests. The last element of this tier, `API FRAMEWORK`, is responsible for building and making the REST API available to the Client. This element implements the routes that are created in the `BUSINESS LOGIC SERVER` element and, in this way, the Client can make HTTP requests to the Server.

Finally, it remains only to present and explain the Database tier, which is also the simplest tier of this architecture. It consists of the system database, which is a document database that stores documents in JSON. All data sent to the Backend tier, via `OBJECT DATA MODELING`, is in JSON, which is an advantage because all data processing and manipulation in the system is always done in this format.

4.2.3 Proposed Architecture

After presenting and explaining all the tiers of the system, we can now form the architecture by joining the tiers according to the structure that was previously defined for the system. It should be noted that this architecture corresponds to the self-diagnosis tests system that will later be integrated into a CPS and, therefore, the physical components referring to the CPS and external software components will not be represented here. Figure 4.5 shows the system architecture.

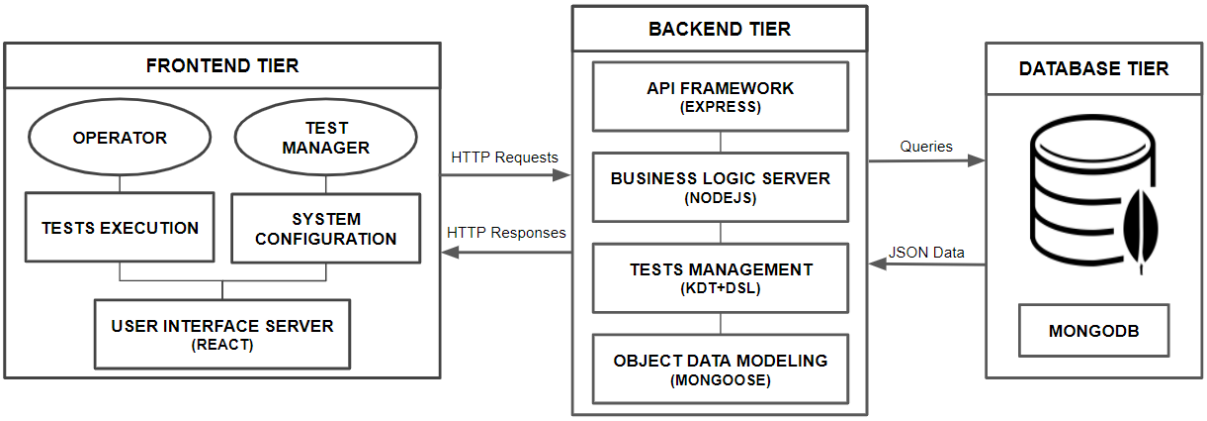


Figure 4.5: Proposed architecture for self-diagnosis tests system

In this architecture are represented the 3 main tiers of the system, already explained in the previous sections. The Frontend tier, responsible for the Client-side, communicates with the Backend tier, responsible for the system logic, through HTTP requests and responses. The Backend tier communicates with the Database tier, responsible for the consistency of the system data, through requests in the form of queries and responses in the form of JSON documents, all processed and validated through the `OBJECT DATA MODELING` module, mentioned and explained above.

The most important point of this architecture, which sets it apart from the rest, is the inclusion and integration of the Test Management and Configuration Architecture, proposed in section 4.1, which, together with the other components of the system, will allow the system to guarantee its integrity and functionality through self-diagnosis tests in real-time and also configuring new tests for the system with much less complexity.

4.3 General Architecture for Cyber-Physical System

In this section, the final CPS architecture is presented and explained, where we integrate all its components with the self-diagnosis tests system. This architecture aims to allow the CPS to obtain the ability to diagnose itself and, thus, be able to identify the failures in case of any internal error. The architecture, being the final abstraction of the system, can be seen in figure 4.6 and is explained below.

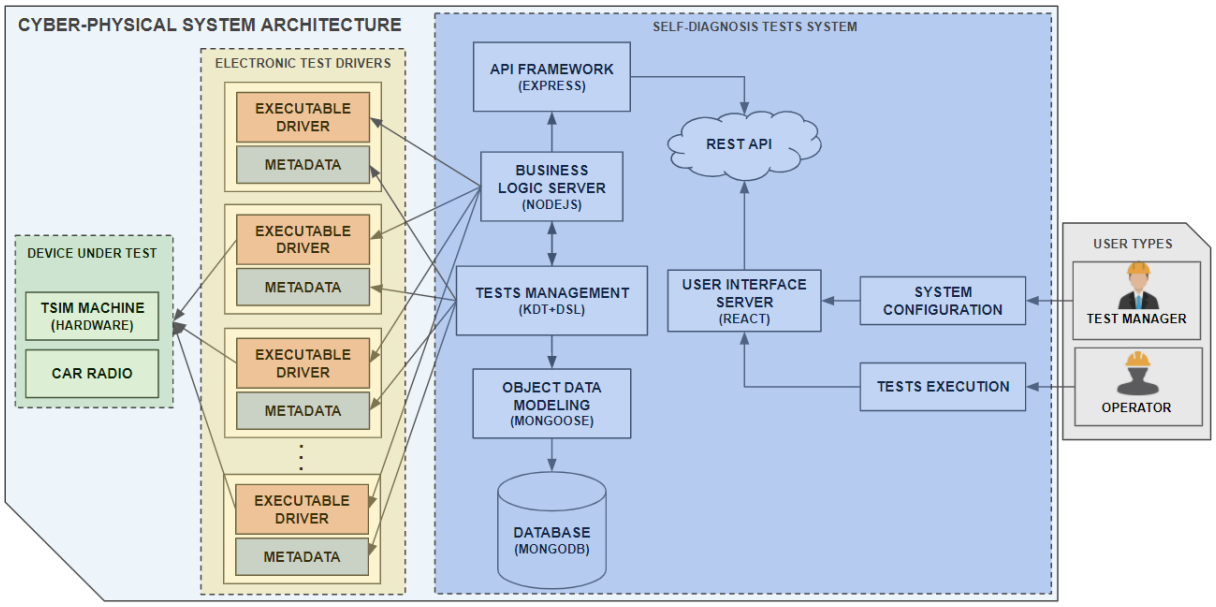


Figure 4.6: Proposed architecture for a self-diagnosis test system integrated with the CPS

In this architecture, we can easily identify 4 groups of components in which three of them will form an integral part of the CPS: Devices Under Test, Electronic Test Drivers and the Self-Diagnosis Tests System. The last group will be an important intervenient, but it is not an integral part of the CPS, the User Types. Each of these groups will be explained in detail, as each has its particularities.

The Devices Under Test group contains, as the name implies, the devices that can be subjected to tests which are the car radios and the machine itself. The elements `CAR RADIO` and `TSIM MACHINE` represent the two types of devices, the car radio and the machine, respectively.

The Electronic Test Drivers group is responsible for the primitive tests of the system, which in this case will be mostly electronic tests, but which can be any type of test as long as they respect the same integration format. Each element of this group must respect the following format:

- `EXECUTABLE DRIVER` - Provides an executable driver file to run that will contain several primitive tests that can be run and test the Devices Under Test;
- `METADATA` - Provides a metadata file that contains all the information about the tests that the driver can perform.

The Self-Diagnosis Tests System group is where the system developed in this work is represented, which will allow users to manage and execute the system tests. This system will be fed with primitive tests from the group of Electronic Test Drivers. The `TESTS MANAGEMENT` element is responsible for loading all the metadata of the primitive tests, available in the `METADATA` files of the Electronic Test Drivers group, and managing them so that they are saved in the system database and are available for execution. The link element with the system database is the `OBJECT DATA MODELING` that will make the connection and handle queries and transactions to the database, which is the `DATABASE` element. This test management is done through the KDT methodology, explained previously, and the configuration of new test

suites made through the developed DSL also explained previously. The tests will be performed by the BUSINESS LOGIC SERVER element, which will receive the execution orders from the end-user and proceed with the executions. The way to do this is to execute the drivers that are available as executable files. This Server will know which tests are available to execute on each driver since the TESTS MANAGEMENT element has already collected the metadata of all drivers and at that moment made available for execution, all the tests contained therein. This entire organization is orchestrated by the Server, which is responsible for the logic of the system and is represented by the element BUSINESS LOGIC SERVER. This Server not only controls all the data and logic of the system but also defines the routes and types of requests that can be made by the Client-side. It defines the services that will be available and this is called an API. The API FRAMEWORK element is responsible for creating and providing a REST API for any client to access, but obviously with the appropriate permissions, also defined by the BUSINESS LOGIC SERVER.

In this system architecture, USER INTERFACE SERVER represents the Client-side, that is, it is the server responsible for creating the web interface for end-users. It makes HTTP requests specifying the services, through routes, that it wants to access, to obtain the data it needs for its pages. Two types of interfaces are available, the execution interface, represented by the TESTS EXECUTION element, and the test and configuration management interface, represented by the SYSTEM CONFIGURATION element. Each of these interfaces will have its correspondent as a user, which brings us to the last group specified in the architecture, the User Types.

This group is represented by the USER TYPES element and represents the different types of users of the final system. The first and most basic type of user is the OPERATOR, that is, the industrial operator who is working and commanding the CPS and performs only the tests or test packages of the system. The second type of user, already more sophisticated, is the TEST MANAGER, who is someone with the responsibility of managing the entire system, using the appropriate interface for that.

4.4 Discussion

As mentioned, in this chapter the architecture of the system was presented and explained, dividing it into 3 stages. In the first step, the architecture for managing and configuring the system tests was built. In the second step, the first architecture is presented as an encapsulated element that integrates the architecture of the self-diagnosis tests system. This architecture represents the system architecture to be developed in this work. Finally, the architecture of the CPS was composed, which integrates all the components of this system, including the architecture of the self-diagnosis tests system. We opted for the greater detail of the architecture of the self-diagnosis tests system and the test management and configuration architecture because in the context of this work they are the focus of development.

The defined architecture represents an innovation for research in self-diagnosis tests systems and CPS, as it will allow the joining of these two types of system in one. Although the focus of the architecture is the application in a CPS, it is also applicable to any type of system, since the test feed is what defines the type of tests that will be performed on the system and, therefore, is generic to accept any level of testing.

This architecture follows the separation and structuring in microservices, which also allows the execution of the tests to be carried out remotely or by any other system with access permissions to the API provided by the Server.

In the next chapter of this dissertation, having already gathered all the necessary information and knowledge, the entire process of implementing the system will be presented.

Chapter 5

Implementation

This dissertation aims to create a self-diagnosis tests system that will be an integrated application in CONTROLAR's cyber-physical machines that will allow its self-diagnosis in real-time. The proposed architecture for the self-diagnosis tests system allows the management, configuration and execution of system tests, presenting a modular and extensible model that allows exploring different levels of tests to be performed on the devices under test.

This chapter describes the implementation of the system and its validation. Thus, Section 5.1 explains each collection of data maintained in our database. Section 5.2 describes the Backend tier where the system logic is, including the management of the system data and the configuration and execution of the tests. Section 5.3 describes the Frontend tier that contains the user interface and the different features available for each type of user. Section 5.4 presents the results obtained from the validation performed to ensure the correct functioning of the system. Finally, section 5.5 discusses the implementation of the system and the results that have emerged from it.

5.1 Database

For the database, MongoDB was used, which is a document database, that is, it stores the data in the form of JSON documents. According to the data that the system needs, 5 collections of data have been identified to be stored in the database: Configurations, Tests, Packages, Reports and Schedules. Each of these collections contains specific attributes and will be explained in detail.

The configuration collection contains attributes about some configurations that may differ from machine to machine and are necessary to ensure the correct functioning of the system. The attributes of this collection are specified and explained in table 5.1:

Attribute	Description	Value Type
_id	This attribute is the unique id of the system configurations	String
dbBinDir	This attribute contains the MongoDB bin directory, which contains the executables needed to perform exports and imports to the database	String
appDir	This attribute contains the directory where the system folder is located	String
backupDir	This attribute contains the directory to which system backups will be exported and imported	String
userConfigurationID	This attribute contains the id that allows access to the system configurations	String

Table 5.1: Attributes of configurations collection

The tests collection stores all metadata for the system's primitive tests. This metadata is provided by those who create and make the primitive tests available, so they are only imported into the system database and updated whenever there are changes. The attributes of this collection are specified and explained in table 5.2:

Attribute	Description	Value Type
_id	This attribute is the unique id of each primitive test in the system	String
active	This attribute has a true or false value, depending on whether the test is active, that is, available for execution, or if it is not	Boolean
id	This attribute is the test id provided by the team that develops the primitive tests and will only be used in the system to call the test execution	String
module	This attribute contains the name of the driver that must be executed to call the test execution	String
name	This attribute contains the name of the test	String
description	This attribute contains a description of the test	String
defaultParam	This attribute contains the parameter that, by default, must be used when performing the test	String

Table 5.2: Attributes of tests collection

The packages collection stores all metadata for the new test suites that are created in the system from the primitive tests. The attributes of this collection are specified and explained in table 5.3:

Attribute	Description	Value Type
_id	This attribute is the unique id of each test suite in the system	String
active	This attribute has a true or false value, depending on whether the test suite is active, that is, available for execution or not	Boolean
name	This attribute contains the name of the test suite	String
description	This attribute contains a description of the test suite	String
code	This attribute contains the test suite code script developed by the system test manager. This code is developed according to the rules of the language specified in the system	String
path	This attribute contains the name of the executable file generated to run the test suite	String
tests	This attribute contains the list of primitive test ids (_id) that are used in the test suite. They work as references to the primitive tests.	Array

Table 5.3: Attributes of packages collection

The reports collection stores all reports of execution of primitive tests or test packages in the system. The attributes of this collection are specified and explained in table 5.4:

Attribute		Description	Value Type	
_id		This attribute is the unique id of each report in the system	String	
id_user		This attribute is the id of the user who performed the execution	String	
date		This attribute contains the execution date	String	
results		This attribute contains the list of results of all tests performed, each with its own attributes	Array	
results	id_test	This attribute is the unique id (_id) of the primitive test that was performed	String	Array
	module	This attribute contains the name of the driver that have been executed to call the test	String	
	name	This attribute contains the name of the test performed	String	
	result	This attribute contains the test result (”success”, ”inconclusive”, ”fail”)	String	
	message	This attribute contains the message sent about running the test	String	
	runtime	This attribute contains the test execution time (ms)	Number	
	resultValue	This attribute contains the return value of the test performed	String	

Table 5.4: Attributes of reports collection

The schedules collection stores all primitive test executions or test suite executions scheduled for a specific time by the user. The attributes of this collection are specified and explained in table 5.5:

Attribute	Description	Value Type
_id	This attribute is the unique id of each schedule in the system	String
active	This attribute has a true or false value, depending on whether the schedule is enabled to execute or not	Boolean
hour	This attribute contains the time when the schedule is to be executed	String
tests	This attribute contains the list of ids (_id) of all primitive tests that must be performed	Array
packages	This attribute contains the list of ids (_id) of all test suites that must be performed	Array

Table 5.5: Attributes of schedules collection

After specifying the data to be saved in each collection of the system's database, the next section will explain how the system interacts with the database, through queries, to obtain the data for its operation.

5.2 Backend

The Backend is the system tier responsible for managing the database and making the data available to Frontend. Therefore, framed in the Model-View-Controller (MVC) architecture, it is the Controller of the system and establishes the connection between the database and the user interfaces, thus guaranteeing the integrity of the data, not allowing other components to access or change them.

The technology used to develop this server was Node.js combined with Framework Express. This server is organized so that there is a division of the code according to its function, that is, instead of all the code being in one file, it was divided into different files and directories according to its purpose on the server. This will allow the reuse and modularity of the developed code, which will also facilitate its maintenance and understanding in the future.

Thus, the server structure is as follows:

- Models: Here are the models that correspond to the collections saved in the database. Each model contains the attributes corresponding to its collection and performs validations related to data types to ensure that wrong data types are not inserted into the database;
- Controllers: Here are the files responsible for performing all system operations, such as database queries, executing primitive tests and test suites, and creating new test suites using the DSL defined;
- Grammar: Corresponds to the DSL developed for the system, where is the grammar, composed by a Lexer and a Parser, and the Visitor that generates the code for the new test suites;

- Routes: Here is the file that routes the requests, from the client, that is, from the user interfaces to the controllers, according to the Uniform Resource Locator (URL) request. As soon as the requested operations are completed, sends the requested data to the client;

- app.js: The server is activated here, through the imported Express module. From the moment it is activated, it can start receiving requests from the Client. This file can be seen in Listing A.1.

Each of these elements mentioned above, has a fundamental role in the Server's logic, so each of them will be explained in the next subsections individually.

5.2.1 Models

The models represent, as mentioned, the collections stored in the database and here each model must then represent its collection and validate the data types of its attributes before transactions are made with the database. In these models, the module "mongoose" is imported, which is an object data modelling that will allow connection to the database in an asynchronous environment and will send and receive data in JSON format, which will facilitate the use of the data in the system.

Listing 5.1, shown below, serves as an example for the structure of the model files, choosing to demonstrate the model of the reports collection as it is the most complex and realizing this, all the others will be of the same or lesser level of complexity.

Listing 5.1: Report Model

```
const mongoose = require("mongoose");

const testsSchema = new mongoose.Schema(
   {
      id_test: { type: mongoose.Schema.Types.ObjectId, required: true },
      module: { type: String, required: true },
      name: { type: String, required: true },
      result: { type: String, required: true },
      message: { type: String, required: false },
      runtime: { type: Number, required: true },
      resultValue: { type: String, required: false },
   },
   { versionKey: false }
);

const reportSchema = new mongoose.Schema(
   {
      id_user: { type: String, required: true },
      date: { type: String, required: true },
      results: [testsSchema],
   },
```

```
22    { versionKey: false }
23 );
24
25 module.exports = mongoose.model("report", reportSchema);
```

We can see in line 1 of Listing 5.1 the import of the "mongoose"module. Next, between lines 3 and 14, inclusive, we see the model in the form of an object that represents the structure of a test result with its attributes and data types. This object serves as an auxiliary structure, in this case, to be introduced in the report model. Between lines 16 and 23, inclusive, we see the report model, also in the form of an object with its attributes and data types. In line 20, where the attribute "results"is specified, which is a list of objects, which in this case are objects with the structure specified above for the results of each test. Finally, in line 25 we see the export of the created model, making it available for use by other files, in this case, it will be used by the Controller who will be responsible for the reports collection operations.

The remaining models were also developed according to the existing collections and follow the same format, but applying their particularities according to the attributes it contains. All of them can be seen in Appendix A.1.

5.2.2 Grammar

The DSL developed in this dissertation aims to enable the creation of new test suites, from the primitive tests available in the system, with rules and logic applied. This will allow the test suites to be optimized to execute in the shortest possible time and may shorten certain executions whenever the suite specifies it. The language was created from the identification of terminal symbols, that is, the symbols that would be identified by Lexer. After this step, the Parser was created, where the rules of logic and sentence construction of the grammar are specified.

The terminal symbols have already been identified in table 4.1. The Lexer structure is shown below in Listing 5.2:

Listing 5.2: Grammar Lexer

```
1  lexer grammar TestLexer;
2
3  NEXT : '->' ;
4  AND : '&' ;
5  OR : '|' ;
6
7  IF : '?' ;
8  ELSE : ':' ;
9
10 RPAREN : ')' ;
11 LPAREN : '(' ;
12
13 END : ';' ;
```

```
14
15 KEYWORD : ([A-Za-z]+([/ _-][A-Za-z]+)*)
16          ;
17
18 WS
19      : [ \r\n\t] -> skip
20      ;
```

The structure of the Lexer is quite simple, starting with its identification and then just specifying all terminal symbols that must be recognized. The way these symbols are specified is through regular expressions, that is, for each symbol the regular expression that represents it is defined, however, always taking care that this definition does not include unexpected elements and, therefore, is not ambiguous.

The symbols we see in this grammar are very intuitive and this is also one of its advantages, as it will be easy for the end-user to understand, which is one of the objectives. The only symbol that gives rise to any further explanation is the `KEYWORD` symbol. This symbol must recognize all the names of the primitive tests introduced in the script and, therefore, its regular expression includes isolated words or also the composition of several words, thus giving the user some freedom to be more expressive in the choice of keywords since this it is also the purpose of the KDT methodology applied in the system.

After defining the terminal symbols and the Lexer specification, it is time to specify the sentence construction rules with these symbols and this is done in the Parser, which is shown below in Listing 5.3:

Listing 5.3: Grammar Parser

```
1  parser grammar TestParser;
2
3  options {
4     tokenVocab=TestLexer;
5  }
6
7  test
8     : statement END
9     ;
10
11 statement
12    : condition #Conditional
13    | seq #Sequence
14    ;
15
16 condition
17    : expr IF statement ELSE statement #IfElse
18    | expr IF statement #If
19    ;
20
21 seq
22    : KEYWORD (NEXT statement)*
```

```
23    ;
24
25 expr
26    : LPAREN KEYWORD (AND KEYWORD)* RPAREN #And
27    | LPAREN KEYWORD (OR KEYWORD)* RPAREN #Or
28    ;
```

The Parser also starts with its identification, following the reference for the Lexer that it provides the symbols to be able to know which are the terminal symbols. After these two steps, the sentences of the grammar are specified and here there is no more than a specification of the sequences that the elements of the language can follow. We can see, for example, in the element `statement` two possibilities. One possible `statement` is the `condition` that represents a conditional expression and the other possibility is a `seq` that represents a tests sequence. The most important part of the Parser to retain is the elements that come at the end of the lines for each possibility determined at the beginning of words by a #. This allows the Visitor to know the possible paths in the parsing tree that this Parser will generate.

So that this grammar can now be used by the system and generate the parsing tree that will be interpreted by the Visitor, it is still necessary to find a way to use it in the system. Since ANTLR offers the transformation of these grammars for several known programming languages, we will proceed to transform the grammar into JavaScript and include the code directly in the system. For this, it is necessary to execute the following command:

```
$   antlr4 -Dlanguage=JavaScript Lexer.g4 Parser.g4 -no-listener -visitor
```

In this command, we specify the Lexer and Parser to be transformed and we also specify that we do not want the generation of a Listener because, by default, it generates the Listener. Finally, we specify the generation of a Visitor because, by default, it does not generate the Visitor. After executing this command, several files will be generated, among which, the Visitor that will be the most important in the next steps, as this is where the code to be generated for the new test suites will be specified.

We can see below, in Listing 5.4, an example of a Visitor function:

Listing 5.4: Grammar Visitor

```
1 TestParserVisitor.prototype.visitAnd = function (ctx) {
2    this.auxOp = 0;
3    for (let i = 0; i < ctx.KEYWORD().length; i++) {
4       this.auxList.push(ctx.KEYWORD(i));
5    }
6    return "";
7 };
```

The Visitor's strategy developed is to go through the code script through the elements specified in the Parser and each element generate the corresponding code. The generated code, within the Visitor, is nothing more than a string that is incremented and filled up to the end of the parsing tree. All keywords

are also being saved in a list so that the list and the string containing the generated script are returned at the end. The list of keywords is necessary because after generating this code it will be necessary to match the keywords with the primitive tests but this is a process already done in the packages controller.

The entire Visitor code can be seen in more detail in Appendix A.9.

5.2.3 Controllers

The controllers, as mentioned earlier, are responsible for performing the system operations, that is, all queries, all executions of primitive tests and test suites and the creation of new test suites. So the way the controllers are structured is similar to the models, there is a file for each model that is responsible for carrying out the operations related to that collection or model. In each controller, several operations are available according to what is necessary for each one, but what is common to all are the Create, Read, Update, and Delete (CRUD) operations. In addition to these operations, there are even more that are particular to only a few models, such as the execution of primitive tests which is an operation developed only on the tests controller, the creation of new test suites that make use of the developed DSL and the execution of those same test suites which are operations developed only on the packages controller.

The following operations demonstrate some examples of the CRUD operations mentioned, with different controllers:

- Method to get all reports from the database, ordered by date in descending order:

Listing 5.5: Read all reports operation

```
1  module.exports.getReports = () => {
2     return Report.find().sort({ date: -1 }).exec();
3  };
```

- Method to get information related to one report, passing the id of the report as an argument:

Listing 5.6: Read one report operation

```
1  module.exports.getReport = (idReport) => {
2     return Report.findOne({ _id: idReport }).exec();
3  };
```

- Method to insert a schedule in the schedules collection, passing as an argument an object with the attributes and values of the new schedule:

Listing 5.7: Create one schedule operation

```
1  module.exports.insertSchedule = (schedule) => {
2     let s = new Schedule(schedule);
3     return s.save();
4  };
```

- Method to update a test in the tests collection, passing as argument the id of the test to be updated and an object with the attributes and values of the updated test:

Listing 5.8: Update one test operation

```
1  module.exports.updateTest = (idTest, newTest) => {
2     return Test.findOneAndUpdate({ _id: idTest }, newTest);
3  };
```

- Method to update all schedules in the schedules collection, in this case, to remove a particular test from all schedules, passing the id of the test to remove as an argument. This method is generally used when a test is removed from the system and it no longer makes sense to have a scheduled execution including the respective test:

Listing 5.9: Update many schedules operation

```
1  module.exports.removeTestFromAll = (idTest) => {
2     return Schedule.updateMany(
3        { },
4        { $pull: { tests: idTest } }
5     ).exec();
6  };
```

- Method to delete a schedule from the schedules collection, passing the id of the schedule to be removed as an argument:

Listing 5.10: Delete one schedule operation

```
1  module.exports.deleteSchedule = (idSchedule) => {
2     return Schedule.deleteOne({ _id: idSchedule });
3  };
```

The operations shown below are those mentioned different from the usual CRUD, however, given the context of the system they are fundamental to its performance:

- Method to execute a primitive test, passing as arguments the directory where the electronic test drivers are kept, the id of the test to be executed and the parameters of the test.

  In the first phase of this method, a query is made to obtain all the information related to the test and the counting of the execution time starts. Then, the driver responsible for executing the test is executed, which executes it and returns the results. As soon as the results arrive, the run time count stops and the test run time is saved in the results. Finally, some information about the test is added to the results to be saved in the reports and the object containing the results of the test execution is returned:

Listing 5.11: Execute one primitive test

```
module.exports.runTest = async (driversDirectory, idTest, defaultParam) => {
   let test = await Test.findOne({ _id: idTest }).exec();
   let startTime = process.hrtime()
   exec(`${driversDirectory}\\${test.module} "${idTest}" "${defaultParam}"`, (err,
       ↪ stdout, stderr) => {
      if (err) return stderr
      else {
         let endTime = process.hrtime(startTime)
         let result = JSON.parse(stdout)
         result.runtime = (endTime[1] / 1000000).toFixed(3);
         result.id_test = idTest;
         result.module = test.module;
         result.name = test.name;
         return result;
      }
   })
};
```

- Method to create a new test suite and insert it into the database, passing an object with the package's attributes as an argument.

  This method starts by using the grammar defined, using the Lexer and the Parser, to analyze the code script that was written by the user. At the end of this process, a parsing tree was generated and passed on to the Visitor of the grammar as an argument. If no error is found in the parsing tree, the Visitor will go through that tree and generate the code for the new test suite. After generating the code for the new script, it will be written to a file that will be saved in the directory where the system's test suites are. Then, the new test suite is also inserted into the database with all its attributes:

Listing 5.12: Create new test suite

```
module.exports.insertPackage = async (package) => {
   let chars = new antlr4.InputStream(package.script);
   let lexer = new Lexer(chars);
   let tokens = new antlr4.CommonTokenStream(lexer);
   let parser = new Parser(tokens);
   parser.buildParseTrees = true;
   let tree = parser.test();

   if (tree.parser._syntaxErrors === 0) {
      let listOfTests = await Test.getTests();
      let visitor = new Visitor(listOfTests);
      visitor.visitTest(tree);
      let textFile = visitor.getRes() + "";
```

```
14          let t = visitor.getTests();
15          let tests = t.filter(function (elem, pos) {
16             return t.indexOf(elem) == pos;
17          });
18          let fileName = package.name.toLowerCase().replace(/\s/g, '_') + ".js";
19          let filePath = scripts_path + fileName;
20
21          fs.writeFile(filePath, textFile, "utf8", function (err) {
22             if (err) throw err;
23             let t = new Package({
24                name: package.name,
25                description: package.description,
26                code: package.script,
27                path: fileName,
28                tests: tests,
29             });
30             return t.save();
31          });
32       } else {
33          return { errors: tree.parser._syntaxErrors };
34       }
35    };
```

- Method to execute a test suite, passing the id of the test suite to be executed as an argument.

  This method starts by querying the package collection for information about the package to be executed. After receiving this information, it only needs to import the file containing the test suite code and call the "execute" method that triggers the execution of the test suite. In the end, wait for the results and return them.

Listing 5.13: Execute a test suite

```
1  module.exports.runPackage = async (idPackage) => {
2     let package = await Package.findOne({ _id: idPackage }).exec();
3     if(package){
4        const file = require("./../public/Packages/" + package.path);
5        return await file.execute();
6     }
7     return {};
8  };
```

The operations demonstrated and explained are examples of the different types of operations that the system performs and supports, however, we can see that all of them have in common the fact that each of them performs only a certain action that allows to isolate these actions and reuse them frequently in different parts of the code for different purposes. This will also allow better maintenance of these

operations, since whenever it is necessary to make any changes in any of them, it will be done only once and in the indicated location, instead of having to change in different locations, which would easily cause inconsistencies in the code in the long-term.

All controllers can be seen in Appendix A.3.

5.2.4 Routes

The server's routes, as mentioned previously, are responsible for defining the requests that the client can request and in this case, they are the ones that receive these requests, forward them to carry out the operations that are necessary to satisfy them, and in end, send the data to the client. The way the routes are built is based on the URL, that is, for each request, a URL is associated and as the defined API follows the REST architecture these routes will follow very specific and clear formats to be more noticeable the type of operation that needs to be executed.

As we saw earlier in the controllers, CRUD operations are the most common and here in routes there is also a way to signal requests to determine the type of operations they are dealing with. In this case, they are HTTP requests and in this system, four types of requests were implemented:

- GET - The GET method is used to retrieve information from the server using a given Uniform Resource Identifier (URI). Requests using GET should only retrieve data and should have no other effect on the data. We can see the GET requests that the api provides in table 5.6;
- POST - A POST request is used to send data to the server. We can see the POST requests that the api provides in table 5.7;
- PUT - Replaces all current representations of the target resource with the loaded content. We can see the PUT requests that the api provides in table 5.8;
- DELETE - Removes all current representations of the target resource provided by the URI. We can see the DELETE requests that the api provides in table 5.9.

Method	Route	Sub-route	Description
GET	/		Checks for updates in primitive tests
GET	/backups		Get backups available in the backups directory
GET	/configurations		Get system configurations
GET	/tests		Get the primitive tests
GET	/tests	/:idTest	Get metadata from a primitive test
GET	/packages		Get the test suites
GET	/packages	/:idPackage	Get metadata from a test suite
GET	/reports		Get the execution reports
GET	/reports	/:idReport	Get an execution report
GET	/schedules		Get the system execution schedules
GET	/schedules	/:idSchedule	Get an execution schedule

Table 5.6: API - GET requests implemented

Method	Route	Sub-route	Description
POST	/login		Log in the user and assign a token
POST	/backups		Back up the system
POST	/restore	/:backup	Restore the system with a backup
POST	/configurations		Create configurations for the system
POST	/tests	/run	Execute a primitive test
POST	/packages		Create a new tests suite
POST	/packages	/run	Execute a tests suite
POST	/reports		Create an execution report
POST	/schedules		Create an execution schedule

Table 5.7: API - POST requests implemented

Method	Route	Sub-route	Description
PUT	/configurations	/:idConfiguration	Update system configurations
PUT	/packages	/:idPackage	Update a tests suite
PUT	/schedules	/:idSchedule	Update an execution shedule

Table 5.8: API - PUT requests implemented

Method	Route	Sub-route	Description
DELETE	/packages	/:idPackage	Delete a tests suite
DELETE	/schedules	/:idSchedule	Delete an execution shedule

Table 5.9: API - DELETE requests implemented

In the previous tables, we see all the services that the developed API makes available to the client. The implementation of all of them will not be detailed here, but only of some, as a demonstrative example of the implementation format, which, except for some more complex requests, is always the same.

In Listing 5.14, shown below, we can see the implementation of the route for the GET /tests request:

Listing 5.14: Example of GET request implementation

```
1 router.get("/tests", function (req, res) {
2    Tests.getTests()
3       .then((data) => res.jsonp(data))
4       .catch((error) => res.status(500).jsonp(error));
5 });
```

In this example, we see in line 2 the use of "Tests", which is the reference already imported into the tests controller. Then, with this reference, the "getTests" method is called, which is exported in the tests controller. So, what is happening is exactly what was described previously, this router is forwarding the operation to the controller responsible for it, and then it just waits for the results to arrive to return them to the client.

In Listing 5.15, shown below, we can see the implementation of the route for the POST /packages request:

Listing 5.15: Example of POST request implementation

```
router.post("/packages", function (req, res) {
   Packages.insertPackage(req.body)
      .then((data) => res.jsonp(data))
      .catch((error) => res.status(500).jsonp(error));
});
```

In this example, the process is very similar to the previous one. The only difference is that in POST requests the information to be saved in the system comes in the body of the request ("req.body") and therefore it is necessary to send this information to the method that deals with the operation.

In Listing 5.16, shown below, we can see the implementation of the route for the PUT /schedules/:idSchedule request:

Listing 5.16: Example of PUT request implementation

```
router.put("/schedules/:idSchedule", function (req, res) {
   Schedules.updateSchedule(req.params.idSchedule, req.body)
      .then((data) => res.jsonp(data))
      .catch((error) => res.status(500).jsonp(error));
});
```

In this example, we see again the same similarities, but with a slight difference. PUT requests usually bring the identifier of the element that we want to update in the sub-route and, therefore, to have access to it we must access the request parameters ("req.params"). In the previous example, we saw the passage of the body information of the request to be passed to the controller and now we also see but with the addition that the identifier of the element to be updated is also passed.

In Listing 5.17, shown below, we can see the implementation of the route for the DELETE /packages/:idPackage request:

Listing 5.17: Example of DELETE request implementation

```
router.delete("/packages/:idPackage", function (req, res, next) {
   Packages.deletePackage(req.params.idPackage)
      .then((data) => res.jsonp(data))
      .catch((error) => res.status(500).jsonp(error));
});
```

In this example, we see the same structure again, but to delete an element from the system, we only need to access the request parameters to get the identifier and pass it to the controller.

All other requests implemented by the API can be seen in Appendix A.4.

5.3 Frontend

The frontend is the system tier responsible for creating and managing graphical interfaces for end-users. In this case, there are two types of users in the system, already specified in section 3.1, and it is important to understand well the limits on what each one should be allowed to do or not do. The first type of user, more basic, will only have access to the execution of primitive tests and test suites. The second type of user, already responsible for managing the system and also the test suites for it, has access to all other features. The technology used to develop this tier was React, as it will allow us to create dynamic interfaces, with components managing their state and the possibility to compose the components themselves. This allows the code to be modularized and, in the future, it will be easier to understand the code.

5.3.1 Components

As mentioned, the development of components in React becomes an asset, but to master the use of technology it is necessary to understand the fundamentals and the way the components interact with each other. The three concepts that I highlight are the following:

- State: The state of a component is mutable and can be changed by the component itself, due to the actions performed by the user. Information stored in a component's state can be accessed as attributes of the component, such as "this.state.name";
- Props: Props are state information from a parent component to a child component, so the child cannot directly change the props but can access them in the same way as the parent, such as "this.props.name". They are generally used to determine some properties of the child component when it is created;
- Events: Events are how the child component should inform the parent component of changes that have occurred. This is how a child component can change the state of the parent component, through events that will inform the parent component so that it updates its state.

Thus, to understand how these concepts apply in practice and make the most of the use of React components, we can see below, in figure 5.1, an illustration of how these concepts are related:

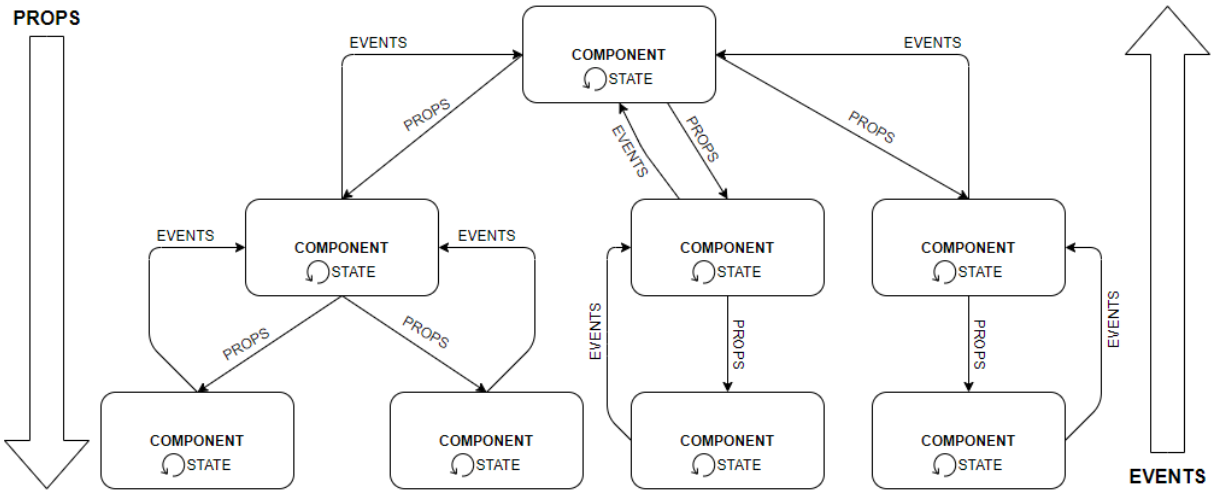


Figure 5.1: Interactions between reaction components

5.3.2 Obtaining API data

Another important aspect for this part of the system to work as planned is to obtain the data that is managed by the Backend tier. For the graphical interfaces built to be as optimized as possible and quick in obtaining data, so that the user does not have to wait long to load the pages, the data must be obtained in the best way. And here the decision made was that the parent components of each page make the data requests to the API at the time of its creation. With this, what happens on the system pages is that whenever the user changes the page or enters a new page, the data is requested and loaded. This will allow the actions taken by the user on the components belonging to these pages to be carried out much more quickly, giving the user the perception that nothing has happened when real events and state changes have already occurred witch allows the page to become dynamic with speed desired.

The way to obtain the data is through HTTP requests, explained previously, therefore, to make the code clearer, a file was created only for the methods of requesting data from the API. This file contains the base URL of the Data API and all methods add only the route and sub-route as needed. We can see below, in Listing 5.18, an example of a method of obtaining data by making an HTTP request to the data API:

Listing 5.18: Example of request to obtain API data

```
export const getTests = async () => {
   try {
      const response = await axios.get(`${url}/tests`);
      return response.data;
   } catch (error) {
      const statusCode = error.response ? error.response.status : 500;
      throw new Error(statusCode.toString());
   }
};
```

In this example, we can see how HTTP requests are made to the API. These requests are made through the imported module ”Axios”since the technology does not provide this functionality natively. Another important feature that we see in this example is the use of the keyword ”await”, which in this particular case makes the method wait for the results of the API. This is also one of the strong characteristics of the technologies used, as they allow to establish of asynchronous communications.

All other methods that make API requests to the Frontend are visible in Appendix B.1.

5.3.3 User Interfaces

Taking into account the users of the system, the division of access to the pages by each user was carried out immediately through the login on the first page, which will allow assigning a JSON Web Token (JWT) to the user and will only give him access to the appropriate functionalities. We can see the login page in figure 5.2:

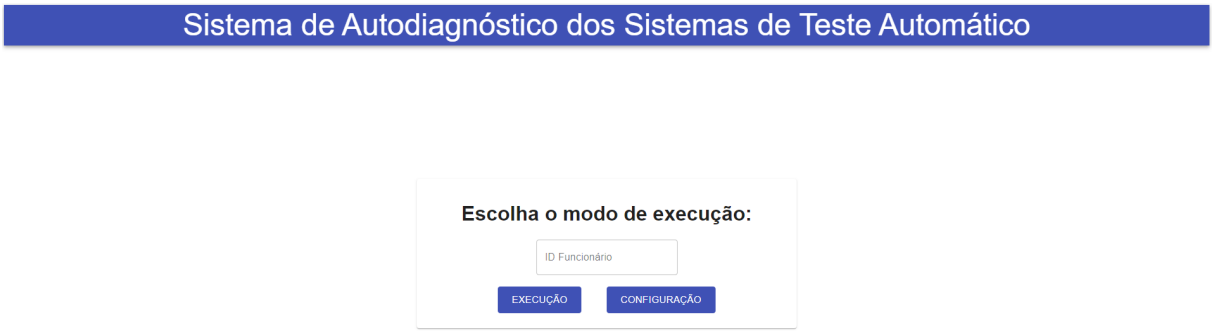


Figure 5.2: Login Page

On this page, the user must enter his ID (provided by CONTROLAR) and enter the appropriate mode. If it is an operator it must enter the execution mode, if it is the test manager it must enter the configuration mode. Still, the system according to the ID only allows users to enter the appropriate mode.

On the page available to the operator, it is only possible to execute primitive tests or test packages. The operator has, on the left side of the interface, a list with all the primitive tests of the system. To execute these tests, it must select the ones he wants, and they can execute all at once, and send them to the list on the right which is properly identified as the list of tests to be executed. After selecting the tests and passing them to the execution pipe, it just needs to press the button to execute. The system will execute the tests and, in the end, a table will be presented to the user with the results obtained. The execution interface and the results table presented to the user can be viewed below, in figures 5.3 and 5.4, respectively:

Figure 5.3: Execution Page

The way described above for the process of selecting and executing primitive tests by the user is the same for the test suites but on the right side of the page. The results in this case are also shown in the same way.

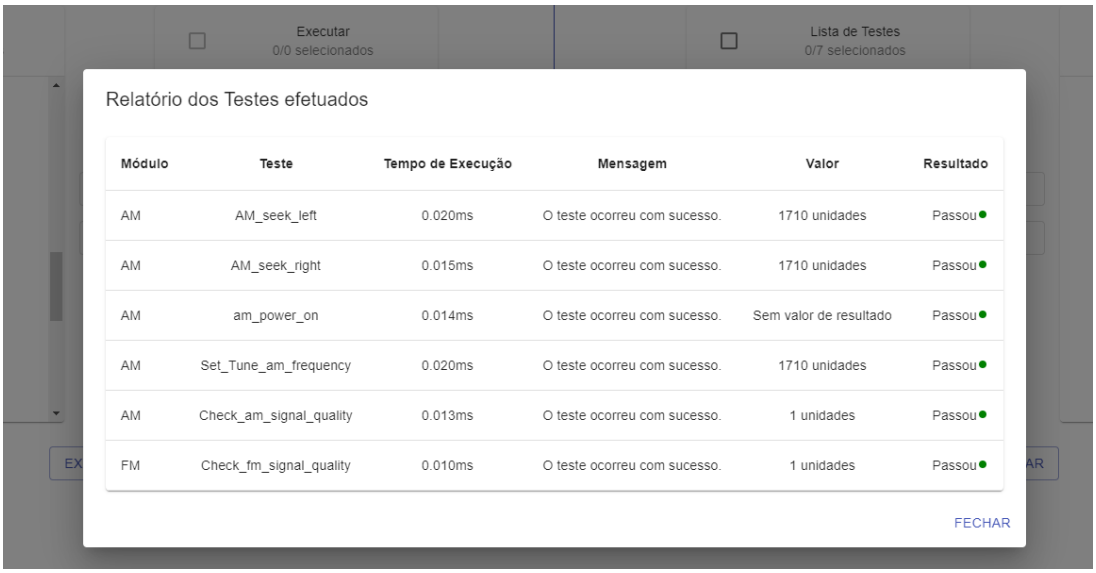


Figure 5.4: Execution Results Table

As for the results table presented to the user, it shows all the data provided to us from the execution drivers and also some metadata that is already added by the system for better understanding. In this case, the table data provided by the execution drivers are the message, the result value and the test result. The message is an informative attribute for the user about the occurrence of the test. The result value is the value returned by the test execution because normally the drivers perform practical functions with results in the order of the units corresponding to the test application. The important metric for the effect of the test results on the system is the result and it simply tells us whether the test passed, failed or was inconclusive. To give greater visual intuition about the test results, coloured balls were also added to the

test result, in which green means that the test passed, yellow was inconclusive and red failed. In this way, the visualization of the results and the interpretation will be better. For the first type of user, the industrial operator, these are the actions and resources to which he has access. Note that the purpose of these interfaces is to be as simple and functional as possible so that there are no ambiguities in the user's decision making.

For the second type of user, the test manager or administrator, there will be more pages and resources accessible. Starting with the execution reports page shown in figure 5.5, this page presents a table with all the execution reports made in the system. What we see in each row of this table is the information summarized in the report, such as the id of the employee who performed the executions of that report, the date of execution, the time of execution of all tests performed and the results, that is, a relationship between the number of approved and failed or inconclusive tests. Again, here we also see the use of colours in the results to facilitate the reading and analysis of the results of the executions.

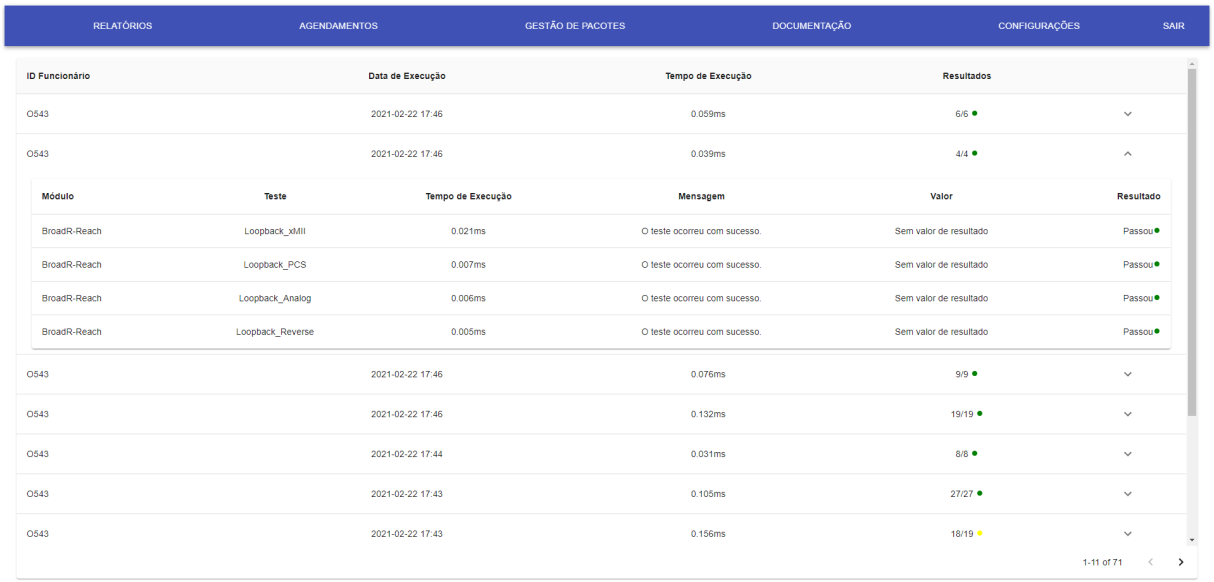


Figure 5.5: Reports page

Each row in the table also allows for expansion that opens an internal table, detailing the results of all tests performed in this report. This internal table follows the same format as the table shown to the user when executing the tests on the test execution page.

The second page that the manager has access to is the execution schedules page, shown in figure 5.6. On this page, we see a list of schedules, in which each element of the list contains the time when it will be executed, a button to determine if is active or not and an option to delete. An informational message is detailed, stating how many hours and minutes are left to run. To edit the schedule and change its properties, just click on the desired element and a form similar to the one for creating the schedule is opened, with the difference that it is already filled in with its information.

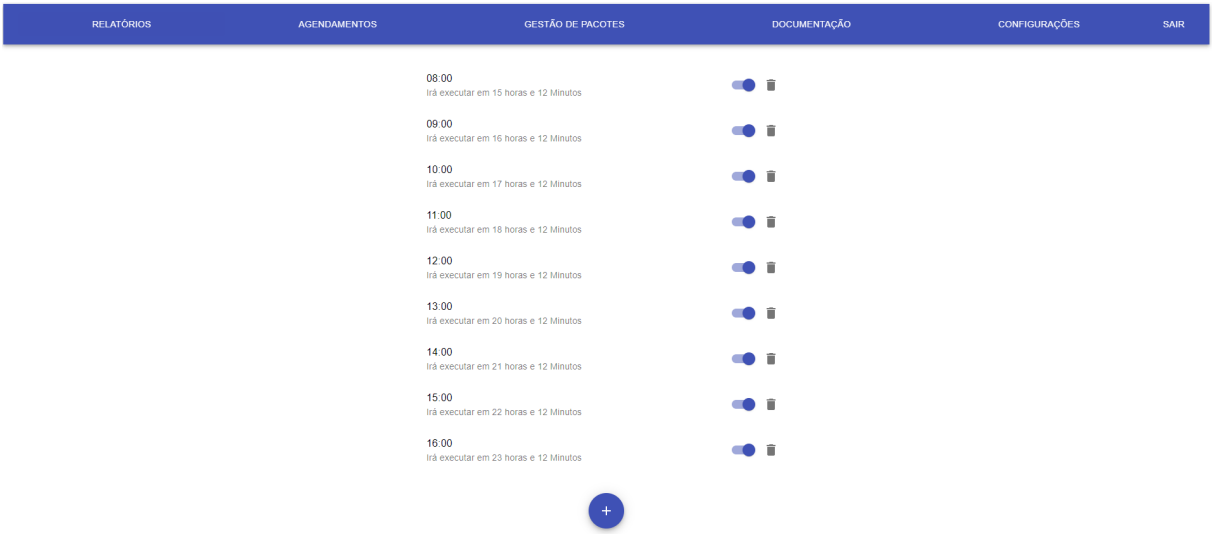


Figure 5.6: Schedules page

In the form, the information to be filled out is the time to execute, the activation of the scheduling and the selection of the primitive tests and test suites to be executed. The form can be seen in figure 5.7.

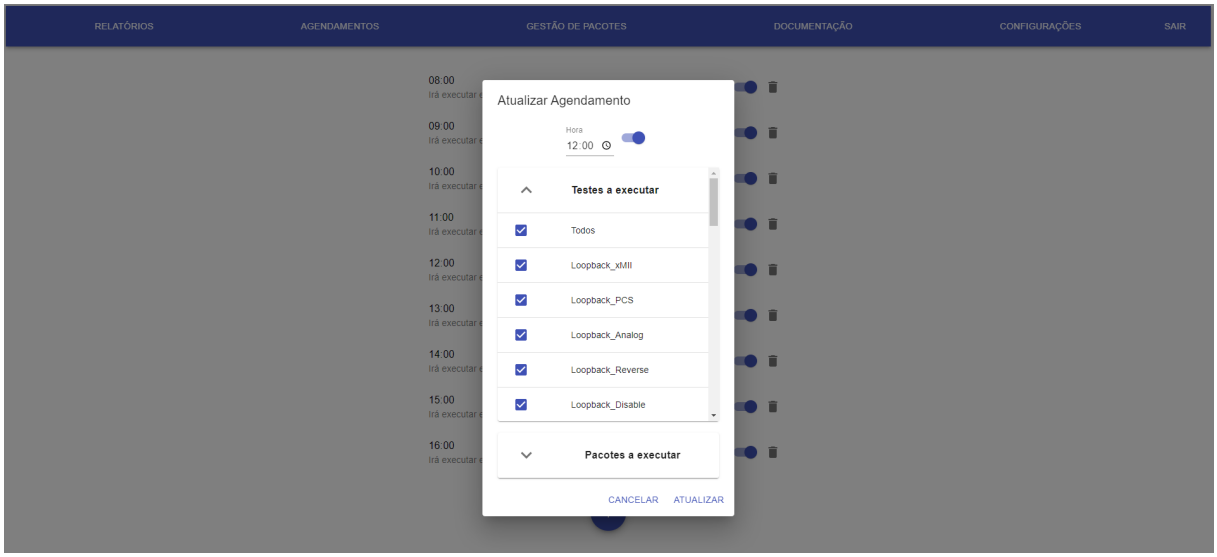


Figure 5.7: Form to add and update schedule

The next page that the user has available is the page for managing and configuring new test suites for the system, which can be seen in figure 5.8. The user has on this page at his disposal the list of existing packages in the system, where he can remove or edit them. There is also a form for creating a new test suite, where the user only needs to specify the name, description and code of the new test suite, the code is written with the DSL developed in this work. In this case, the elements that can be used to write the code are the connectors below the form that are made available to the user according to the status of their script, to help the user and try to avoid errors. The other elements to include in the script are the primitive tests, and these are made available in a list next to the form where the user can even see the description to understand what the test does. To include a test in the script just needs to click on it and it

is automatically added to the script. This way, the user does not need to write anything manually, having to select the elements he wants to add to the script.

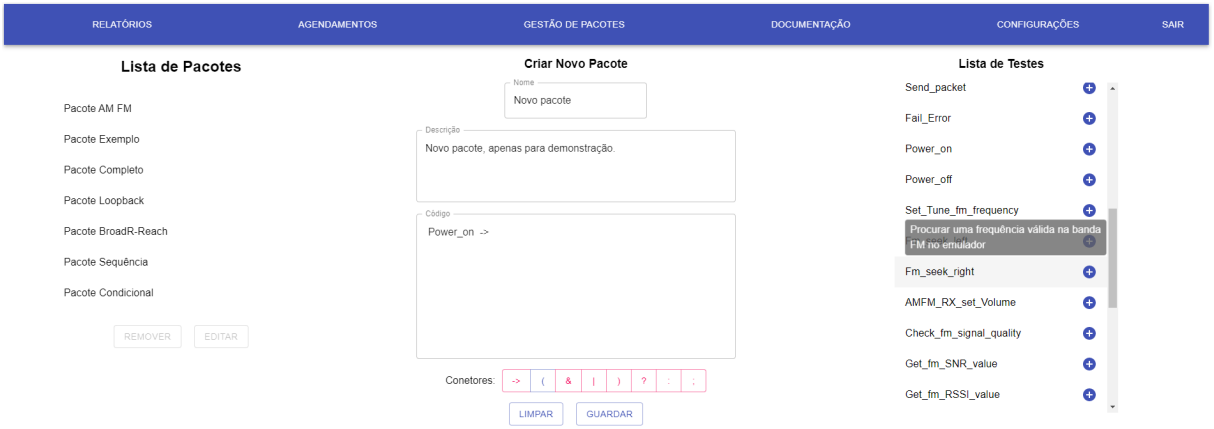


Figure 5.8: Package creation and management page

The fourth page that the manager has access to is the primitive tests documentation page, and this is a purely informative but important page for those who will manage the tests and test suites for the system. This is because whoever develops the primitive tests may not be the same person who later manages the system and even who creates the test suites, so there needs to be a page where users can be informed about the primitive tests of the system. This page can be seen in figure 5.9.

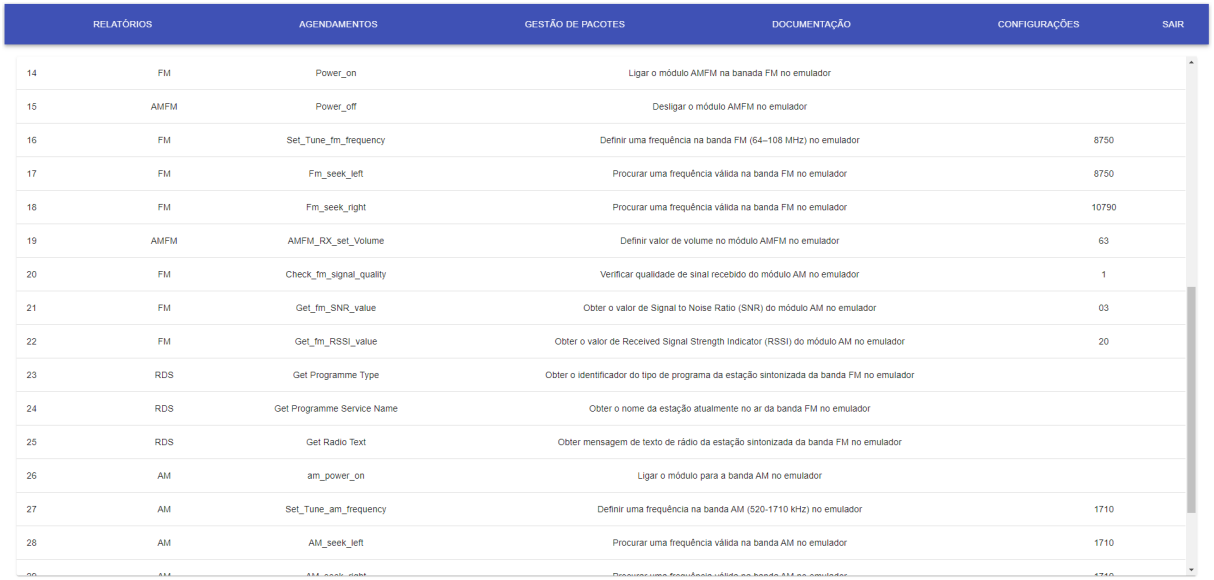


Figure 5.9: Documental page about available primitive tests

The manager's last page is the page he has access to and must update the system configurations whenever necessary. These configurations are necessary for the system to function, as they are used in fundamental processes all the time. As configurations there is the manager ID, which is the only one to have access to this system mode, it has the MongoDB bin directory, that is, the directory where MongoDB

executable files are located and which allow data extractions and imports, there is the directory where the system itself is installed and finally the directory to which backups are to be exported and also where they are to be imported from. This page can be seen in figure 5.10:

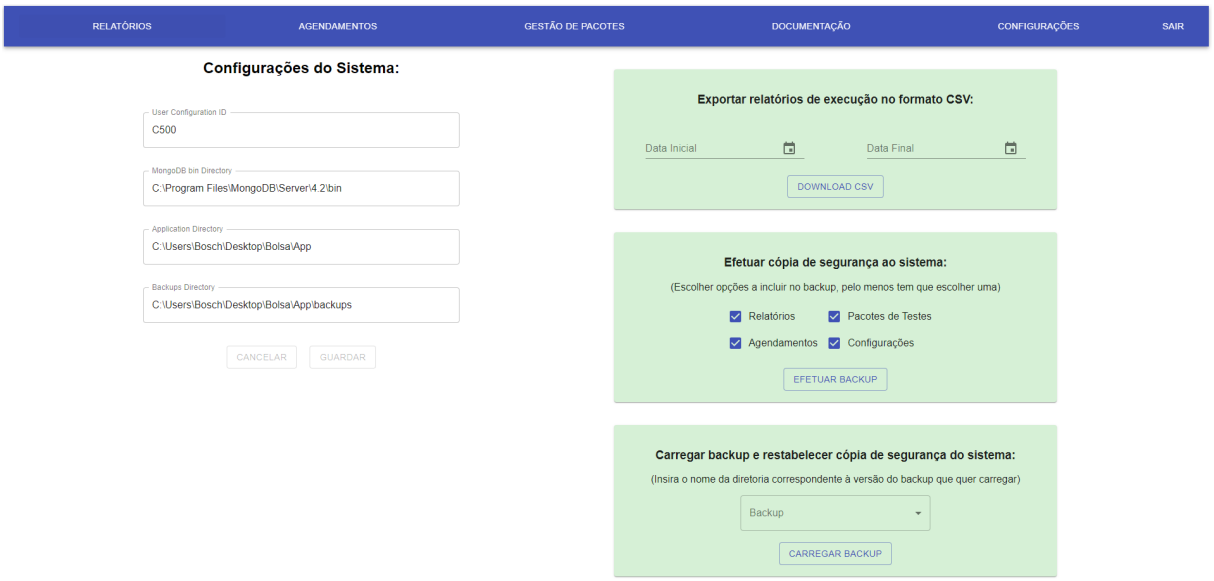


Figure 5.10: System configurations page and data export and import

The features that this page has available to the user, in addition to the management of the system configurations, are the possibility to export the system execution reports in CSV format, which can be filtered between two dates or not, the possibility to make backup copies, be able to customize the data you want to copy and restore backup copies on the system. The backup copies will allow the system to be always aware of situations of failure or data corruption. However, all of these features require that the system configurations are properly completed and correct.

Having thus presented all the pages that the system makes available, all the code developed for them can be found in Appendix B.4.

5.4 Validation

Having already implemented the system with all the requirements that were established, several test cases were created to be carried out in the system to validate the solution and confirm the fulfilment of all the proposed objectives. The first tests were carried out on the most fundamental functionalities of the system, the execution of the tests and the automation of the update in the face of changes introduced in its supply. Several test scenarios were simulated and the system behaved as expected, passing all tests performed.

We can see the tests performed and their results in table 5.10.

Test Case	Test Steps	Test Data	Expected Result	Actual Results	Pass/Fail
Check the execution of primitive tests	1. Select tests to execute 2. Click the button to run 3. When window to confirm appear, select yes	Tests Selected: am_power_on set_frequency am_seek_left	A table with the test results must appear	Table presented with the results	Pass
Check the execution of all primitive tests	1. Select all tests to execute 2. Click the button to run 3. When window to confirm appear, select yes	Tests Selected: All available	A table with the all test results must appear	Table presented with all results	Pass
Check for a message when no test is selected to notify	1. Click button to execute without selecting tests	None	A warning message should appear notifying you that the user has not selected any tests	Message appeared with the notification	Pass
Check if system updates the removed primitve tests	1. Go to the electronic test drivers directory 2. Remove driverA.json from directory 3. Open the system in the execution mode 4. Check if tests of driver A are available	None	Tests from driver A must not appear to be selected	Tests are not available	Pass
Check if system updates the added primitve tests	1. Go to the electronic test drivers directory 2. Add driverA.json from directory 3. Open the system in the execution mode 4. Check if tests of driver A are available	None	Tests from driver A must appear to be selected	Tests are available	Pass
Check the execution of a test suite	1. Select test suite to execute 2. Click the button to run	Test suites Selected: Pacote AM FM	A table with the test suite results must appear	Table presented with the results	Pass
Check the execution of all test suites	1. Select all test suites to execute 2. Click the button to run 3. When window to confirm appear, select yes	Test suitesSelected: All available	A table with the all test suite results must appear	Table presented with all results	Pass
Check for a message when no test suite is selected to notify	1. Click button to execute without selecting tests	None	A warning message should appear notifying you that the user has not selected any test suites	Message appeared with the notification	Pass

Table 5.10: Results of test cases performed on test executions and management

This table contains in each row:

- Test Case - The test case is the description of the test and the functionality that will be tested, so it

must be as descriptive and explicit as possible so that anyone who does not know the functionality of the system can understand what the test does;

- Test Steps - The test steps are the steps that the tester must follow strictly to reproduce exactly the same result or attempt;
- Test Data - The test data is the data that will be needed to perform the test, it may be necessary to insert in forms in the interface for example;
- Expected Result - The expected result is the result that the test must achieve to meet the system requirements;
- Actual Result - The actual result is the result that the test obtained after the execution;
- Pass/Fail - The test, in the end, must pass or fail. It must pass if the actual result obtained from its realization is equal to the expected result and fails otherwise.

After carrying out the test cases discussed above, the test cases were performed for all other features of the system, with the results tables all having the same format. We can see the results and test cases remaining in the following tables: 5.11, 5.12 and 5.13.

Test Case	Test Steps	Test Data	Expected Result	Actual Results	Pass/Fail
Check the execution reports	1. Open tab "Relatórios"	None	A table with the all reports must appear	Table Presented	Pass
Check the details of an execution report	1. Open tab "Relatórios" 2. Choose a report and click on it	None	A table with the evidence of the results of the tests carried out in that report must be presented	Table Presented	Pass
Check documentation of primitive tests	1. Open tab "Documentação"	None	A table with all primitve tests metada must be presented	Table Presented	Pass
Check the schedule of a new execution	1. Click in the button to add schedule 2. Enter time 3. Activate the option button 4. Select the primitive tests to execute 5. Select the test suites to execute 6. Click in the save button	Time: 8:00 Active: True Tests Selected: power_on set_fm_frequency fm_seek_right	A new schedule must be added to the system	Schedule created	Pass
Check trying to schedule a new execution without selecting any test	1. Click in the button to add schedule 2. Enter time 3. Activate the option button 4. Click in the save button	Time: 10:00 Active: True	An error message must appear stating that at least one test must be selected	Message appeared	Pass
Check the update of a schedule	1. Click in the schedule timed to 8:00 2. Change time to 9:00 3. Click in the save button	Time: 9:00	The schedule must be updated	Schedule updated	Pass
Check the removal of a schedule	1. Click in the optin button to remove in the schedule timed to 9:00	None	The schedule must be removed	Schedule removed	Pass

Table 5.11: Results of test cases performed on visualization reports, documentation of primitive tests and scheduling of executions

Test Case	Test Steps	Test Data	Expected Result	Actual Results	Pass/Fail
Check the creation of a new test suite with wrong script	1. Open tab "Gestão de Pacotes" 2. Enter package name 3. Enter package description 4. Write the script 5. Click in the save button	Package Name: Novo Pacote Package Description: Pacote para demonstrar erro Package Script: InventedTest -> am_power_on ->	An error alert must appear to the user saying the code is not correct	Alert showned	Pass
Check the creation of a new test suite	1. Open tab "Gestão de Pacotes" 2. Enter package name 3. Enter package description 4. Write the script 5. Click in the save button	Package name: New Package Package description: Package for demonstrattion Package script: power_on-> am_power_on ;	Test suite must be created and should appear in the list on the left side	Test suite created and available	Pass
Check the update of a test suite	1. Open tab "Gestão de Pacotes" 2. Click in the package named "New Package" 3. Click in the edit button 4. Change the name 5. Click in the save button	Package name: New Package to Remove	Test suite must be updated	Test suite was updated	Pass
Check the removal of a test suite	1. Open tab "Gestão de Pacotes" 2. Click in the package named "New Package to remove" 3. Click in the remove button	None	Test suite must be removed	Test suite was removed	Pass
Check the export of reports to a CSV with dates no covered	1. Open tab "Configurações" 2. Enter begin date on export CSV 3. Enter end date on export CSV 4. Click download button	Begin Date: 25/07/2021 End Date: 01/09/2021	A CSV file must be downloaded without lines	CSV file was downloaded with no lines	Pass
Check the export of reports to a CSV	1. Open tab "Configurações" 2. Enter begin date on export CSV 3. Enter end date on export CSV 4. Click download button	Begin Date: 01/01/2021 End Date: 01/08/2021	A CSV file must be downloaded that contains all the reports from the specified interval	CSV file downloaded with the all reports from the interval	Pass

Table 5.12: Results of test cases performed in managing and creating test suites and exporting reports to CSV

Test Case	Test Steps	Test Data	Expected Result	Actual Results	Pass/Fail
Check system backup, including schedules and packages	1. Open tab "Configurações" 2. Include Packages and Schedules 3. Click int the make backup button	None	A zip file must be saved in the backup directory specified in the system configurations and a success message should appear	Zip file successfully saved to the backup directory and the success message appeared	Pass
Check system backup, including all system data	1. Open tab "Configurações" 2. Include all options 3. Click int the make backup button	None	A zip file must be saved in the backup directory specified in the system configurations and a success message should appear	Zip file successfully saved to the backup directory and the success message appeared	Pass
Check the system backup, with incomplete system configurations fields	1. Open tab "Configurações" 2. Include all options 3. Click int the make backup button	None	An error message must appear informing that all fields of system configurations must be completed	Error message appeared stating that all fields must be completed	Pass
Check the restore of a backup in the system, including schedules and packages	1. Open tab "Agendamentos" 2. Delete the 8:00 schedule 3. Open tab "Gestão de Pacoes" 4. Delete package with name "Pacote Exemplo" 5. Open tab "Configurações" 6. Select backup to restore with name ending with substring "_sp" 7. Click the button to make restore	None	The removed schedules and packages must be on the system again and a success message should appear	Schedules and packages have been re-established and the message of success has appeared	Pass
Check the restore of a backup in the system, including all data	1. Open tab "Agendamentos" 2. Delete all schedules 3. Open tab "Gestão de Pacoes" 4. Delete all packages 5. Open tab "Configurações" 6. Select backup to restore with name ending with substring"_crsp" 7. Click the button to make restore	None	The removed elements must be on the system again and a success message should appear	All data was re-established and the message of success has appeared	Pass
Check the restore of a backup in the system with incomplete fields on system configurations	1. Open tab "Configurações" 2. Select backup to restore 3. Click the button to make restore	None	An error message must appear informing that all fields of system configurations must be completed	Error message appeared stating that all fields must be completed	Pass
Check the update of system configurations	1. Open tab "Configurações" 2. Delete the backup directory 3. Click in the save button	None	Backup directory must be empty	Backup directory is empty	Pass

Table 5.13: Results of test cases performed on system backups, restoring backup versions and managing system configurations

In total, 28 test cases were carried out covering all the functionality of the system and in some of them with more than one test case. No more test cases were carried out because the time it would take to do so is immense, but the test cases performed were considered to be the most comprehensive cases and therefore will give the greatest coverage of requirements. After analyzing all the results obtained in the tests and verifying that they all passed, we can say that all requirements have been successfully implemented and the system is ready to be integrated with the other components.

5.5 Discussion

As mentioned, the entire implementation of the system was presented in this chapter. The different tiers of the system were presented, as well as the technologies, methods and strategies used in each one to develop a system that would respond in the best way to all requirements. The entire code was not explained or detailed because it is very extensive, but the most relevant parts were explained and allow, for those who read it, the reproduction of this work and application in its context. It should also be noted that the developed system is prepared to be integrated with a CPS and with the integration of electronic tests guaranteed and, therefore, ready to carry out the self-diagnosis of the CONTROLAR machines. To validate the implementation of the system and its compliance with the established requirements, 28 test cases were carried out to cover all requirements. The results show that all test cases have been approved and, therefore, the system meets all the proposed requirements.

In the next chapter, the conclusions of this dissertation will be presented, as well as an analysis of their contributions and some suggestions for future work.

Chapter 6

Conclusions and Future Work

The main contributions of this dissertation are the design of the architecture to integrate a self-diagnosis tests system into a CPS and its implementation. An integrated system that will allow CONTROLAR to carry out the self-diagnosis of its machines in real-time, thus guaranteeing their integrity.

After reviewing the current state of the art, we found that the existing solutions for testing systems are still very much focused only on software testing and very little on the integration of other types of tests and consequently on integration in CPS. In light of the knowledge that has been acquired through the analysis of software test systems and test automation, it is now possible to create a system with some of these characteristics, but designed to be integrated and self-diagnose the CPS. With that in mind, an architecture for the self-diagnosis tests system was designed that combines the KDT methodology with a DSL to manage and configure the tests of the system. This architecture provides a modular and extensible solution so that the system can be integrated with the CPS and perform any type of test. Also, another architecture was designed to extend and integrate the self-diagnosis tests system into a CPS that proves the modularity of the proposed architecture for self-diagnosis tests system, demonstrating how we can extend it into a CPS.

The last phase of the work in this dissertation was the implementation of the system, according to the specified requirements. The system was implemented based on the proposed architecture, proving that it gives the system the ability to be modular and allow self-diagnosis by performing any type of test. To validate the implementation of the system and its compliance with the established requirements, 28 test cases were carried out to cover all requirements. The results show that all test cases have passed and, therefore, the system meets all the proposed requirements.

The proposed modular and extensible architecture represents an innovation for research in self-diagnosis systems and CPS, as it allows the combination of these two types of systems, through the use of KDT methodology with a DSL to manage and configure the tests of the system. This architecture also allows the execution of the tests to be done remotely or by any other system with permission to make HTTP requests

to the API REST provided. Although the focus of the architecture is the application in a CPS, it is also applicable to any type of system, since it is generic to accept any type of test. With this work, we proved that it is possible to integrate self-diagnosis tests systems into a CPS with a practical and also generic solution that can be integrated with other types of testing systems. These contributions offer the guarantee of safety, performance and functionality when using CONTROLAR's machines, as they can now be diagnosed in real-time, allowing clients like Bosch to make the most of their use in the production environment.

6.1 Future Work

As future work, it would be interesting to improve the interface for creating new test suites in the system. Although the solution currently implemented is practical and allows good use, it could be even more practical and simple for the user if a drag and drop window were developed in the design of new test suites instead of writing a code script. Another aspect that could be expanded would be the possibility of introducing weekly or monthly schedules according to the annual production calendars, which would be even more useful for this resource.

Although the system is quite complete and correctly implements all the features required by CONTROLAR, there is still a potential for expansion in other types of uses that have not been explored by the company, such as, for example, the introduction of machine learning. It will be possible in the future, from the data generated in a production environment, to make predictions of the daily or even weekly moments when the machine will be more vulnerable to errors. This is a topic that can be extremely interesting, as it can give users of the machines a better perception of how they should and not use them to obtain the best performance from them.

Bibliography

[1] L. Abreu. *Nodejs - Construção De Aplicações Web*. FCA, 2016.

[2] C. Academy. *What is REST?* 2021. url: https://www.codecademy.com/articles/what-is-rest. (accessed: 28.01.2021).

[3] S. Aggarwal. *Modern Web-Development using ReactJS*. Tech. rep. 2018.

[4] C. ALGORITMI. *Ongoing Projects*. url: http://algoritmi.uminho.pt/projects/ongoing-projects/. (accessed: 14.12.2020).

[5] S. A. Asadollah, R. Inam, and H. Hansson. "A survey on testing for cyber physical system." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2015. isbn: 9783319259444. doi: 10.1007/978-3-319-25945-1_12.

[6] M. F. A. Carvalho. *Automatização de Testes de Software*. 2010. url: https://files.isec.pt/DOCUMENTOS/SERVICOS/BIBLIO/teses/Tese_Mest_Marcio-Carvalho.pdf. (accessed: 20.01.2021).

[7] CCG. *TSIM – Test System Intelligent Machines*. url: https://www.ccg.pt/my-product/tsim-test-system-intelligent-machines/. (accessed: 14.12.2020).

[8] M. H. Cintuglu, O. A. Mohammed, K. Akkaya, and A. S. Uluagac. *A Survey on Smart Grid Cyber-Physical System Testbeds*. 2017. doi: 10.1109/COMST.2016.2627399.

[9] S. Ciraci, J. C. Fuller, J. Daily, A. Makhmalbaf, and D. Callahan. "A Runtime Verification Framework for Control System Simulation." In: *2014 IEEE 38th Annual Computer Software and Applications Conference*. 2014, pp. 75–84. doi: 10.1109/COMPSAC.2014.14.

[10] Controlar. *Máquina Inteligente de Sistema de Testes Funcionais*. url: https://controlar.com/areas-de-negocio/sistemas-de-teste/tsim/. (accessed: 14.12.2020).

[11] B. Costa, P. F. Pires, F. C. Delicato, and P. Merson. "Evaluating a Representational State Transfer (REST) architecture: What is the impact of REST in my architecture?" In: *Proceedings - Working IEEE/IFIP Conference on Software Architecture 2014, WICSA 2014*. 2014. isbn: 9781479934126. doi: 10.1109/WICSA.2014.29.

[12] B. Costa, P. F. Pires, F. C. Delicato, and P. Merson. “Evaluating REST architectures - Approach, tooling and guidelines.” In: *Journal of Systems and Software*. 2016. doi: 10.1016/j.jss.2015.09.039.

[13] Express.js. *Express - Node.js web application framework*. 2017.

[14] M. Focus. *Silk test automation for web, mobile and enterprise apps*. 2021. url: https://www.microfocus.com/en-us/products/silk-test/overview. (accessed: 20.01.2021).

[15] O. Foundation. *Node.js*. 2021. url: https://nodejs.org/en/. (accessed: 29.12.2020).

[16] R. Framework. *Robot Framework*. url: https://robotframework.org/. (accessed: 20.01.2021).

[17] Guru99. *What is Automation Testing?* 2021. url: https://www.guru99.com/automation-testing.html. (accessed: 20.01.2021).

[18] R. Hametner, D. Winkler, and A. Zoitl. “Agile testing concepts based on keyword-driven testing for industrial automation systems.” In: *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*. 2012, pp. 3727–3732. doi: 10.1109/IECON.2012.6389298.

[19] F. Hermans, M. Pinzger, and A. Van Deursen. “Domain-specific languages in practice: A user study on the success factors.” In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2009. isbn: 3642044247. doi: 10.1007/978-3-642-04425-0_33.

[20] A. Holmes and M. Kellogg. “Automating functional tests using Selenium.” In: *AGILE 2006 (AGILE’06)*. 2006, 6 pp.–275. doi: 10.1109/AGILE.2006.19.

[21] S. Holmes. *Mongoose for Application Development*. 2013. isbn: 978-1-78216-819-5. arXiv: arXiv:1011.1669v3.

[22] L. Hu, N. Xie, Z. Kuang, and K. Zhao. “Review of cyber-physical system architecture.” In: *Proceedings - 2012 15th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops, ISORCW 2012*. 2012. isbn: 9780769546698. doi: 10.1109/ISORCW.2012.15.

[23] F. Inc. *React - A JavaScript library for building user interfaces*. 2021. url: https://reactjs.org/. (accessed: 30.12.2020).

[24] A. Javeed. “Performance Optimization Techniques for ReactJS.” In: *Proceedings of 2019 3rd IEEE International Conference on Electrical, Computer and Communication Technologies, ICECCT 2019*. 2019. isbn: 9781538681572. doi: 10.1109/ICECCT.2019.8869134.

[25] L. Jian-Ping, L. Juan-Juan, and W. Dong-Long. “Application Analysis of Automated Testing Framework Based on Robot.” In: *2012 Third International Conference on Networking and Distributed Computing*. 2012, pp. 194–197. doi: 10.1109/ICNDC.2012.53.

[26] Jingfan Tang, Xiaohua Cao, and A. Ma. "Towards adaptive framework of keyword driven automation testing." In: *2008 IEEE International Conference on Automation and Logistics*. 2008, pp. 1631–1636. doi: 10.1109/ICAL.2008.4636415.

[27] json.org. *JSON*. 2021. url: https://www.json.org/json-en.html. (accessed: 30.12.2020).

[28] M. Kaur and R. Kumari. "Comparative Study of Automated Testing Tools: TestComplete and QuickTest Pro." In: *International Journal of Computer Applications* 24.1 (2011), pp. 1–7. doi: 10.5120/2918-3844.

[29] T. Kosar, S. Bohra, and M. Mernik. "Domain-Specific Languages: A Systematic Mapping Study." In: *Information and Software Technology* (2016). issn: 09505849. doi: 10.1016/j.infsof.2015.11.001.

[30] C. W. Krueger. "Software Reuse." In: *ACM Computing Surveys (CSUR)* (1992). issn: 15577341. doi: 10.1145/130844.130856.

[31] T. Lalwani. *QuickTest Professional Unplugged: 2nd Edition*. KnowledgeInbox, 2011. isbn: 0983675910.

[32] E. A. Lee. "Cyber physical systems: Design challenges." In: *Proceedings - 11th IEEE Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, ISORC 2008*. 2008. isbn: 9780769531328. doi: 10.1109/ISORC.2008.25.

[33] K. Lei, Y. Ma, and Z. Tan. "Performance comparison and evaluation of web development technologies in PHP, Python and Node.js." In: *Proceedings - 17th IEEE International Conference on Computational Science and Engineering, CSE 2014, Jointly with 13th IEEE International Conference on Ubiquitous Computing and Communications, IUCC 2014, 13th International Symposium on Pervasive Systems, Algorithms, and Networks, I-SPAN 2014 and 8th International Conference on Frontier of Computer Science and Technology, FCST 2014*. 2015. isbn: 9781479979813. doi: 10.1109/CSE.2014.142.

[34] P. Leitão. "Agent-based distributed manufacturing control: A state-of-the-art survey." In: *Engineering Applications of Artificial Intelligence* (2009). issn: 09521976. doi: 10.1016/j.engappai.2008.09.005.

[35] P. Leitão, A. W. Colombo, and S. Karnouskos. "Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges." In: *Computers in Industry* (2016). issn: 01663615. doi: 10.1016/j.compind.2015.08.004.

[36] T. Lima, A. Dantas, and L. Vasconcelos. "Usando o SilkTest para automatizar testes: um Relato de Experiência." In: *Icomp.Ufam.Edu.Br*. 2012.

[37] M. Mernik, J. Heering, and A. M. Sloane. "When and how to develop domain-specific languages." In: *ACM Computing Surveys* (2005). issn: 03600300. doi: 10.1145/1118890.1118892.

[38] Mozilla. *JavaScript*. 2021. url: https://developer.mozilla.org/pt-PT/docs/Web/JavaScript. (accessed: 30.12.2020).

[39] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng. "IoT Middleware: A Survey on Issues and Enabling Technologies." In: *IEEE Internet of Things Journal* (2017). issn: 23274662. doi: 10.1109/JIOT.2016.2615180.

[40] E. Obinna. *Use the React Profiler for Performance*. 2018.

[41] J. Palsberg and C. B. Jay. "The essence of the Visitor pattern." In: *Proceedings - International Computer Software and Applications Conference*. 1998. isbn: 0818685859. doi: 10.1109/CMPSAC.1998.716629.

[42] T. J. Parr and R. W. Quong. "ANTLR: A predicated LL(k) parser generator." In: *Software: Practice and Experience* (1995). issn: 1097024X. doi: 10.1002/spe.4380250705.

[43] T. Parr and K. Fisher. "LL(*): The foundation of the ANTLR parser generator." In: *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*. 2011. isbn: 9781450306638. doi: 10.1145/1993498.1993548.

[44] T. Parr, S. Harwell, and K. Fisher. "Adaptive LL(*) parsing." In: *ACM SIGPLAN Notices* (2014). issn: 0362-1340. doi: 10.1145/2714064.2660202.

[45] Pluralsight. *JavaScript*. 2021. url: https://www.javascript.com/. (accessed: 29.12.2020).

[46] P. Porter, S. Yang, and X. Xi. "The Design and Implementation of a RESTful IoT Service Using the MERN Stack." In: *Proceedings - 2019 IEEE 16th International Conference on Mobile Ad Hoc and Smart Systems Workshops, MASSW 2019*. 2019. isbn: 9781728141213. doi: 10.1109/MASSW.2019.00035.

[47] Ranorex. *Test Automation Tools*. 2021. url: https://www.ranorex.com/test-automation-tools/. (accessed: 20.01.2021).

[48] R. A. Razak and F. R. Fahrurazi. "Agile testing with Selenium." In: *2011 Malaysian Conference in Software Engineering*. 2011, pp. 217–219. doi: 10.1109/MySEC.2011.6140672.

[49] Selenium. *About Selenium*. url: https://www.selenium.dev/about/. (accessed: 20.01.2021).

[50] S. A. Seshia, S. Hu, W. Li, and Q. Zhu. "Design Automation of Cyber-Physical Systems: Challenges, Advances, and Opportunities." In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2017). issn: 02780070. doi: 10.1109/TCAD.2016.2633961.

[51] SmartBear. *TestComplete System Requirements*. 2020. url: https://support.smartbear.com/testcomplete/docs/general-info/system-requirements.html. (accessed: 20.01.2021).

[52] SmartBear. *TestComplete Automated UI Testing Tool*. 2021. url: https://smartbear.com/product/testcomplete/overview/. (accessed: 20.01.2021).

[53] V. Subramanian and V. Subramanian. "MongoDB." In: *Pro MERN Stack*. 2019. doi: 10.1007/978-1-4842-4391-6_6.

[54] S. Tilkov and S. Vinoski. “Node.js: Using JavaScript to build high-performance network programs.” In: *IEEE Internet Computing* (2010). issn: 10897801. doi: 10.1109/MIC.2010.145.

[55] G. Tomassetti. *The ANTLR Mega Tutorial*. 2021. url: https://tomassetti.me/antlr-mega-tutorial/. (accessed: 27.01.2021).

[56] Q. Tutorial. *Tutorialspoint*. url: https://www.tutorialspoint.com/qtp/index.htm. (accessed: 20.01.2021).

[57] V. Vyatkin. *Software engineering in industrial automation: State-of-the-art review*. 2013. doi: 10.1109/TII.2013.2258165.

[58] W3C. *HTML5: A vocabulary and associated APIs for HTML and XHTML*. 2010. url: https://www.w3.org/TR/2010/WD-html5-20100624/. (accessed: 30.12.2020).

[59] W3C. *Cascading Style Sheets*. 2021. url: https://www.w3.org/Style/CSS/Overview.en.html. (accessed: 30.12.2020).

[60] X. Zhou, X. Gou, T. Huang, and S. Yang. “Review on Testing of Cyber Physical Systems: Methods and Testbeds.” In: *IEEE Access* 6 (2018), pp. 52179–52194. doi: 10.1109/ACCESS.2018.2869834.

Appendix A

Backend code

This appendix contains the complete code for the Backend tier, that is, for the Server of the System. Although the examples specified and presented in the main document are more than sufficient to understand the work done, all the code has been exposed here as supporting documentation for this work. Thus, it is possible to reproduce the work done with the presented documentation and code.

Listing A.1: App.js

```
1  const createError = require("http-errors");
2  const express = require("express");
3  const path = require("path");
4  const logger = require("morgan");
5  const mongoose = require("mongoose");
6  const cors = require("cors");
7
8  mongoose
9     .connect("mongodb://localhost:27017/TSIM", {
10       useNewUrlParser: true,
11       useUnifiedTopology: true,
12    })
13    .then(() => console.log("Mongo Ready: " + mongoose.connection.readyState))
14    .catch((erro) => console.log("Mongo: erro na conexão: " + erro));
15
16 const apiRouter = require("./routes/api");
17
18 const app = express();
19
20 // view engine setup
21 app.set("views", path.join(__dirname, "views"));
22 app.set("view engine", "pug");
```

```
23
24 const corsOpts = {
25    origin: "*",
26    credentials: true,
27    methods: ["GET", "PUT", "POST", "DELETE", "OPTIONS"],
28    allowedHeaders: [
29       "Accept",
30       "Authorization",
31       "Cache-Control",
32       "Content-Type",
33       "DNT",
34       "If-Modified-Since",
35       "Keep-Alive",
36       "Origin",
37       "User-Agent",
38       "X-Requested-With",
39       "Content-Length",
40    ],
41 };
42 app.use(cors(corsOpts));
43 app.use(logger("dev"));
44 app.use(express.json());
45 app.use(express.urlencoded({ extended: false }));
46 app.use(express.static(path.join(__dirname, "public")));
47
48 app.use("/", apiRouter);
49
50 // catch 404 and forward to error handler
51 app.use(function (req, res, next) {
52    next(createError(404));
53 });
54
55 // error handler
56 app.use(function (err, req, res, next) {
57    // set locals, only providing error in development
58    res.locals.message = err.message;
59    res.locals.error = req.app.get("env") === "development" ? err : {};
60
61    // render the error page
62    res.status(err.status || 500);
63    res.render("error");
64 });
65
66 module.exports = app;
```

A.1 Models

Listing A.2: Configurations Model

```
const mongoose = require("mongoose");

const configurationSchema = new mongoose.Schema(
   {
      dbBinDir: { type: String, required: true },
      appDir: { type: String, required: false },
      userConfigurationID: { type: String, required: false },
      backupDir: { type: String, required: false },
   },
   { versionKey: false }
);

module.exports = mongoose.model("configuration", configurationSchema);
```

Listing A.3: Packages Model

```
const mongoose = require("mongoose");

const packageSchema = new mongoose.Schema(
   {
      active: { type: Boolean, default: true },
      name: { type: String, required: true },
      description: { type: String, required: false },
      code: { type: String, required: true },
      path: { type: String, required: true },
      tests: [{ type: mongoose.Schema.Types.ObjectId, required: true }],
   },
   { versionKey: false }
);

module.exports = mongoose.model("package", packageSchema);
```

Listing A.4: Reports Model

```
const mongoose = require("mongoose");

const testsSchema = new mongoose.Schema(
   {
      id_test: { type: mongoose.Schema.Types.ObjectId, required: true },
      module: { type: String, required: true },
      name: { type: String, required: true },
      result: { type: String, required: true },
      message: { type: String, required: false },
```

```
      runtime: { type: Number, required: true },
      resultValue: { type: String, required: false },
   },
   { versionKey: false }
);

const reportSchema = new mongoose.Schema(
   {
      id_user: { type: String, required: true },
      date: { type: String, required: true },
      results: [testsSchema],
   },
   { versionKey: false }
);

module.exports = mongoose.model("report", reportSchema);
```

Listing A.5: Schedules Model

```
const mongoose = require("mongoose");

const scheduleSchema = new mongoose.Schema(
   {
      hour: { type: String, required: true },
      active: { type: Boolean, required: true },
      tests: [{ type: mongoose.Schema.Types.ObjectId, required: true }],
      packages: [{ type: mongoose.Schema.Types.ObjectId, required: true }],
   },
   { versionKey: false }
);

module.exports = mongoose.model("schedule", scheduleSchema);
```

Listing A.6: Tests Model

```
const mongoose = require("mongoose");

const testSchema = new mongoose.Schema(
   {
      id: { type: Number, required: true },
      module: { type: String, required: true },
      name: { type: String, required: true },
      description: { type: String, required: false },
      defaultParam: { type: String, required: false },
      active: { type: Boolean, default: true },
   },
   { versionKey: false }
```

```
);

module.exports = mongoose.model("test", testSchema);
```

A.2 Grammar

Listing A.7: Lexer

```
lexer grammar TestLexer;

NEXT : '->' ;
AND : '&' ;
OR : '|' ;

IF : '?' ;
ELSE : ':' ;

RPAREN : ')' ;
LPAREN : '(' ;

END : ';' ;

KEYWORD : ([A-Za-z]+([/ _-][A-Za-z]+)*)
          ;

WS
     : [ \r\n\t] -> skip
     ;
```

Listing A.8: Parser

```
parser grammar TestParser;

options {
   tokenVocab=TestLexer;
}

test
   : statement END
   ;

statement
   : condition #Conditional
   | seq #Sequence
   ;
```

```
15
16 condition
17    : expr IF statement ELSE statement #IfElse
18    | expr IF statement #If
19    ;
20
21 seq
22    : KEYWORD (NEXT statement)*
23    ;
24
25 expr
26    : LPAREN KEYWORD (AND KEYWORD)* RPAREN #And
27    | LPAREN KEYWORD (OR KEYWORD)* RPAREN #Or
28    ;
```

Listing A.9: Visitor

```
1  const antlr4 = require("antlr4/index");
2
3  const get_id = (list, name) => {
4     let test = list.filter((t) => t.name == name);
5     return test[0]._id;
6  };
7
8  const getId = (list, name) => {
9     let test = list.filter((t) => t.name == name);
10    return test[0].id;
11 };
12
13 const getDefaultParam = (list, name) => {
14    let test = list.filter((t) => t.name == name);
15    return test[0].defaultParam;
16 };
17
18 const getModule = (list, name) => {
19    let test = list.filter((t) => t.name == name);
20    return test[0].module;
21 };
22
23 // This class defines a complete generic visitor for a parse tree produced by TestParser.
24
25 function TestParserVisitor(listOfTests) {
26    antlr4.tree.ParseTreeVisitor.call(this);
27    this.tabs = 4;
28    this.auxOp = 0;
29    this.auxList = [];
```

```
30    this.tests = [];
31    this.auxTests = listOfTests;
32    this.res =
33       'const TsimTest = require("./../TsimTest/TsimTest");\n\nmodule.exports.execute = () =>
          ↪ {\n    let results = [];\n\n';
34    this.getRes = () => {
35       return this.res;
36    };
37    this.getTests = () => {
38       return this.tests;
39    };
40    return this;
41 }
42
43 TestParserVisitor.prototype = Object.create(
44    antlr4.tree.ParseTreeVisitor.prototype
45 );
46 TestParserVisitor.prototype.constructor = TestParserVisitor;
47
48 // Visit a parse tree produced by TestParser#test.
49 TestParserVisitor.prototype.visitTest = function (ctx) {
50    this.visit(ctx.statement());
51    this.res += "\n" + " ".repeat(this.tabs) + "return results;\n};\n";
52    return "";
53 };
54
55 // Visit a parse tree produced by TestParser#Conditional.
56 TestParserVisitor.prototype.visitConditional = function (ctx) {
57    return this.visit(ctx.condition());
58 };
59
60 // Visit a parse tree produced by TestParser#Sequence.
61 TestParserVisitor.prototype.visitSequence = function (ctx) {
62    return this.visit(ctx.seq());
63 };
64
65 // Visit a parse tree produced by TestParser#IfElse.
66 TestParserVisitor.prototype.visitIfElse = function (ctx) {
67    this.visit(ctx.expr());
68    let t = " ".repeat(this.tabs);
69    this.auxList.map((n) => {
70       let id = getId(this.auxTests, n);
71       let param = getDefaultParam(this.auxTests, n);
72       let module = getModule(this.auxTests, n);
73       n += "";
```

```
74        let countOccorrences = this.tests.filter(t => t === get_id(this.auxTests, n)).length
75        let k = n.replace(/\/|\-| /g, "_");
76        this.res += t + "let " + k + id + countOccorrences + " = TsimTest.RunTest(" + id + ",\""
              ↪  + param + "\");\n";
77        this.res += t + k + id + countOccorrences + '.id_test = "' + get_id(this.auxTests, n) +
              ↪ '";\n';
78        this.res += t + k + id + countOccorrences + '.name = "' + n + '";\n';
79        this.res += t + k + id + countOccorrences + '.module = "' + module + '";\n';
80        this.res += t + "results.push(" + k + id + countOccorrences + ");\n";
81        this.tests.push(get_id(this.auxTests, n));
82     });
83     this.res += "\n" + t + "if (";
84     if (this.auxOp == 0) {
85        this.auxList.map((n) => {
86           let id = getId(this.auxTests, n);
87           n += "";
88           let countOccorrences = this.tests.filter(t => t === get_id(this.auxTests, n)).length
                 ↪ -1
89           let k = n.replace(/\/|\-| /g, "_");
90           this.res += k + id + countOccorrences + '.result === "success" && ';
91        });
92     } else {
93        this.auxList.map((n) => {
94           let id = getId(this.auxTests, n);
95           n += "";
96           let countOccorrences = this.tests.filter(t => t === get_id(this.auxTests, n)).length
                 ↪ -1
97           k = n.replace(/\/|\-| /g, "_");
98           this.res += k + id + countOccorrences + '.result === "success" || ';
99        });
100    }
101    this.auxList = [];
102    this.res = this.res.substring(0, this.res.length - 4);
103    this.res += ") {\n";
104    this.tabs += 4;
105    this.visit(ctx.statement(0));
106    this.tabs -= 4;
107    this.res += t + "}\n" + t + "else {\n";
108    this.tabs += 4;
109    this.visit(ctx.statement(1));
110    this.tabs -= 4;
111    this.res += t + "}\n";
112    return "";
113 };
114
```

```
115 // Visit a parse tree produced by TestParser#If.
116 TestParserVisitor.prototype.visitIf = function (ctx) {
117    this.visit(ctx.expr());
118    let t = " ".repeat(this.tabs);
119    this.auxList.map((n) => {
120       let id = getId(this.auxTests, n);
121       let param = getDefaultParam(this.auxTests, n);
122       let module = getModule(this.auxTests, n);
123       n += "";
124       let countOccorrences = this.tests.filter(t => t === get_id(this.auxTests, n)).length
125       k = n.replace(/\/|\-| /g, "_");
126       this.res += t + "let " + k + id + countOccorrences + " = TsimTest.RunTest(" + id + ",\""
              ↪  + param + "\");\n";
127       this.res +=
128          t + k + id + countOccorrences + '.id_test = "' + get_id(this.auxTests, n) + '";\n';
129       this.res += t + k + id + countOccorrences + '.name = "' + n + '";\n';
130       this.res += t + k + id + countOccorrences + '.module = "' + module + '";\n';
131       this.res += t + "results.push(" + k + id + countOccorrences + ");\n";
132       this.tests.push(get_id(this.auxTests, n));
133    });
134    this.res += "\n" + t + "if (";
135    if (this.auxOp == 0) {
136       this.auxList.map((n) => {
137          let id = getId(this.auxTests, n);
138          n += "";
139          let countOccorrences = this.tests.filter(t => t === get_id(this.auxTests, n)).length
                 ↪ -1
140          k = n.replace(/\/|\-| /g, "_");
141          this.res += k + id + countOccorrences + '.result === "success" && ';
142       });
143    } else {
144       this.auxList.map((n) => {
145          let id = getId(this.auxTests, n);
146          n += "";
147          let countOccorrences = this.tests.filter(t => t === get_id(this.auxTests, n)).length
                 ↪ -1
148          k = n.replace(/\/|\-| /g, "_");
149          this.res += k + id + countOccorrences + '.result === "success" || ';
150       });
151    }
152    this.auxList = [];
153    this.res = this.res.substring(0, this.res.length - 4);
154    this.res += ") {\n";
155    this.tabs += 4;
156    this.visit(ctx.statement());
```

```
157     this.tabs -= 4;
158     this.res += t + "}\n";
159     return "";
160 };
161 
162 // Visit a parse tree produced by TestParser#seq.
163 TestParserVisitor.prototype.visitSeq = function (ctx) {
164     let n = ctx.KEYWORD() + "";
165     let id = getId(this.auxTests, n);
166     let param = getDefaultParam(this.auxTests, n);
167     let module = getModule(this.auxTests, n);
168     let t = " ".repeat(this.tabs);
169     let countOccorrences = this.tests.filter(t => t === get_id(this.auxTests, n)).length
170     k = n.replace(/\/|\-| /g, "_");
171     this.res += t + "let " + k + id + countOccorrences + " = TsimTest.RunTest(" + id + ",\"" +
        ↪ param + "\");\n";
172     this.res += t + k + id + countOccorrences + '.id_test = "' + get_id(this.auxTests, n) + '
        ↪ ";\n';
173     this.res += t + k + id + countOccorrences + '.name = "' + n + '";\n';
174     this.res += t + k + id + countOccorrences + '.module = "' + module + '";\n';
175     this.res += t + "results.push(" + k + id + countOccorrences + ");\n";
176     this.tests.push(get_id(this.auxTests, n));
177     this.visitChildren(ctx);
178     return "";
179 };
180 
181 // Visit a parse tree produced by TestParser#And.
182 TestParserVisitor.prototype.visitAnd = function (ctx) {
183     this.auxOp = 0;
184     for (let i = 0; i < ctx.KEYWORD().length; i++) {
185         this.auxList.push(ctx.KEYWORD(i));
186     }
187     return "";
188 };
189 
190 // Visit a parse tree produced by TestParser#Or.
191 TestParserVisitor.prototype.visitOr = function (ctx) {
192     this.auxOp = 1;
193     for (let i = 0; i < ctx.KEYWORD().length; i++) {
194         this.auxList.push(ctx.KEYWORD(i));
195     }
196     return "";
197 };
198 
199 exports.TestParserVisitor = TestParserVisitor;
```

A.3 Controllers

Listing A.10: Configurations Controller

```
const Configuration = require("../models/configurations");

module.exports.getConfigurations = () => {
   return Configuration.find().exec();
};

module.exports.insertConfiguration = (configuration) => {
   let c = new Configuration(configuration);
   return c.save();
};

module.exports.updateConfiguration = (idConfiguration, configuration) => {
   return Configuration.findOneAndUpdate({ _id: idConfiguration }, configuration);
};
```

Listing A.11: Packages Controller

```
const fs = require("fs");
const antlr4 = require("antlr4");
const Lexer = require("../public/Grammar/TestLexer").TestLexer;
const Parser = require("../public/Grammar/TestParser").TestParser;
const Visitor = require("../public/Grammar/TestParserVisitor")
   .TestParserVisitor;
const Package = require("../models/packages");
const Test = require("./tests");
const scripts_path = "./../tsim-tests-api/public/Packages/";

module.exports.getPackages = () => {
   return Package.find().exec();
};

module.exports.getPackage = async (idPackage) => {
   let p = await Package.findOne({ _id: idPackage }).exec();
   let script = fs.readFileSync(scripts_path + p.path, "utf8");
   return {
      _id: p._id,
      name: p.name,
      description: p.description,
      code: p.code,
      script: script,
      tests: p.tests,
   };
};
```

```
27
28 module.exports.insertPackage = async (package) => {
29    let chars = new antlr4.InputStream(package.script);
30    let lexer = new Lexer(chars);
31    let tokens = new antlr4.CommonTokenStream(lexer);
32    let parser = new Parser(tokens);
33    parser.buildParseTrees = true;
34    let tree = parser.test();
35
36    if (tree.parser._syntaxErrors === 0) {
37       let listOfTests = await Test.getTests();
38       let visitor = new Visitor(listOfTests);
39       visitor.visitTest(tree);
40       let textFile = visitor.getRes() + "";
41       let t = visitor.getTests();
42       let tests = t.filter(function (elem, pos) {
43          return t.indexOf(elem) == pos;
44       });
45       let fileName = package.name.toLowerCase().replace(/\s/g, '_') + ".js";
46       let filePath = scripts_path + fileName;
47
48       fs.writeFile(filePath, textFile, "utf8", function (err) {
49          if (err) throw err;
50          let t = new Package({
51             name: package.name,
52             description: package.description,
53             code: package.script,
54             path: fileName,
55             tests: tests,
56          });
57          return t.save();
58       });
59    } else {
60       return { errors: tree.parser._syntaxErrors };
61    }
62 };
63
64 module.exports.updatePackage = async (idPackage, package) => {
65    // Insert New
66    let chars = new antlr4.InputStream(package.script);
67    let lexer = new Lexer(chars);
68    let tokens = new antlr4.CommonTokenStream(lexer);
69    let parser = new Parser(tokens);
70    parser.buildParseTrees = true;
71    let tree = parser.test();
```

```
72
73    if (tree.parser._syntaxErrors === 0) {
74       let listOfTests = await Test.getTests();
75       let visitor = new Visitor(listOfTests);
76       visitor.visitTest(tree);
77       let textFile = visitor.getRes() + "";
78       let t = visitor.getTests();
79       let tests = t.filter(function (elem, pos) {
80          return t.indexOf(elem) == pos;
81       });
82       let fileName = package.name.toLowerCase().replace(/\s/g, '_') + ".js";
83       let filePath = scripts_path + fileName;
84
85       // Delete old
86       let p = await Package.findOne({ _id: idPackage }).exec();
87       try {
88          fs.unlinkSync(scripts_path + p.path);
89          let del = await Package.deleteOne({ _id: idPackage });
90       } catch (err) {
91          throw err;
92       }
93
94       fs.writeFile(filePath, textFile, "utf8", function (err) {
95          if (err) throw err;
96          let t = new Package({
97             _id: idPackage,
98             name: package.name,
99             description: package.description,
100            code: package.script,
101            path: fileName,
102            tests: tests,
103            active: true
104         });
105         return t.save();
106      });
107   } else {
108      return { errors: tree.parser._syntaxErrors };
109   }
110};
111
112module.exports.deactivatePackageAndRemoveTest = async (idPackage, package, idTest, testName)
    ↪ => {
113   return Package.findOneAndUpdate({ _id: idPackage }, {
114      active: false,
115      name: package.name,
```

```
116       description: package.description,
117       code: package.code.replace(new RegExp(testName,"g"),''),
118       path: package.path,
119       tests: package.tests.filter(t => t !== idTest),
120    });
121 };
122
123 module.exports.deletePackage = async (idPackage) => {
124    let p = await Package.findOne({ _id: idPackage }).exec();
125    try {
126       fs.unlinkSync(scripts_path + p.path);
127       return await Package.deleteOne({ _id: idPackage });
128    } catch (err) {
129       throw err;
130    }
131 };
132
133 module.exports.runPackage = async (idPackage) => {
134    let package = await Package.findOne({ _id: idPackage }).exec();
135    if(package){
136       const file = require("./../public/Packages/" + package.path);
137       return await file.execute();
138    }
139    return {};
140 };
```

Listing A.12: Reports Controller

```
1  const Report = require("../models/reports");
2
3  module.exports.getReports = () => {
4     return Report.find().sort({ date: -1 }).exec();
5  };
6
7  module.exports.getReport = (idReport) => {
8     return Report.findOne({ _id: idReport }).exec();
9  };
10
11 module.exports.insertReport = (report) => {
12    let r = new Report(report);
13    return r.save();
14 };
15
16 module.exports.updateReport = (idReport, report) => {
17    return Report.findOneAndUpdate({ _id: idReport }, report);
18 };
```

```
19
20 module.exports.deleteReport = (idReport) => {
21    return Report.deleteOne({ _id: idReport });
22 };
```

Listing A.13: Schedules Controller

```
1  const Schedule = require("../models/schedules");
2
3  module.exports.getSchedules = () => {
4     return Schedule.find().sort({ hour: 1 }).exec();
5  };
6
7  module.exports.getSchedule = (idSchedule) => {
8     return Schedule.findOne({ _id: idSchedule }).exec();
9  };
10
11 module.exports.insertSchedule = (schedule) => {
12    let s = new Schedule(schedule);
13    return s.save();
14 };
15
16 module.exports.updateSchedule = (idSchedule, schedule) => {
17    return Schedule.findOneAndUpdate({ _id: idSchedule }, schedule);
18 };
19
20 module.exports.removeTestFromAll = (idTest) => {
21    return Schedule.updateMany(
22       { },
23       { $pull: { tests: idTest } }
24    ).exec();
25 };
26
27 module.exports.removePackageFromAll = (idPackage) => {
28    return Schedule.updateMany(
29       { },
30       { $pull: { packages: idPackage } }
31    ).exec();
32 };
33
34 module.exports.deleteSchedule = (idSchedule) => {
35    return Schedule.deleteOne({ _id: idSchedule });
36 };
```

Listing A.14: Tests Controller

```
1  const Test = require("../models/tests");
```

```
2
3  module.exports.getTests = () => {
4     return Test.find({active: true}).sort({ id: 1 }).exec();
5  };
6
7  module.exports.getTest = (idTest) => {
8     return Test.findOne({ _id: idTest }).exec();
9  };
10
11 module.exports.insertTest = (test) => {
12    let t = new Test(test);
13    return t.save();
14 };
15
16 module.exports.updateTest = (idTest, newTest) => {
17    return Test.findOneAndUpdate({ _id: idTest }, newTest);
18 };
19
20 module.exports.runTest = async (driversDirectory, idTest, defaultParam) => {
21    let test = await Test.findOne({ _id: idTest }).exec();
22    let startTime = process.hrtime()
23    exec(`${driversDirectory}\\${test.module} "${idTest}" "${defaultParam}"`, (err, stdout,
          ↪ stderr) => {
24       if (err) return stderr
25       else {
26          let endTime = process.hrtime(startTime)
27          let result = JSON.parse(stdout)
28          result.runtime = (endTime[1] / 1000000).toFixed(3);
29          result.id_test = idTest;
30          result.module = test.module;
31          result.name = test.name;
32          return result;
33       }
34    })
35 };
```

A.4 Routes

Listing A.15: API Routes

```
1  const express = require("express");
2  const router = express.Router();
3  const exec = require('child_process').exec;
4  const fs = require('fs');
```

```
5  const jwt = require("jsonwebtoken");
6  const cron = require("node-cron");
7  const _ = require("underscore")
8  const Tests = require("../controllers/tests");
9  const Packages = require("../controllers/packages");
10 const Reports = require("../controllers/reports");
11 const Schedules = require("../controllers/schedules");
12 const Configurations = require("../controllers/configurations");
13 const TsimTest = require("./../public/TsimTest/TsimTest");
14
15 const checkForUpdates = async () => {
16    let newTests = await TsimTest.GetListOfTests();
17    let oldTests = await Tests.getTests();
18    let oldIds = oldTests.map(t => t.id)
19    let newIds = newTests.map(t => t.id)
20    let updates = 0
21    let unrwap = ({id, module, name, description, defaultParam}) => ({id, module, name,
          ↪ description, defaultParam})
22
23    newTests.map(test => {
24       if(oldIds.includes(test.id)){
25          let old = oldTests.filter(t => t.id === test.id)[0]
26          let checkOld = unrwap(old)
27          if(!(_.isEqual(test, checkOld))){
28             Tests.updateTest(old._id, test)
29                .then()
30                .catch((error) => res.status(500).jsonp(error));
31             updates = 1
32          }
33       }
34       else{
35          Tests.insertTest(test)
36             .then()
37             .catch((error) => res.status(500).jsonp(error));
38          updates = 1
39       }
40    })
41
42    oldTests.map(old => {
43       if(!newIds.includes(old.id)){
44          Tests.updateTest(old._id, Object.assign(old, {active: false}))
45             .then((dados) => {
46                Schedules.removeTestFromAll(old._id)
47                   .then()
48                   .catch((error) => res.status(500).jsonp(error));
```

```
49                Packages.getPackages()
50                   .then((packages) => {
51                      packages.map(p => {
52                         if(p.tests.includes(old._id)){
53                            Schedules.removePackageFromAll(p._id)
54                               .then()
55                               .catch((error) => res.status(500).jsonp(error));
56                            Packages.deactivatePackageAndRemoveTest(p._id, p, old._id, old.name
                                 ↪ )
57                               .then()
58                               .catch((error) => res.status(500).jsonp(error));
59                         }
60                      })
61                   })
62                   .catch((error) => res.status(500).jsonp(error));
63             })
64             .catch((error) => res.status(500).jsonp(error));
65          updates = 1
66       }
67    })
68
69    return updates
70 }
71
72 const generateBackupName = (options) => {
73    var tempDate = new Date();
74    let tmpDay = tempDate.getDate() + "";
75    let day = tmpDay.length === 1 ? "0" + tmpDay : tmpDay;
76    let tmpMonth = tempDate.getMonth() + 1 + "";
77    let month = tmpMonth.length === 1 ? "0" + tmpMonth : tmpMonth;
78    let year = tempDate.getFullYear() + "";
79    var tempHour = tempDate.getHours() + "";
80    var tempMin = tempDate.getMinutes() + "";
81    var tempSec = tempDate.getSeconds() + "";
82    let hour = tempHour.length === 1 ? "0" + tempHour : tempHour;
83    let min = tempMin.length === 1 ? "0" + tempMin : tempMin;
84    let sec = tempSec.length === 1 ? "0" + tempSec : tempSec;
85    var date = year + month + day + hour + min + sec;
86    return "Backup_v" + date + "_"+ (options.configurations ? "c" : "") + (options.reports ? "r
        ↪ " : "") + (options.schedules ? "s" : "") + (options.packages ? "p" : "");
87 };
88
89 const getDate = () => {
90    var tempDate = new Date();
91    let tmpDay = tempDate.getDate() + "";
```

```
92    let day = tmpDay.length === 1 ? "0" + tmpDay : tmpDay;
93    let tmpMonth = tempDate.getMonth() + 1 + "";
94    let month = tmpMonth.length === 1 ? "0" + tmpMonth : tmpMonth;
95    let year = tempDate.getFullYear() + "";
96    var tempHour = tempDate.getHours() + "";
97    var tempMin = tempDate.getMinutes() + "";
98    let hour = tempHour.length === 1 ? "0" + tempHour : tempHour;
99    let min = tempMin.length === 1 ? "0" + tempMin : tempMin;
100   var date = year + "-" + month + "-" + day + " " + hour + ":" + min;
101   return date;
102 };
103
104 const getHour = () => {
105   var tempDate = new Date();
106   var tempHour = tempDate.getHours() + "";
107   var tempMin = tempDate.getMinutes() + "";
108   let hour = tempHour.length === 1 ? "0" + tempHour : tempHour;
109   let min = tempMin.length === 1 ? "0" + tempMin : tempMin;
110   var date = hour + ":" + min;
111   return date;
112 };
113
114 cron.schedule("* * * * *", function () {
115   Schedules.getSchedules()
116      .then(async (data) => {
117         let date = getDate();
118         let hour = getHour();
119         let list = data.filter((s) => s.hour === hour && s.active);
120         let listOfTests = list.map((s) => s.tests);
121         let listOfPackages = list.map((s) => s.packages);
122
123         if (listOfTests.length > 0 || listOfPackages.length > 0) {
124            let tests = listOfTests[0];
125            let packages = listOfPackages[0];
126            let results = [];
127
128            for (let i = 0; i < tests.length; i++) {
129               let r = await Tests.runTest(tests[i]);
130               results.push(r);
131            }
132
133            for (let i = 0; i < packages.length; i++) {
134               let r = await Packages.runPackage(packages[i]);
135               if(r[0])
136                  r.map((res) => results.push(res));
```

```
137             }
138
139             await Reports.insertReport({
140                id_user: 5000,
141                date: date,
142                results: results,
143             });
144          }
145       })
146       .catch((error) => console.log(error));
147 });
148
149 router.get("/", function (req, res) {
150    checkForUpdates()
151       .then((data) => res.jsonp(data))
152       .catch((error) => res.status(500).jsonp(error));
153 });
154
155 router.get("/backups", function (req, res) {
156    Configurations.getConfigurations()
157       .then((data) =>
158          fs.readdir(data[0].backupDir, (err, files) => {
159             if(err)
160                res.status(500).jsonp(err)
161             else{
162                let validFiles = []
163                files.forEach(file => {
164                   let name = file.split(".zip")[0];
165                   let words = name.split("_v")
166                   if(words[1]){
167                      let ids = words[1].split("_")
168                      if(words[0] === "Backup" && /^\d+$/.test(ids[0]) && /[a-zA-Z]+/.test(ids
                            ↪ [1])){
169                         validFiles.push(file)
170                      }
171                   }
172                });
173                res.jsonp(validFiles)
174             }
175          })
176       )
177       .catch((error) => res.status(500).jsonp(error));
178 });
179
180 router.post("/backups", async function (req, res) {
```

```
181    Configurations.getConfigurations()
182       .then((data) => {
183          let backupName = generateBackupName(req.body)
184          fs.mkdir(`${data[0].backupDir}\\${backupName}`, function(err) {
185             if (err) res.status(500).jsonp(err)
186             else {
187                fs.writeFile(`${data[0].backupDir}\\${backupName}\\info.json`, JSON.stringify(
                       ↪ req.body, null, 4), (err) => {
188                   if (err) res.status(500).jsonp(err)
189                   else {
190                      exec(`${data[0].appDir}\\bin\\exportBackup.bat "${data[0].appDir}" "${
                             ↪ data[0].dbBinDir}" "${data[0].backupDir}" "${req.body.packages}" "
                             ↪ ${req.body.reports}" "${req.body.schedules}" "${req.body.
                             ↪ configurations}" ${backupName}`, (err, stdout, stderr) => {
191                         if (err) res.status(500).jsonp(err)
192                         else res.jsonp(stdout)
193                      })
194                   }
195                })
196             }
197          })
198       })
199       .catch((error) => res.status(500).jsonp(error));
200 });
201
202 router.post("/restore/:backup", function (req, res) {
203    let strOptions = req.params.backup.split("_")[2].split(".zip")[0]
204    let reports = strOptions.includes("r")
205    let packages = strOptions.includes("p")
206    let schedules = strOptions.includes("s")
207    let configurations = strOptions.includes("c")
208    Configurations.getConfigurations()
209       .then((data) => {
210          exec(`${data[0].appDir}\\bin\\restoreBackup.bat "${data[0].appDir}" "${data[0].
                 ↪ dbBinDir}" "${data[0].backupDir}" ${req.params.backup.split(".")[0]} "${
                 ↪ packages}" "${reports}" "${schedules}" "${configurations}"`, (err, stdout,
                 ↪ stderr) => {
211             if (err) res.status(500).jsonp(err)
212             else res.jsonp(stdout)
213          });
214       })
215       .catch((error) => res.status(500).jsonp(error));
216 });
217
218 router.post("/login", function (req, res) {
```

```
219    Configurations.getConfigurations()
220       .then((data) => {
221          if (req.body.id == data[0].userConfigurationID) {
222             res.jsonp(
223                jwt.sign({ id: req.body.id, role: "admin" }, "tsim_secret", {
224                   expiresIn: "1h",
225                })
226             );
227          } else {
228             res.jsonp(
229                jwt.sign({ id: req.body.id, role: "worker" }, "tsim_secret", {
230                   expiresIn: "1h",
231                })
232             );
233          }
234       })
235       .catch((error) => res.status(500).jsonp(error));
236 });
237
238 // Configuration Requests
239 router.get("/configurations", function (req, res) {
240    Configurations.getConfigurations()
241       .then((data) => res.jsonp(data[0]))
242       .catch((error) => res.status(500).jsonp(error));
243 });
244
245 router.post("/configurations", function (req, res) {
246    Configurations.insertConfiguration(req.body)
247       .then((data) => res.jsonp(data))
248       .catch((error) => res.status(500).jsonp(error));
249 });
250
251 router.put("/configurations/:idConfiguration", function (req, res) {
252    Configurations.updateConfiguration(req.params.idConfiguration, req.body)
253       .then((data) => res.jsonp(data))
254       .catch((error) => res.status(500).jsonp(error));
255 });
256
257 // Test Requests
258 router.get("/tests", function (req, res) {
259    Tests.getTests()
260       .then((data) => res.jsonp(data))
261       .catch((error) => res.status(500).jsonp(error));
262 });
263
```

```
264 router.get("/tests/:idTest", function (req, res) {
265    Tests.getTest(req.params.idTest)
266       .then((data) => res.jsonp(data))
267       .catch((error) => res.status(500).jsonp(error));
268 });
269
270 router.post("/tests/run", async function (req, res) {
271    let tests = req.body.tests;
272    let results = [];
273
274    for (let i = 0; i < tests.length; i++) {
275       let r = await Tests.runTest(tests[i]);
276       results.push(r);
277    }
278    await Reports.insertReport({
279       id_user: req.body.id_user,
280       date: req.body.date,
281       results: results,
282    });
283    res.jsonp(results);
284 });
285
286 // Packages Requests
287 router.get("/packages", function (req, res) {
288    Packages.getPackages()
289       .then((data) => res.jsonp(data))
290       .catch((error) => res.status(500).jsonp(error));
291 });
292
293 router.get("/packages/:idPackage", function (req, res) {
294    Packages.getPackage(req.params.idPackage)
295       .then((data) => res.jsonp(data))
296       .catch((error) => res.status(500).jsonp(error));
297 });
298
299 router.post("/packages", function (req, res) {
300    Packages.insertPackage(req.body)
301       .then((data) => res.jsonp(data))
302       .catch((error) => res.status(500).jsonp(error));
303 });
304
305 router.post("/packages/run", async function (req, res) {
306    let packages = req.body.packages;
307    let newReport = {
308       id_user: req.body.id_user,
```

```
309       date: req.body.date,
310       results: [],
311    };
312    for (let i = 0; i < packages.length; i++) {
313       let r = await Packages.runPackage(packages[i]);
314       r.map((res) => newReport.results.push(res));
315    }
316    await Reports.insertReport(newReport);
317    res.jsonp(newReport.results);
318 });
319
320 router.put("/packages/:idPackage", function (req, res) {
321    Packages.updatePackage(req.params.idPackage, req.body)
322       .then((data) => res.jsonp(data))
323       .catch((error) => res.status(500).jsonp(error));
324 });
325
326 router.delete("/packages/:idPackage", function (req, res, next) {
327    Packages.deletePackage(req.params.idPackage)
328       .then((data) => res.jsonp(data))
329       .catch((error) => res.status(500).jsonp(error));
330 });
331
332 // Reports Requests
333 router.get("/reports", function (req, res) {
334    Reports.getReports()
335       .then((data) => res.jsonp(data))
336       .catch((error) => res.status(500).jsonp(error));
337 });
338
339 router.get("/reports/:idReport", function (req, res) {
340    Reports.getReport(req.params.idReport)
341       .then((data) => res.jsonp(data))
342       .catch((error) => res.status(500).jsonp(error));
343 });
344
345 router.post("/reports", function (req, res) {
346    Reports.insertReport(req.body)
347       .then((data) => res.jsonp(data))
348       .catch((error) => res.status(500).jsonp(error));
349 });
350
351 // Schedules Requests
352 router.get("/schedules", function (req, res) {
353    Schedules.getSchedules()
```

```
354       .then((data) => res.jsonp(data))
355       .catch((error) => res.status(500).jsonp(error));
356 });
357
358 router.get("/schedules/:idSchedule", function (req, res) {
359    Schedules.getSchedule(req.params.idSchedule)
360       .then((data) => res.jsonp(data))
361       .catch((error) => res.status(500).jsonp(error));
362 });
363
364 router.post("/schedules", function (req, res) {
365    Schedules.insertSchedule(req.body)
366       .then((data) => res.jsonp(data))
367       .catch((error) => res.status(500).jsonp(error));
368 });
369
370 router.put("/schedules/:idSchedule", function (req, res) {
371    Schedules.updateSchedule(req.params.idSchedule, req.body)
372       .then((data) => res.jsonp(data))
373       .catch((error) => res.status(500).jsonp(error));
374 });
375
376 router.delete("/schedules/:idSchedule", function (req, res, next) {
377    Schedules.deleteSchedule(req.params.idSchedule)
378       .then((data) => res.jsonp(data))
379       .catch((error) => res.status(500).jsonp(error));
380 });
381
382 module.exports = router;
```

Appendix B

Frontend code

This appendix contains the complete code for the Frontend tier, that is, for the System Interface. Although the examples specified and presented in the main document are more than sufficient to understand the work done, all the code has been exposed here as supporting documentation for this work. Thus, it is possible to reproduce the work done with the presented documentation and code.

B.1 API Requests

Listing B.1: Methods for retrieving data from the API via HTTP requests

```
import axios from "axios";
import { getToken } from "../auth/auth";

const url = "http://localhost:5000";

export const getDate = () => {
   var tempDate = new Date();
   let tmpDay = tempDate.getDate() + "";
   let day = tmpDay.length === 1 ? "0" + tmpDay : tmpDay;
   let tmpMonth = tempDate.getMonth() + 1 + "";
   let month = tmpMonth.length === 1 ? "0" + tmpMonth : tmpMonth;
   let year = tempDate.getFullYear() + "";
   var tempHour = tempDate.getHours() + "";
   var tempMin = tempDate.getMinutes() + "";
   let hour = tempHour.length === 1 ? "0" + tempHour : tempHour;
   let min = tempMin.length === 1 ? "0" + tempMin : tempMin;
   var date = year + "-" + month + "-" + day + " " + hour + ":" + min;
   return date;
};
```

```
20
21 export const checkUpdates = async () => {
22    try {
23       const response = await axios.get(`${url}/`);
24       return response.data;
25    } catch (error) {
26       const statusCode = error.response ? error.response.status : 500;
27       throw new Error(statusCode.toString());
28    }
29 };
30
31 export const login = async (id) => {
32    try {
33       const response = await axios.post(`${url}/login`, {
34          id: id,
35       });
36       return response.data;
37    } catch (error) {
38       const statusCode = error.response ? error.response.status : 500;
39       throw new Error(statusCode.toString());
40    }
41 };
42
43 export const getConfigurations = async () => {
44    try {
45       const response = await axios.get(`${url}/configurations`);
46       return response.data;
47    } catch (error) {
48       const statusCode = error.response ? error.response.status : 500;
49       throw new Error(statusCode.toString());
50    }
51 };
52
53 export const updateConfigurations = async (id, newConfigs) => {
54    try {
55       const response = await axios.put(`${url}/configurations/${id}`, newConfigs);
56       return response.data;
57    } catch (error) {
58       const statusCode = error.response ? error.response.status : 500;
59       throw new Error(statusCode.toString());
60    }
61 };
62
63 export const getBackups = async () => {
64    try {
```

```
65        const response = await axios.get(`${url}/backups`);
66        return response.data;
67     } catch (error) {
68        return []
69     }
70  };
71
72  export const makeBackup = async (options) => {
73     try {
74        const response = await axios.post(`${url}/backups`, options);
75        return response.data;
76     } catch (error) {
77        return -1;
78     }
79  };
80
81  export const restoreBackup = async (backup) => {
82     try {
83        const response = await axios.post(`${url}/restore/${backup.split(".")[0]}`);
84        return response.data;
85     } catch (error) {
86        const statusCode = error.response ? error.response.status : 500;
87        throw new Error(statusCode.toString());
88     }
89  };
90
91  export const getTests = async () => {
92     try {
93        const response = await axios.get(`${url}/tests`);
94        return response.data;
95     } catch (error) {
96        const statusCode = error.response ? error.response.status : 500;
97        throw new Error(statusCode.toString());
98     }
99  };
100
101 export const getPackages = async () => {
102    try {
103       const response = await axios.get(`${url}/packages`);
104       return response.data;
105    } catch (error) {
106       const statusCode = error.response ? error.response.status : 500;
107       throw new Error(statusCode.toString());
108    }
109 };
```

```
110
111 export const runTests = async (tests) => {
112    try {
113       const d = getDate();
114       const token = await getToken();
115       const response = await axios.post(`${url}/tests/run`, {
116          tests: tests,
117          id_user: token.id,
118          date: d,
119       });
120       return response.data;
121    } catch (error) {
122       const statusCode = error.response ? error.response.status : 500;
123       throw new Error(statusCode.toString());
124    }
125 };
126
127 export const runPackages = async (packages) => {
128    try {
129       const d = getDate();
130       const token = await getToken();
131       const response = await axios.post(`${url}/packages/run`, {
132          packages: packages,
133          id_user: token.id,
134          date: d,
135       });
136       return response.data;
137    } catch (error) {
138       const statusCode = error.response ? error.response.status : 500;
139       throw new Error(statusCode.toString());
140    }
141 };
142
143 export const getReports = async () => {
144    try {
145       const response = await axios.get(`${url}/reports`);
146       return response.data;
147    } catch (error) {
148       const statusCode = error.response ? error.response.status : 500;
149       throw new Error(statusCode.toString());
150    }
151 };
152
153 export const getSchedules = async () => {
154    try {
```

```
155       const response = await axios.get(`${url}/schedules`);
156       return response.data;
157    } catch (error) {
158       const statusCode = error.response ? error.response.status : 500;
159       throw new Error(statusCode.toString());
160    }
161 };
162 
163 export const insertSchedule = async (newSchedule) => {
164    try {
165       const response = await axios.post(`${url}/schedules`, newSchedule);
166       return response.data;
167    } catch (error) {
168       const statusCode = error.response ? error.response.status : 500;
169       throw new Error(statusCode.toString());
170    }
171 };
172 
173 export const updateSchedule = async (id, newSchedule) => {
174    try {
175       const response = await axios.put(`${url}/schedules/${id}`, newSchedule);
176       return response.data;
177    } catch (error) {
178       const statusCode = error.response ? error.response.status : 500;
179       throw new Error(statusCode.toString());
180    }
181 };
182 
183 export const deleteSchedule = async (id) => {
184    try {
185       const response = await axios.delete(`${url}/schedules/${id}`);
186       return response.data;
187    } catch (error) {
188       const statusCode = error.response ? error.response.status : 500;
189       throw new Error(statusCode.toString());
190    }
191 };
192 
193 export const deletePackage = async (id) => {
194    try {
195       const response = await axios.delete(`${url}/packages/${id}`);
196       return response.data;
197    } catch (error) {
198       const statusCode = error.response ? error.response.status : 500;
199       throw new Error(statusCode.toString());
```

```
200     }
201 };
202
203 export const updatePackage = async (id, newPackage) => {
204     try {
205         const response = await axios.put(`${url}/packages/${id}`, newPackage);
206         return response.data;
207     } catch (error) {
208         return { errors: 1 };
209     }
210 };
211
212 export const insertPackage = async (newPackage) => {
213     try {
214         const response = await axios.post(`${url}/packages`, newPackage);
215         return response.data;
216     } catch (error) {
217         return { errors: 1 };
218     }
219 };
```

B.2 Authentication

Listing B.2: Authentication.js

```
1  import Cookie from "js-cookie";
2  import decode from "jwt-decode";
3
4  export const getToken = () => {
5      const token = Cookie.get("tsimToken");
6      return token ? JSON.parse(token) : null;
7  };
8
9  export const addToken = (encodedToken) => {
10     deleteToken();
11     const decodedToken = decode(encodedToken);
12     const token = {
13         encoded: encodedToken,
14         ...decodedToken,
15     };
16     Cookie.set("tsimToken", token, { expires: 1 });
17     return token;
18 };
19
```

```
export const deleteToken = () => Cookie.remove("tsimToken");

export const isAuthenticated = () => {
   const token = getToken();
   if (!token) return false;
   return token.role;
};
```

B.3 Routing

Listing B.3: HomeRoute.js

```
import React from "react";
import { Route, Redirect } from "react-router-dom";
import { isAuthenticated } from "../../auth/auth";

const PrivateRoute = ({ children, path, ...rest }) => {
   let role = isAuthenticated();
   return (
      <Route
         {...rest}
         render={(props) =>
            role === "admin" ? (
               <Redirect to="/configs" />
            ) : role === "worker" ? (
               <Redirect to="/execution" />
            ) : (
               children
            )
         }
      />
   );
};

export default PrivateRoute;
```

Listing B.4: PrivateRoute.js

```
import React from "react";
import { Route, Redirect } from "react-router-dom";
import { isAuthenticated } from "../../auth/auth";

const PrivateRoute = ({ children, role, path, ...rest }) => (
   <Route
      {...rest}
```

```
      render={(props) =>
         isAuthenticated() === role ? children : <Redirect to="/" />
      }
   />
);

export default PrivateRoute;
```

B.4 Pages Code

Listing B.5: index.html

```
<!DOCTYPE html>
<html lang="pt">
   <head>
      <meta charset="utf-8" />
      <meta name="viewport" content="width=device-width, initial-scale=1" />
      <title>TSIM Tests Aplication</title>
   </head>
   <body style="font-family: Arial, Helvetica, sans-serif;">
      <div id="root"></div>
   </body>
</html>
```

Listing B.6: serviceWorker.js

```
const isLocalhost = Boolean(
   window.location.hostname === "localhost" ||
      window.location.hostname === "[::1]" ||
      window.location.hostname.match(
         /^127(?:\.(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)){3}$/
      )
);

export function register(config) {
   if (process.env.NODE_ENV === "production" && "serviceWorker" in navigator) {
      const publicUrl = new URL(process.env.PUBLIC_URL, window.location.href);
      if (publicUrl.origin !== window.location.origin) {
         return;
      }

      window.addEventListener("load", () => {
         const swUrl = `${process.env.PUBLIC_URL}/service-worker.js`;

         if (isLocalhost) {
```

```
            checkValidServiceWorker(swUrl, config);
            navigator.serviceWorker.ready.then(() => {
               console.log(
                  "This web app is being served cache-first by a service " +
                     "worker. To learn more, visit https://bit.ly/CRA-PWA"
               );
            });
         } else {
            registerValidSW(swUrl, config);
         }
      });
   }
}

function registerValidSW(swUrl, config) {
   navigator.serviceWorker
      .register(swUrl)
      .then((registration) => {
         registration.onupdatefound = () => {
            const installingWorker = registration.installing;
            if (installingWorker == null) {
               return;
            }
            installingWorker.onstatechange = () => {
               if (installingWorker.state === "installed") {
                  if (navigator.serviceWorker.controller) {
                     console.log(
                        "New content is available and will be used when all " +
                           "tabs for this page are closed. See https://bit.ly/CRA-PWA."
                     );

                     if (config && config.onUpdate) {
                        config.onUpdate(registration);
                     }
                  } else {
                     console.log("Content is cached for offline use.");

                     if (config && config.onSuccess) {
                        config.onSuccess(registration);
                     }
                  }
               }
            };
         };
      })
```

```
65       .catch((error) => {
66          console.error("Error during service worker registration:", error);
67       });
68 }
69
70 function checkValidServiceWorker(swUrl, config) {
71    fetch(swUrl, {
72       headers: { "Service-Worker": "script" },
73    })
74       .then((response) => {
75          const contentType = response.headers.get("content-type");
76          if (
77             response.status === 404 ||
78             (contentType != null &&
79                contentType.indexOf("javascript") === -1)
80          ) {
81             navigator.serviceWorker.ready.then((registration) => {
82                registration.unregister().then(() => {
83                   window.location.reload();
84                });
85             });
86          } else {
87             registerValidSW(swUrl, config);
88          }
89       })
90       .catch(() => {
91          console.log(
92             "No internet connection found. App is running in offline mode."
93          );
94       });
95 }
96
97 export function unregister() {
98    if ("serviceWorker" in navigator) {
99       navigator.serviceWorker.ready
100          .then((registration) => {
101             registration.unregister();
102          })
103          .catch((error) => {
104             console.error(error.message);
105          });
106    }
107 }
```

Listing B.7: index.js

```
import React from "react";
import ReactDOM from "react-dom";
import App from "./App";
import * as serviceWorker from "./serviceWorker";

ReactDOM.render(<App />, document.getElementById("root"));

serviceWorker.unregister();
```

Listing B.8: App.js

```
import React, {useEffect} from "react";
import { BrowserRouter, Switch } from "react-router-dom";
import Execution from "./pages/Execution";
import Configs from "./pages/Configs";
import Login from "./pages/Login";
import PrivateRoute from "./components/routes/PrivateRoute";
import HomeRoute from "./components/routes/HomeRoute";
import {checkUpdates} from "./api/api"

export default function App() {

   useEffect(() => {
      async function fetchData() {
         await checkUpdates();
      }
      fetchData();
   }, []);

   return (
      <BrowserRouter>
         <Switch>
            <PrivateRoute path="/execution" role="worker">
               <Execution />
            </PrivateRoute>
            <PrivateRoute path="/configs" role="admin">
               <Configs />
            </PrivateRoute>
            <HomeRoute path="/">
               <Login />
            </HomeRoute>
         </Switch>
      </BrowserRouter>
   );
}
```

Listing B.9: Login Component

```
import React, { useState } from "react";
import { useHistory } from "react-router-dom";
import AppBar from "@material-ui/core/AppBar";
import Toolbar from "@material-ui/core/Toolbar";
import Typography from "@material-ui/core/Typography";
import Button from "@material-ui/core/Button";
import TextField from "@material-ui/core/TextField";
import Card from '@material-ui/core/Card';
import Grid from '@material-ui/core/Grid';
import CardContent from '@material-ui/core/CardContent';
import { login, getConfigurations } from "../api/api";
import { addToken } from "../auth/auth";

export default function Login() {
   const history = useHistory();
   const [id, setId] = useState("");
   const [error, setError] = useState(false);

   const handleChange = (event) => {
      const { name, value } = event.target;
      if (name === "id") setId(value);
   };

   const handleClickExecution = async () => {
      let configurations = await getConfigurations();
      if (id === "" || id === configurations.userConfigurationID) {
         setError(true);
      } else {
         setError(false);
         let res = await login(id);
         await addToken(res);
         history.push("/execution");
      }
   };

   const handleClickConfigs = async () => {
      let configurations = await getConfigurations();
      if (id === "" || id !== configurations.userConfigurationID) {
         setError(true);
      } else {
         setError(false);
         let res = await login(id);
         await addToken(res);
         history.push("/configs");
```

```
45        }
46    };
47
48    return (
49       <div style={{ textAlign: ”center” }}>
50          <AppBar position=”static” style={{ alignItems: ”center” }}>
51             <Toolbar>
52                <Typography variant=”h3”>
53                   Sistema de Autodiagnóstico dos Sistemas de Teste
54                   Automático
55                </Typography>
56             </Toolbar>
57          </AppBar>
58          <Grid
59             container
60             spacing={0}
61             direction=”column”
62             alignItems=”center”
63             justify=”center”
64             style={{ minHeight: '70vh' }}
65          >
66             <Grid item xs={12}>
67                <Card style={{width: 600, textAlign: ”center”}}>
68                   <CardContent>
69                      <h1>Escolha o modo de execução:</h1>
70                      <TextField
71                         name=”id”
72                         error={error}
73                         label=”ID Funcionário”
74                         variant=”outlined”
75                         value={id}
76                         onChange={handleChange}
77                      />
78                      <br />
79                      <br />
80                      <Button
81                         style={{ marginRight: 20 }}
82                         variant=”contained”
83                         color=”primary”
84                         size=”large”
85                         disableElevation
86                         onClick={handleClickExecution}
87                      >
88                         Execução
89                      </Button>
```

```
                    <Button
                        style={{ marginLeft: 20 }}
                        variant="contained"
                        color="primary"
                        size="large"
                        disableElevation
                        onClick={handleClickConfigs}
                    >
                        Configuração
                    </Button>
                </CardContent>
            </Card>
        </Grid>
    </Grid>
  </div>
 );
}
```

B.4.1 Execution Components

Listing B.10: Execution.js

```
import React from "react";
import { useHistory } from "react-router-dom";
import { deleteToken } from "../auth/auth";
import Button from "@material-ui/core/Button";
import Typography from "@material-ui/core/Typography";
import AppBar from "@material-ui/core/AppBar";
import Toolbar from "@material-ui/core/Toolbar";
import Test from "../components/execution/tests/Test";

export default function Execution() {
  const history = useHistory();

  const logout = async () => {
    await deleteToken();
    history.push("/");
  };

  return (
    <div>
      <AppBar position="static">
        <Toolbar>
          <Typography variant="h4" style={{ flexGrow: 1, textAlign: "center" }}>
            Execução de Testes
```

```
            </Typography>
            <Button color="inherit" onClick={logout}>
               Sair
            </Button>
         </Toolbar>
      </AppBar>
      <Test />
   </div>
  );
}
```

Listing B.11: Tests.js

```
import React, { useEffect, useState } from "react";
import TransferList from "./TransferList";
import { getTests, getPackages, runTests, runPackages } from "../../../api/api";
import SplitPane from "react-split-pane";
import ResultsTable from "./ResultsTable";
import CircularProgress from "@material-ui/core/CircularProgress";

export default function Testar() {
   const [loading, setLoading] = useState(false);
   const [executingTests, setExecutingTests] = useState(false);
   const [executingPackages, setExecutingPackages] = useState(false);
   const [tests, setTests] = useState([]);
   const [packages, setPackages] = useState([]);
   const [results, setResults] = useState([]);

   const executeTests = async (listOfTests) => {
      setExecutingTests(true);
      let results = await runTests(listOfTests);
      setExecutingTests(false);
      setResults(results);
   };

   const executePackages = async (listOfPackages) => {
      setExecutingPackages(true);
      let results = await runPackages(listOfPackages);
      setExecutingPackages(false);
      setResults(results);
   };

   useEffect(() => {
      setLoading(true);
      async function fetchData() {
         let tests = await getTests();
```

```
34          setTests(tests);
35          let packages = await getPackages();
36          setPackages(packages.filter(p => p.active));
37          setLoading(false);
38       }
39       fetchData();
40    }, []);
41
42    let showResults =
43       results.length > 0 ? (
44          <ResultsTable results={results} setResults={setResults} />
45       ) : (
46          ""
47       );
48
49    return (
50       <div>
51          {loading ? (
52             <h3 style={{ textAlign: "center" }}>A carregar dados...</h3>
53          ) : (
54             <SplitPane split="vertical" defaultSize={"50%"}>
55                <div
56                   style={{
57                      textAlign: "center",
58                      borderRight: "1px solid #3f51b5",
59                   }}
60                >
61                   <h2>Testes Primitivos</h2>
62                   <TransferList tests={tests} execute={executeTests} />
63                   <br />
64                   <br />
65                   {executingTests ? <CircularProgress /> : ""}
66                </div>
67                <div style={{ textAlign: "center" }}>
68                   <h2>Pacotes de Testes</h2>
69                   <TransferList
70                      tests={packages}
71                      execute={executePackages}
72                   />
73                   <br />
74                   <br />
75                   {executingPackages ? <CircularProgress /> : ""}
76                </div>
77             </SplitPane>
78          )}
```

```
79          {showResults}
80       </div>
81    );
82 }
```

Listing B.12: TransferList.js

```
1  import React, { useState } from "react";
2  import { makeStyles } from "@material-ui/core/styles";
3  import Grid from "@material-ui/core/Grid";
4  import List from "@material-ui/core/List";
5  import Card from "@material-ui/core/Card";
6  import CardHeader from "@material-ui/core/CardHeader";
7  import ListItem from "@material-ui/core/ListItem";
8  import ListItemText from "@material-ui/core/ListItemText";
9  import ListItemIcon from "@material-ui/core/ListItemIcon";
10 import Checkbox from "@material-ui/core/Checkbox";
11 import Button from "@material-ui/core/Button";
12 import Divider from "@material-ui/core/Divider";
13 import ResponsiveButton from "./ResponsiveButton";
14
15 const useStyles = makeStyles((theme) => ({
16    root: {
17       margin: "auto",
18    },
19    cardHeader: {
20       padding: theme.spacing(2, 3),
21    },
22    list: {
23       width: 300,
24       height: 350,
25       backgroundColor: theme.palette.background.paper,
26       overflow: "auto",
27    },
28    button: {
29       margin: theme.spacing(1, 0),
30    },
31 }));
32
33 function not(a, b) {
34    return a.filter((value) => b.indexOf(value) === -1);
35 }
36
37 function intersection(a, b) {
38    return a.filter((value) => b.indexOf(value) !== -1);
39 }
```

```
40
41 function union(a, b) {
42    return [...a, ...not(b, a)];
43 }
44
45 function getTestName(list, id) {
46    return list.filter((t) => t._id === id)[0].name;
47 }
48
49 export default function TransferList(props) {
50    const classes = useStyles();
51    const [checked, setChecked] = useState([]);
52    const [left, setLeft] = useState(props.tests.map((t) => t._id));
53    const [right, setRight] = useState([]);
54
55    const leftChecked = intersection(checked, left);
56    const rightChecked = intersection(checked, right);
57
58    const handleToggle = (value) => () => {
59       const currentIndex = checked.indexOf(value);
60       const newChecked = [...checked];
61
62       if (currentIndex === -1) {
63          newChecked.push(value);
64       } else {
65          newChecked.splice(currentIndex, 1);
66       }
67
68       setChecked(newChecked);
69    };
70
71    const cleanList = () => {
72       setChecked([]);
73       setLeft(left.concat(right));
74       setRight([]);
75    };
76
77    const numberOfChecked = (items) => intersection(checked, items).length;
78
79    const handleToggleAll = (items) => () => {
80       if (numberOfChecked(items) === items.length) {
81          setChecked(not(checked, items));
82       } else {
83          setChecked(union(checked, items));
84       }
```

```
85    };
86
87    const handleCheckedRight = () => {
88       setRight(right.concat(leftChecked));
89       setLeft(not(left, leftChecked));
90       setChecked(not(checked, leftChecked));
91    };
92
93    const handleCheckedLeft = () => {
94       setLeft(left.concat(rightChecked));
95       setRight(not(right, rightChecked));
96       setChecked(not(checked, rightChecked));
97    };
98
99    const customList = (title, items) => (
100      <Card>
101         <CardHeader
102            className={classes.cardHeader}
103            avatar={
104               <Checkbox
105                  color="primary"
106                  onClick={handleToggleAll(items)}
107                  checked={
108                     numberOfChecked(items) === items.length &&
109                     items.length !== 0
110                  }
111                  indeterminate={
112                     numberOfChecked(items) !== items.length &&
113                     numberOfChecked(items) !== 0
114                  }
115                  disabled={items.length === 0}
116               />
117            }
118            title={title}
119            subheader={`${numberOfChecked(items)}/${
120               items.length
121            } selecionados`}
122         />
123         <Divider />
124         <List className={classes.list} dense component="div" role="list">
125            {items.map((value) => {
126               const labelId = `transfer-list-all-item-${value}-label`;
127
128               return (
129                  <ListItem
```

```
130                     key={value}
131                     role="listitem"
132                     button
133                     onClick={handleToggle(value)}
134                  >
135                     <ListItemIcon>
136                        <Checkbox
137                           color="primary"
138                           checked={checked.indexOf(value) !== -1}
139                           tabIndex={-1}
140                           disableRipple
141                           inputProps={{ "aria-labelledby": labelId }}
142                        />
143                     </ListItemIcon>
144                     <ListItemText
145                        id={labelId}
146                        primary={getTestName(props.tests, value)}
147                     />
148                  </ListItem>
149               );
150            })}
151         </List>
152      </Card>
153   );
154
155   return (
156      <div>
157         <Grid
158            container
159            spacing={2}
160            justify="center"
161            alignItems="center"
162            className={classes.root}
163         >
164            <Grid item>{customList("Lista de Testes", left)}</Grid>
165            <Grid item>
166               <Grid container direction="column" alignItems="center">
167                  <Button
168                     variant="outlined"
169                     className={classes.button}
170                     onClick={handleCheckedRight}
171                     disabled={leftChecked.length === 0}
172                  >
173                     &gt;
174                  </Button>
```

```
                  <Button
                     variant="outlined"
                     className={classes.button}
                     onClick={handleCheckedLeft}
                     disabled={rightChecked.length === 0}
                  >
                     &lt;
                  </Button>
               </Grid>
            </Grid>
            <Grid item>{customList("Executar", right)}</Grid>
         </Grid>
         <br />
         <ResponsiveButton
            tests={right}
            execute={props.execute}
            clean={cleanList}
         />
      </div>
   );
}
```

Listing B.13: ExecuteButton.js

```
import React from "react";
import Button from "@material-ui/core/Button";
import Dialog from "@material-ui/core/Dialog";
import DialogActions from "@material-ui/core/DialogActions";
import DialogContent from "@material-ui/core/DialogContent";
import DialogContentText from "@material-ui/core/DialogContentText";
import DialogTitle from "@material-ui/core/DialogTitle";
import useMediaQuery from "@material-ui/core/useMediaQuery";
import { useTheme } from "@material-ui/core/styles";

export default function ResponsiveButton(props) {
   const [open, setOpen] = React.useState(false);
   const [noTest, setNoTest] = React.useState(false);
   const theme = useTheme();
   const fullScreen = useMediaQuery(theme.breakpoints.down("sm"));

   const handleClickOpen = () => {
      props.tests.length === 0 ? setNoTest(true) : setOpen(true);
   };

   const handleClose = () => {
      setNoTest(false)
```

```
23       setOpen(false);
24    };
25
26    const handleExecute = () => {
27       setOpen(false);
28       props.clean();
29       props.execute(props.tests);
30    };
31
32    return (
33       <div>
34          <Button
35             variant="outlined"
36             color="primary"
37             onClick={handleClickOpen}
38          >
39             Executar
40          </Button>
41          <Dialog
42             disableBackdropClick={true}
43             fullScreen={fullScreen}
44             open={noTest}
45          >
46             <DialogTitle>
47                {"Tem que selecionar pelo menos um teste."}
48             </DialogTitle>
49             <DialogActions>
50                <Button onClick={handleClose} color="primary" autoFocus>
51                   Ok
52                </Button>
53             </DialogActions>
54          </Dialog>
55          <Dialog
56             disableBackdropClick={true}
57             fullScreen={fullScreen}
58             open={open}
59          >
60             <DialogTitle>
61                {"Tem a certeza que quer executar estes testes?"}
62             </DialogTitle>
63             <DialogContent>
64                <DialogContentText>
65                   Selecionou {props.tests.length} testes.
66                </DialogContentText>
67             </DialogContent>
```

```
            <DialogActions>
               <Button autoFocus onClick={handleClose} color="primary">
                  Não
               </Button>
               <Button onClick={handleExecute} color="primary" autoFocus>
                  Sim
               </Button>
            </DialogActions>
         </Dialog>
      </div>
   );
}
```

Listing B.14: ResultsTable.js

```
import React from "react";
import Button from "@material-ui/core/Button";
import Dialog from "@material-ui/core/Dialog";
import DialogActions from "@material-ui/core/DialogActions";
import DialogContent from "@material-ui/core/DialogContent";
import DialogTitle from "@material-ui/core/DialogTitle";
import Table from "@material-ui/core/Table";
import TableBody from "@material-ui/core/TableBody";
import TableCell from "@material-ui/core/TableCell";
import TableContainer from "@material-ui/core/TableContainer";
import TableHead from "@material-ui/core/TableHead";
import TableRow from "@material-ui/core/TableRow";
import Paper from "@material-ui/core/Paper";
import FiberManualRecordIcon from "@material-ui/icons/FiberManualRecord";

export default function ResultsTable(props) {
   const [open, setOpen] = React.useState(true);

   const handleClose = () => {
      props.setResults([]);
      setOpen(false);
   };

   return (
      <Dialog maxWidth="lg" disableBackdropClick={true} open={open}>
         <DialogTitle>{"Relatório dos Testes efetuados"}</DialogTitle>
         <DialogContent>
            <TableContainer component={Paper}>
               <Table>
                  <TableHead>
                     <TableRow>
```

```
32                         <TableCell>
33                            <b>Módulo</b>
34                         </TableCell>
35                         <TableCell align="center">
36                            <b>Teste</b>
37                         </TableCell>
38                         <TableCell align="center">
39                            <b>Tempo de Execução</b>
40                         </TableCell>
41                         <TableCell align="center">
42                            <b>Mensagem</b>
43                         </TableCell>
44                         <TableCell align="center">
45                            <b>Valor</b>
46                         </TableCell>
47                         <TableCell align="right">
48                            <b>Resultado</b>
49                         </TableCell>
50                      </TableRow>
51                   </TableHead>
52                   <TableBody>
53                      {props.results.map((test) => {
54                         return (
55                            <TableRow key={test._id}>
56                               <TableCell component="th" scope="row">
57                                  {test.module}
58                               </TableCell>
59                               <TableCell align="center">
60                                  {test.name}
61                               </TableCell>
62                               <TableCell align="center">
63                                  {test.runtime}ms
64                               </TableCell>
65                               <TableCell align="center">
66                                  {test.message}
67                               </TableCell>
68                               <TableCell align="center">
69                                  {test.resultValue}
70                               </TableCell>
71                               <TableCell align="right">
72                                  {test.result === "success"
73                                     ? "Passou"
74                                     : (test.result === "fail" ? "Falhou" : "Inconclusivo")}
75                                  {test.result === "success" ? (
76                                     <FiberManualRecordIcon
```

```
                                        style={{
                                            color: "green",
                                            fontSize: "12px",
                                            verticalAlign: "center",
                                        }}
                                    />
                                ) : ( test.result === "fail" ?
                                    <FiberManualRecordIcon
                                        style={{
                                            color: "red",
                                            fontSize: "12px",
                                            verticalAlign: "center",
                                        }}
                                    /> :
                                    <FiberManualRecordIcon
                                        style={{
                                            color: "yellow",
                                            fontSize: "12px",
                                            verticalAlign: "center",
                                        }}
                                    />
                                )}
                            </TableCell>
                        </TableRow>
                    );
                })}
            </TableBody>
        </Table>
    </TableContainer>
    </DialogContent>
    <DialogActions>
        <Button autoFocus onClick={handleClose} color="primary">
            Fechar
        </Button>
    </DialogActions>
</Dialog>
);
}
```

B.4.2 Configuration Components

Listing B.15: Configuration.js

```
import React, { useState } from "react";
import { useHistory } from "react-router-dom";
```

```
3  import { deleteToken } from "../auth/auth";
4  import Button from "@material-ui/core/Button";
5  import AppBar from "@material-ui/core/AppBar";
6  import Toolbar from "@material-ui/core/Toolbar";
7  import Schedules from "../components/configs/schedules/Schedules";
8  import Docs from "../components/configs/docs/Docs";
9  import Packages from "../components/configs/packages/Packages";
10 import Historico from "../components/execution/reports/Historico";
11 import Configuracoes from "../components/configs/configurations/Configurations";
12
13 export default function Configs() {
14    const history = useHistory();
15    const [section, setSection] = useState("Relatórios");
16
17    const logout = async () => {
18       await deleteToken();
19       history.push("/");
20    };
21
22    return (
23       <div>
24          <AppBar position="static">
25             <Toolbar>
26                <Button
27                   color="inherit"
28                   onClick={() => setSection("Relatórios")}
29                   style={{ flexGrow: 1 }}
30                >
31                   Relatórios
32                </Button>
33                <Button
34                   color="inherit"
35                   onClick={() => setSection("Agendamentos")}
36                   style={{ flexGrow: 1 }}
37                >
38                   Agendamentos
39                </Button>
40                <Button
41                   color="inherit"
42                   onClick={() => setSection("Testes")}
43                   style={{ flexGrow: 1 }}
44                >
45                   Gestão de Pacotes
46                </Button>
47                <Button
```

```
                  color="inherit"
                  onClick={() => setSection("Documentação")}
                  style={{ flexGrow: 1 }}
               >
                  Documentação
               </Button>
               <Button
                  color="inherit"
                  onClick={() => setSection("Configurações")}
                  style={{ flexGrow: 1 }}
               >
                  Configurações
               </Button>
               <Button color="inherit" onClick={logout}>
                  Sair
               </Button>
            </Toolbar>
         </AppBar>
         {
            section === "Agendamentos" ? <Schedules /> :
            section === "Testes" ? <Packages /> :
            section === "Documentação" ? <Docs /> :
            section === "Configurações" ? <Configuracoes /> : <Historico />
         }
      </div>
   );
}
```

Listing B.16: Reports.js

```
import React, { useEffect, useState } from "react";
import ReportsTable from "./ReportsTable";
import { getReports } from "../../../api/api";

export default function Historico() {
   const [loading, setLoading] = useState(false);
   const [data, setData] = useState([]);

   useEffect(() => {
      setLoading(true);
      async function fetchData() {
         let reports = await getReports();
         setData(reports);
         setLoading(false);
      }
      fetchData();
```

```
17    }, []);
18
19    return (
20       <div>
21          {loading ? (
22             <h3 style={{ textAlign: "center" }}>A carregar dados...</h3>
23          ) : (
24             <ReportsTable data={data} />
25          )}
26       </div>
27    );
28 }
```

Listing B.17: ReportsTableWithDetail.js

```
1  import React, {useState} from 'react';
2  import { makeStyles } from '@material-ui/core/styles';
3  import Paper from '@material-ui/core/Paper';
4  import Table from '@material-ui/core/Table';
5  import TableBody from '@material-ui/core/TableBody';
6  import TableCell from '@material-ui/core/TableCell';
7  import TableContainer from '@material-ui/core/TableContainer';
8  import TableHead from '@material-ui/core/TableHead';
9  import TablePagination from '@material-ui/core/TablePagination';
10 import TableRow from '@material-ui/core/TableRow';
11 import FiberManualRecordIcon from "@material-ui/icons/FiberManualRecord";
12 import IconButton from '@material-ui/core/IconButton';
13 import KeyboardArrowDownIcon from '@material-ui/icons/KeyboardArrowDown';
14 import KeyboardArrowUpIcon from '@material-ui/icons/KeyboardArrowUp';
15 import Box from '@material-ui/core/Box';
16 import Collapse from '@material-ui/core/Collapse';
17 import ReportDetail from "./ReportDetail"
18
19 const useStyles = makeStyles({
20    root: {
21       margin: "1%",
22       width: '98%',
23    },
24    container: {
25       maxHeight: 770,
26    },
27 });
28
29 const useRowStyles = makeStyles({
30    root: {
31       '& > *': {
```

```
32          borderBottom: 'unset',
33       },
34    },
35 });
36
37 function Row(props) {
38    const { row } = props;
39    const [open, setOpen] = useState(false);
40    const classes = useRowStyles();
41
42    return (
43       <React.Fragment>
44          <TableRow hover className={classes.root}>
45             <TableCell>{row.id_user}</TableCell>
46             <TableCell align="center">{row.date}</TableCell>
47             <TableCell align="center">{row.results.map(r => r.runtime).reduce((a, b) => a + b,
                   ↪ 0).toFixed(3)}ms</TableCell>
48             <TableCell align="center">
49                {row.results.filter(r => r.result === "success").length}/{row.results.length}{"
                      ↪ "}
50                <FiberManualRecordIcon
51                   style={{
52                      color: row.results.filter(r => r.result === "fail").length > 0
53                      ? "red"
54                      : row.results.filter(r => r.result === "inconclusive").length > 0
55                      ? "yellow"
56                      : "green",
57                      fontSize: "12px",
58                   }}
59                />
60             </TableCell>
61             <TableCell align="center">
62                <IconButton size="small" onClick={() => setOpen(!open)}>
63                   {open ? <KeyboardArrowUpIcon /> : <KeyboardArrowDownIcon />}
64                </IconButton>
65             </TableCell>
66          </TableRow>
67          <TableRow>
68             <TableCell style={{ paddingBottom: 0, paddingTop: 0 }} colSpan={6}>
69                <Collapse in={open} timeout="auto" unmountOnExit>
70                   <Box margin={1}>
71                      <ReportDetail report={row.results} />
72                   </Box>
73                </Collapse>
74             </TableCell>
```

```
75          </TableRow>
76       </React.Fragment>
77    );
78 }
79
80 export default function ReportsTable(props) {
81    const classes = useStyles();
82    const [page, setPage] = useState(0);
83    const [rowsPerPage, setRowsPerPage] = useState(11);
84
85    const handleChangePage = (event, newPage) => {
86       setPage(newPage);
87    };
88
89    const handleChangeRowsPerPage = (event) => {
90       setRowsPerPage(+event.target.value);
91       setPage(0);
92    };
93
94    return (
95       <Paper className={classes.root}>
96          <TableContainer className={classes.container}>
97             <Table stickyHeader>
98                <TableHead>
99                   <TableRow>
100                     <TableCell><b>ID Funcionário</b></TableCell>
101                     <TableCell align="center"><b>Data de Execução</b></TableCell>
102                     <TableCell align="center"><b>Tempo de Execução</b></TableCell>
103                     <TableCell align="center"><b>Resultados</b></TableCell>
104                     <TableCell align="center"/>
105                  </TableRow>
106               </TableHead>
107               <TableBody>
108                  {props.data.slice(page * rowsPerPage, page * rowsPerPage + rowsPerPage).map
                         ↪ ((row) => <Row row={row} />)}
109               </TableBody>
110            </Table>
111         </TableContainer>
112         <TablePagination
113            rowsPerPageOptions={[11]}
114            labelRowsPerPage="Relatórios por Página:"
115            component="div"
116            count={props.data.length}
117            rowsPerPage={rowsPerPage}
118            page={page}
```

```
            onChangePage={handleChangePage}
            onChangeRowsPerPage={handleChangeRowsPerPage}
         />
      </Paper>
   );
}
```

Listing B.18: ReportsDetail.js

```
import React from "react";
import Table from "@material-ui/core/Table";
import TableBody from "@material-ui/core/TableBody";
import TableCell from "@material-ui/core/TableCell";
import TableContainer from "@material-ui/core/TableContainer";
import TableHead from "@material-ui/core/TableHead";
import TableRow from "@material-ui/core/TableRow";
import Paper from "@material-ui/core/Paper";
import FiberManualRecordIcon from "@material-ui/icons/FiberManualRecord";

export default function ReportDetail(props) {
   return (
      <TableContainer component={Paper}>
         <Table>
            <TableHead>
               <TableRow>
                  <TableCell>
                     <b>Módulo</b>
                  </TableCell>
                  <TableCell align="center">
                     <b>Teste</b>
                  </TableCell>
                  <TableCell align="center">
                     <b>Tempo de Execução</b>
                  </TableCell>
                  <TableCell align="center">
                     <b>Mensagem</b>
                  </TableCell>
                  <TableCell align="center">
                     <b>Valor</b>
                  </TableCell>
                  <TableCell align="right">
                     <b>Resultado</b>
                  </TableCell>
               </TableRow>
            </TableHead>
            <TableBody>
```

```
38                {props.report.map((r) => {
39                   return (
40                      <TableRow key={r._id}>
41                         <TableCell component="th" scope="row">
42                            {r.module}
43                         </TableCell>
44                         <TableCell align="center">
45                            {r.name}
46                         </TableCell>
47                         <TableCell align="center">
48                            {r.runtime}ms
49                         </TableCell>
50                         <TableCell align="center">
51                            {r.message}
52                         </TableCell>
53                         <TableCell align="center">
54                            {r.resultValue}
55                         </TableCell>
56                         <TableCell align="right">
57                            {r.result === "success"
58                               ? "Passou"
59                               : ( r.result === "fail" ? "Falhou" : "Inconclusivo")}
60                            {r.result === "success" ? (
61                               <FiberManualRecordIcon
62                                  style={{
63                                     color: "green",
64                                     fontSize: "12px",
65                                     verticalAlign: "center",
66                                  }}
67                               />
68                            ) : (r.result === "fail" ?
69                               <FiberManualRecordIcon
70                                  style={{
71                                     color: "red",
72                                     fontSize: "12px",
73                                     verticalAlign: "center",
74                                  }}
75                               /> :
76                               <FiberManualRecordIcon
77                                  style={{
78                                     color: "yellow",
79                                     fontSize: "12px",
80                                     verticalAlign: "center",
81                                  }}
82                               />
```

```
                            )}
                        </TableCell>
                    </TableRow>
                );
            })}
        </TableBody>
    </Table>
</TableContainer>
);
}
```

Listing B.19: Schedules.js

```
import React, { useEffect, useState } from "react";
import { getSchedules, getTests, getPackages } from "../../../api/api";
import ListSchedules from "./ListSchedules";

export default function Schedules() {
    const [loading, setLoading] = useState(false);
    const [schedules, setSchedules] = useState([]);
    const [tests, setTests] = useState([]);
    const [packages, setPackages] = useState([]);

    useEffect(() => {
        setLoading(true);
        async function fetchData() {
            let res = await getSchedules();
            let t = await getTests();
            let p = await getPackages();
            setSchedules(res);
            setTests(t);
            setPackages(p.filter(pa => pa.active));
            setLoading(false);
        }
        fetchData();
    }, []);

    return (
        <div style={{ textAlign: "center" }}>
            {loading ? (
                <h3 style={{ textAlign: "center" }}>A carregar dados...</h3>
            ) : (
                <ListSchedules
                    schedules={schedules}
                    setSchedules={setSchedules}
                    tests={tests}
```

```
                packages={packages}
            />
        )}
    </div>
  );
}
```

Listing B.20: ListSchedules.js

```
import React, { useState } from "react";
import { makeStyles } from "@material-ui/core/styles";
import List from "@material-ui/core/List";
import ListItem from "@material-ui/core/ListItem";
import ListItemSecondaryAction from "@material-ui/core/ListItemSecondaryAction";
import ListItemText from "@material-ui/core/ListItemText";
import Switch from "@material-ui/core/Switch";
import DeleteIcon from "@material-ui/icons/Delete";
import IconButton from "@material-ui/core/IconButton";
import AddIcon from "@material-ui/icons/Add";
import Fab from "@material-ui/core/Fab";
import UpdateSchedule from "./UpdateSchedule";
import AddSchedule from "./AddSchedule";
import {
  updateSchedule,
  deleteSchedule,
  getSchedules,
  insertSchedule,
} from "../../../api/api";

const nextExecute = (h) => {
  var time = h.split(":");
  var hour = parseInt(time[0]);
  var min = parseInt(time[1]);
  var date = new Date();
  var hourNow = parseInt("" + date.getHours());
  var minNow = parseInt("" + date.getMinutes());
  let nextMin = 0;
  let nextHour = 0;
  var result = "";
  if (hour > hourNow) {
    nextHour = hour - hourNow;
    if (min >= minNow) nextMin = min - minNow;
    else nextMin = 60 - (minNow - min);
  } else if (hour < hourNow) {
    nextHour = 24 - (hourNow - hour);
    if (min >= minNow) nextMin = min - minNow;
```

```
38          else nextMin = 60 - (minNow - min);
39       } else {
40          if (min > minNow) {
41             nextHour = 0;
42             nextMin = min - minNow;
43          } else if (min < minNow) {
44             nextHour = 23;
45             nextMin = 60 - (minNow - min);
46          }
47       }
48       if (nextHour === 0 && nextMin === 0) result += "A executar";
49       else
50          result +=
51             "Irá executar em " + nextHour + " horas e " + nextMin + " Minutos";
52       return result;
53    };
54
55    const useStyles = makeStyles((theme) => ({
56       root: {
57          marginTop: "20px",
58          justifyContent: "center",
59          display: "flex",
60       },
61       list: {
62          width: "100%",
63          maxWidth: 600,
64          maxHeight: 710,
65          overflow: "auto",
66          backgroundColor: theme.palette.background.paper,
67       },
68       fab: {
69          margin: theme.spacing(3),
70       },
71    }));
72
73    export default function ListSchedules(props) {
74       const classes = useStyles();
75       const [add, setAdd] = useState(false);
76       const [update, setUpdate] = useState(false);
77       const [schedule, setSchedule] = useState("");
78
79       const handleUpdate = (s) => {
80          setSchedule(s);
81          setUpdate(true);
82       };
```

```
83
84    const handleToggle = async (i, s) => {
85       await updateSchedule(i, {
86          tests: s.tests,
87          packages: s.packages,
88          hour: s.hour,
89          active: !s.active,
90       });
91       let scheds = await getSchedules();
92       props.setSchedules(scheds);
93    };
94
95    const handleInsert = async (newSchedule) => {
96       await insertSchedule(newSchedule);
97       let scheds = await getSchedules();
98       props.setSchedules(scheds);
99    };
100
101   const handleUpdateSchedule = async (idSchedule, newSchedule) => {
102      await updateSchedule(idSchedule, newSchedule);
103      setUpdate(false);
104      let scheds = await getSchedules();
105      props.setSchedules(scheds);
106   };
107
108   const handleDelete = async (i) => {
109      await deleteSchedule(i);
110      let scheds = await getSchedules();
111      props.setSchedules(scheds);
112   };
113
114   return (
115      <div>
116         <div className={classes.root}>
117            <List className={classes.list}>
118               {props.schedules.map((s) => {
119                  return (
120                     <ListItem button onClick={() => handleUpdate(s)}>
121                        <ListItemText
122                           id={s._id}
123                           primary={s.hour}
124                           secondary={
125                              s.active
126                                 ? nextExecute(s.hour)
127                                 : "Desligado"
```

```
                        }
                    />
                    <ListItemSecondaryAction>
                        <Switch
                            color="primary"
                            edge="end"
                            onChange={() => handleToggle(s._id, s)}
                            checked={s.active}
                        />
                        <IconButton
                            onClick={() => handleDelete(s._id)}
                        >
                            <DeleteIcon color="action" />
                        </IconButton>
                    </ListItemSecondaryAction>
                </ListItem>
            );
        })}
    </List>
</div>
<div>
    <Fab
        color="primary"
        className={classes.fab}
        onClick={() => setAdd(true)}
    >
        <AddIcon />
    </Fab>
</div>
<div>
    {add ? (
        <AddSchedule
            add={add}
            setAdd={setAdd}
            tests={props.tests}
            handleInsert={handleInsert}
            packages={props.packages}
        />
    ) : (
        ""
    )}
    {update ? (
        <UpdateSchedule
            update={update}
            setUpdate={setUpdate}
```

```
                    schedule={schedule}
                    tests={props.tests}
                    packages={props.packages}
                    handleUpdate={handleUpdateSchedule}
                />
            ) : (
                ""
            )}
        </div>
    </div>
  );
}
```

Listing B.21: AddSchedule.js

```
import React, { useState } from "react";
import { makeStyles } from "@material-ui/core/styles";
import Button from "@material-ui/core/Button";
import Dialog from "@material-ui/core/Dialog";
import DialogActions from "@material-ui/core/DialogActions";
import DialogContent from "@material-ui/core/DialogContent";
import DialogTitle from "@material-ui/core/DialogTitle";
import useMediaQuery from "@material-ui/core/useMediaQuery";
import { useTheme } from "@material-ui/core/styles";
import Switch from "@material-ui/core/Switch";
import Checkbox from "@material-ui/core/Checkbox";
import TextField from "@material-ui/core/TextField";
import Alert from "@material-ui/lab/Alert";
import Table from '@material-ui/core/Table';
import TableBody from '@material-ui/core/TableBody';
import TableCell from '@material-ui/core/TableCell';
import TableContainer from '@material-ui/core/TableContainer';
import TableHead from '@material-ui/core/TableHead';
import TableRow from '@material-ui/core/TableRow';
import Paper from '@material-ui/core/Paper';
import IconButton from '@material-ui/core/IconButton';
import KeyboardArrowDownIcon from '@material-ui/icons/KeyboardArrowDown';
import KeyboardArrowUpIcon from '@material-ui/icons/KeyboardArrowUp';

const useStyles = makeStyles((theme) => ({
   form: {
      textAlign: "center",
   },
   formControl: {
      margin: theme.spacing(2),
      width: 300,
```

```
32    },
33 }));
34
35 export default function AddSchedule(props) {
36    const theme = useTheme();
37    const classes = useStyles();
38    const fullScreen = useMediaQuery(theme.breakpoints.down("sm"));
39    const [hour, setHour] = useState("08:00");
40    const [active, setActive] = useState(false);
41    const [tests, setTests] = useState([]);
42    const [packages, setPackages] = useState([]);
43    const [alert, setAlert] = useState(false);
44    const [openTests, setOpenTests] = useState(false);
45    const [openPackages, setOpenPackages] = useState(false);
46
47    const handleChangeHour = (event) => {
48       setHour(event.target.value);
49    };
50
51    const handleChangePackages = (event, value) => {
52       if(value === "Todos"){
53          if(props.packages.length === packages.length) setPackages([])
54          else setPackages(props.packages.map(p => p._id))
55       }
56       else{
57          packages.includes(value) === true
58             ? setPackages(
59                packages.filter((id) => {
60                   if (id === value) return false;
61                   else return true;
62                })
63             )
64             : setPackages(packages.concat(value));
65       }
66    };
67
68    const handleChangeTests = (event, value) => {
69       if(value === "Todos"){
70          if(props.tests.length === tests.length) setTests([])
71          else setTests(props.tests.map(t => t._id))
72       }
73       else{
74          tests.includes(value) === true
75             ? setTests(
76                tests.filter((id) => {
```

```
77                if (id === value) return false;
78                else return true;
79             })
80          )
81          : setTests(tests.concat(value));
82       }
83    };
84
85    const handleClose = () => {
86       props.setAdd(false);
87       setHour("08:00");
88       setActive(false);
89       setTests([]);
90       setPackages([]);
91    };
92
93    const handleInsert = async () => {
94       if (hour === "" || (tests.length === 0 && packages.length === 0)) {
95          setAlert(true);
96       } else {
97          let newSchedule = {
98             hour: hour,
99             active: active,
100            tests: tests,
101            packages: packages
102         };
103         props.handleInsert(newSchedule);
104         props.setAdd(false);
105         setHour("08:00");
106         setActive(false);
107         setTests([]);
108         setPackages([]);
109      }
110   };
111
112   let displayAlert =
113      alert === false ? (
114         ""
115      ) : (
116         <div>
117            <Alert
118               severity="error"
119               variant="filled"
120               onClose={() => {
121                  setAlert(false);
```

```
122                }}
123             >
124                Tem que preencher os campos todos!
125             </Alert>
126             <br />
127          </div>
128       );
129
130    return (
131       <Dialog
132          disableBackdropClick
133          fullScreen={fullScreen}
134          open={props.add}
135          onClose={handleClose}
136       >
137          <DialogTitle>Adicionar Novo Agendamento</DialogTitle>
138          <DialogContent>
139             <div className={classes.form}>
140                <TextField
141                   label="Hora"
142                   type="time"
143                   name="hour"
144                   value={hour}
145                   onChange={handleChangeHour}
146                   inputProps={{ step: 900 }}
147                />
148                <Switch
149                   color="primary"
150                   onChange={() => setActive((prev) => !prev)}
151                   checked={active}
152                />
153             </div>
154             <br/>
155             <TableContainer component={Paper} style={{maxHeight: 300, width: 350}}>
156                <Table size="small">
157                   <TableHead>
158                      <TableRow onClick={() => setOpenTests(!openTests)} style={{cursor: "
                            ↪ pointer"}}>
159                         <TableCell>
160                            <IconButton >
161                               {openTests ? <KeyboardArrowUpIcon fontSize="large" /> : <
                                     ↪ KeyboardArrowDownIcon fontSize="large" />}
162                            </IconButton>
163                         </TableCell>
164                         <TableCell>
```

```
165                         <h3>Testes a executar</h3>
166                      </TableCell>
167                   </TableRow>
168                </TableHead>
169                <TableBody>
170                   {
171                      openTests ?
172                      <TableRow key={"Todos"}>
173                         <TableCell>
174                            <Checkbox
175                               checked={tests.length === props.tests.length}
176                               color="primary"
177                               onClick={(event) => handleChangeTests(event, "Todos")}
178                            />
179                         </TableCell>
180                         <TableCell>Todos</TableCell>
181                      </TableRow> : ""
182                   }
183                   {
184                      openTests ?
185                         props.tests.map((t) => (
186                            <TableRow key={t._id}>
187                               <TableCell>
188                                  <Checkbox
189                                     checked={tests.includes(t._id)}
190                                     color="primary"
191                                     onClick={(event) => handleChangeTests(event, t._id)}
192                                  />
193                               </TableCell>
194                               <TableCell>{t.name}</TableCell>
195                            </TableRow>
196                         )) : ""
197                   }
198                </TableBody>
199             </Table>
200          </TableContainer>
201          <br/>
202          <TableContainer component={Paper} style={{maxHeight: 300, width: 350}}>
203             <Table size="small">
204                <TableHead>
205                   <TableRow onClick={() => setOpenPackages(!openPackages)} style={{cursor:
                       ↪ "pointer"}}>
206                      <TableCell>
207                         <IconButton >
```

```
208                          {openPackages ? <KeyboardArrowUpIcon fontSize="large" /> : <
                                 ↪ KeyboardArrowDownIcon fontSize="large" />}
209                       </IconButton>
210                    </TableCell>
211                    <TableCell>
212                       <h3>Pacotes a executar</h3>
213                    </TableCell>
214                 </TableRow>
215              </TableHead>
216              <TableBody>
217                 {
218                    openPackages ?
219                    <TableRow key={"Todos"}>
220                       <TableCell>
221                          <Checkbox
222                             checked={packages.length === props.packages.length}
223                             color="primary"
224                             onClick={(event) => handleChangePackages(event, "Todos")}
225                          />
226                       </TableCell>
227                       <TableCell>Todos</TableCell>
228                    </TableRow> : ""
229                 }
230                 {
231                    openPackages ?
232                       props.packages.map((t) => (
233                          <TableRow key={t._id}>
234                             <TableCell>
235                                <Checkbox
236                                   checked={packages.includes(t._id)}
237                                   color="primary"
238                                   onClick={(event) => handleChangePackages(event, t._id)}
239                                />
240                             </TableCell>
241                             <TableCell>{t.name}</TableCell>
242                          </TableRow>
243                       )) : ""
244                 }
245              </TableBody>
246           </Table>
247        </TableContainer>
248        {displayAlert}
249     </DialogContent>
250     <DialogActions>
251        <Button autoFocus onClick={handleClose} color="primary">
```

```
                Cancelar
            </Button>
            <Button autoFocus onClick={handleInsert} color="primary">
                Adicionar
            </Button>
         </DialogActions>
      </Dialog>
   );
}
```

Listing B.22: UpdateSchedule.js

```
import React, { useState } from "react";
import { makeStyles } from "@material-ui/core/styles";
import Button from "@material-ui/core/Button";
import Dialog from "@material-ui/core/Dialog";
import DialogActions from "@material-ui/core/DialogActions";
import DialogContent from "@material-ui/core/DialogContent";
import DialogTitle from "@material-ui/core/DialogTitle";
import useMediaQuery from "@material-ui/core/useMediaQuery";
import { useTheme } from "@material-ui/core/styles";
import Switch from "@material-ui/core/Switch";
import Checkbox from "@material-ui/core/Checkbox";
import TextField from "@material-ui/core/TextField";
import Alert from "@material-ui/lab/Alert";
import Table from '@material-ui/core/Table';
import TableBody from '@material-ui/core/TableBody';
import TableCell from '@material-ui/core/TableCell';
import TableContainer from '@material-ui/core/TableContainer';
import TableHead from '@material-ui/core/TableHead';
import TableRow from '@material-ui/core/TableRow';
import Paper from '@material-ui/core/Paper';
import IconButton from '@material-ui/core/IconButton';
import KeyboardArrowDownIcon from '@material-ui/icons/KeyboardArrowDown';
import KeyboardArrowUpIcon from '@material-ui/icons/KeyboardArrowUp';

const useStyles = makeStyles((theme) => ({
   form: {
      textAlign: "center",
   },
   formControl: {
      margin: theme.spacing(2),
      width: 300,
   },
}));

```

```
export default function UpdateSchedule(props) {
   const theme = useTheme();
   const classes = useStyles();
   const fullScreen = useMediaQuery(theme.breakpoints.down("sm"));
   const [hour, setHour] = useState(props.schedule.hour);
   const [active, setActive] = useState(props.schedule.active);
   const [tests, setTests] = useState(props.schedule.tests);
   const [alert, setAlert] = useState(false);
   const [packages, setPackages] = useState(props.schedule.packages);
   const [openTests, setOpenTests] = useState(false);
   const [openPackages, setOpenPackages] = useState(false);

   const handleChangeHour = (event) => {
      setHour(event.target.value);
   };

   const handleChangePackages = (event, value) => {
      if(value === "Todos"){
         if(props.packages.length === packages.length) setPackages([])
         else setPackages(props.packages.map(p => p._id))
      }
      else{
         packages.includes(value) === true
            ? setPackages(
               packages.filter((id) => {
                  if (id === value) return false;
                  else return true;
               })
            )
            : setPackages(packages.concat(value));
      }
   };

   const handleChangeTests = (event, value) => {
      if(value === "Todos"){
         if(props.tests.length === tests.length) setTests([])
         else setTests(props.tests.map(t => t._id))
      }
      else{
         tests.includes(value) === true
            ? setTests(
               tests.filter((id) => {
                  if (id === value) return false;
                  else return true;
               })
```

```
            )
            : setTests(tests.concat(value));
      }
   };

   const handleClose = () => {
      props.setUpdate(false);
   };

   const handleUpdate = async () => {
      if (hour === "" || (tests.length === 0 && packages.length === 0)) {
         setAlert(true);
      } else {
         let newSchedule = {
            hour: hour,
            active: active,
            tests: tests,
            packages: packages
         };
         props.handleUpdate(props.schedule._id, newSchedule)
      }
   };

   let displayAlert =
      alert === false ? (
         ""
      ) : (
         <div>
            <Alert
               severity="error"
               variant="filled"
               onClose={() => {
                  setAlert(false);
               }}
            >
               Tem que preencher os campos todos!
            </Alert>
            <br />
         </div>
      );

   return (
      <Dialog
         disableBackdropClick
         fullScreen={fullScreen}
```

```
125         open={props.update}
126         onClose={handleClose}
127       >
128         <DialogTitle>Atualizar Agendamento</DialogTitle>
129         <DialogContent>
130           <div className={classes.form}>
131             <TextField
132               label="Hora"
133               type="time"
134               name="hour"
135               value={hour}
136               onChange={handleChangeHour}
137               inputProps={{ step: 900 }}
138             />
139             <Switch
140               color="primary"
141               onChange={() => setActive((prev) => !prev)}
142               checked={active}
143             />
144           </div>
145           <br/>
146           <TableContainer component={Paper} style={{maxHeight: 400, width: 350}}>
147             <Table size="small">
148               <TableHead>
149                 <TableRow onClick={() => setOpenTests(!openTests)} style={{cursor: "
                      ↪ pointer"}}>
150                   <TableCell>
151                     <IconButton >
152                       {openTests ? <KeyboardArrowUpIcon fontSize="large" /> : <
                            ↪ KeyboardArrowDownIcon fontSize="large" />}
153                     </IconButton>
154                   </TableCell>
155                   <TableCell>
156                     <h3>Testes a executar</h3>
157                   </TableCell>
158                 </TableRow>
159               </TableHead>
160               <TableBody>
161                 {
162                   openTests ?
163                   <TableRow key={"Todos"}>
164                     <TableCell>
165                       <Checkbox
166                         checked={tests.length === props.tests.length}
167                         color="primary"
```

```
168                                 onClick={(event) => handleChangeTests(event, "Todos")}
169                                 />
170                              </TableCell>
171                              <TableCell>Todos</TableCell>
172                           </TableRow> : ""
173                        }
174                        {
175                           openTests ?
176                              props.tests.map((t) => (
177                                 <TableRow key={t._id}>
178                                    <TableCell>
179                                       <Checkbox
180                                          checked={tests.includes(t._id)}
181                                          color="primary"
182                                          onClick={(event) => handleChangeTests(event, t._id)}
183                                       />
184                                    </TableCell>
185                                    <TableCell>{t.name}</TableCell>
186                                 </TableRow>
187                              )) : ""
188                        }
189                     </TableBody>
190                  </Table>
191               </TableContainer>
192               <br/>
193               <TableContainer component={Paper} style={{maxHeight: 300, width: 350}}>
194                  <Table size="small">
195                     <TableHead>
196                        <TableRow onClick={() => setOpenPackages(!openPackages)} style={{cursor:
                              ↪ "pointer"}}>
197                           <TableCell>
198                              <IconButton >
199                                 {openPackages ? <KeyboardArrowUpIcon fontSize="large" /> : <
                                       ↪ KeyboardArrowDownIcon fontSize="large" />}
200                              </IconButton>
201                           </TableCell>
202                           <TableCell>
203                              <h3>Pacotes a executar</h3>
204                           </TableCell>
205                        </TableRow>
206                     </TableHead>
207                     <TableBody>
208                        {
209                           openPackages ?
210                           <TableRow key={"Todos"}>
```

```
                        <TableCell>
                            <Checkbox
                                checked={packages.length === props.packages.length}
                                color="primary"
                                onClick={(event) => handleChangePackages(event, "Todos")}
                            />
                        </TableCell>
                        <TableCell>Todos</TableCell>
                    </TableRow> : ""
                }
                {
                    openPackages ?
                        props.packages.map((t) => (
                            <TableRow key={t._id}>
                                <TableCell>
                                    <Checkbox
                                        checked={packages.includes(t._id)}
                                        color="primary"
                                        onClick={(event) => handleChangePackages(event, t._id)}
                                    />
                                </TableCell>
                                <TableCell>{t.name}</TableCell>
                            </TableRow>
                        )) : ""
                }
            </TableBody>
        </Table>
    </TableContainer>
    {displayAlert}
</DialogContent>
<DialogActions>
    <Button autoFocus onClick={handleClose} color="primary">
        Cancelar
    </Button>
    <Button autoFocus onClick={handleUpdate} color="primary">
        Atualizar
    </Button>
</DialogActions>
</Dialog>
);
}
```

Listing B.23: Packages.js

```
import React, { useEffect, useState } from "react";
import SplitPane from "react-split-pane";
```

```
3  import { getPackages, getTests, deletePackage } from "../../../api/api";
4  import ListItem from "@material-ui/core/ListItem";
5  import ListItemText from "@material-ui/core/ListItemText";
6  import { List } from "@material-ui/core";
7  import Button from "@material-ui/core/Button";
8  import EditPackage from "./EditPackage";
9  import NewPackage from "./NewPackage";
10 import ResponsiveButton from "./ResponsiveButton";
11 import ErrorIcon from '@material-ui/icons/Error';
12 import Tooltip from '@material-ui/core/Tooltip';
13 import { Typography } from "@material-ui/core";
14
15 export default function Packages() {
16    const [loading, setLoading] = useState(false);
17    const [tests, setTests] = useState([]);
18    const [packages, setPackages] = useState([]);
19    const [selectedIndex, setSelectedIndex] = useState("");
20    const [edit, setEdit] = useState(false);
21    const [open, setOpen] = useState(false);
22
23    useEffect(() => {
24       setLoading(true);
25       async function fetchData() {
26          let t = await getTests();
27          setTests(t);
28          let p = await getPackages();
29          setPackages(p);
30          setLoading(false);
31       }
32       fetchData();
33    }, []);
34
35    const handleCancelEdition = () => {
36       setEdit(false);
37       setSelectedIndex("");
38    };
39
40    const handleSelect = (id) => {
41       if (edit) {
42          setEdit(false);
43          setSelectedIndex(id);
44       } else {
45          setSelectedIndex(id);
46       }
47    };
```

```
48
49    const handleEdit = () => {
50       setEdit(true);
51    };
52
53    const handleDelete = async () => {
54       setOpen(false);
55       setEdit(false);
56       await deletePackage(selectedIndex);
57       let p = await getPackages();
58       setSelectedIndex("");
59       setPackages(p);
60    };
61
62    const handleRefresh = async () => {
63       setEdit(false);
64       let p = await getPackages();
65       setPackages(p);
66    };
67
68    return (
69       <div>
70          {loading ? (
71             <h3 style={{ textAlign: "center" }}>A carregar dados...</h3>
72          ) : (
73             <SplitPane split="vertical" defaultSize={"30%"}>
74                <div
75                   style={{
76                      textAlign: "center",
77                   }}
78                >
79                   <h2>Lista de Pacotes</h2>
80                   <List
81                      style={{
82                         marginLeft: "15%",
83                         marginRight: "15%",
84                         overflow: "auto",
85                         maxHeight: 600,
86                      }}
87                   >
88                      {packages.map((p) => {
89                         if(!p.active){
90                            return (
91                               <ListItem
92                                  button
```

```
93                                 key={p._id}
94                                 selected={selectedIndex === p._id}
95                                 onClick={() => handleSelect(p._id)}
96                              >
97                                 <Tooltip placement="top-start" title={<Typography>O pacote
                                       ↪ precisa de ser atualizado devido a atualizações nos
                                       ↪ testes primitivos.</Typography>}>
98                                    <ListItemText primary={p.name} />
99                                 </Tooltip>
100                                <ErrorIcon />
101                             </ListItem>
102                          )
103                       }
104                       else{
105                          return (
106                             <ListItem
107                                button
108                                key={p._id}
109                                selected={selectedIndex === p._id}
110                                onClick={() => handleSelect(p._id)}
111                             >
112                                <ListItemText primary={p.name} />
113                             </ListItem>
114                          )
115                       }
116                    })}
117             </List>
118             <br />
119             <div>
120                <Button
121                   style={{ marginRight: 10 }}
122                   disabled={!(selectedIndex.length > 0)}
123                   variant="outlined"
124                   color="primary"
125                   onClick={() => setOpen(true)}
126                >
127                   Remover
128                </Button>
129                <Button
130                   style={{ marginLeft: 10 }}
131                   disabled={!(selectedIndex.length > 0)}
132                   variant="outlined"
133                   color="primary"
134                   onClick={handleEdit}
135                >
```

```
                        Editar
                     </Button>
                  </div>
                  <ResponsiveButton
                     selectedIndex={selectedIndex}
                     open={open}
                     setOpen={setOpen}
                     handleDelete={handleDelete}
                  />
               </div>
               <div style={{ textAlign: "center" }}>
                  {!edit ? (
                     <NewPackage
                        handleRefresh={handleRefresh}
                        tests={tests}
                        packages={packages}
                     />
                  ) : (
                     <EditPackage
                        handleRefresh={handleRefresh}
                        tests={tests}
                        packages={packages}
                        package={
                           packages.filter(
                              (p) => p._id === selectedIndex
                           )[0]
                        }
                        handleCancelEdition={handleCancelEdition}
                     />
                  )}
               </div>
            </SplitPane>
         )}
      </div>
   );
}
```

Listing B.24: Keywords.js

```
import React from "react";
import { makeStyles } from "@material-ui/core/styles";
import ListItem from "@material-ui/core/ListItem";
import ListItemText from "@material-ui/core/ListItemText";
import List from "@material-ui/core/List";
import AddCircleIcon from "@material-ui/icons/AddCircle";
import Tooltip from '@material-ui/core/Tooltip';
```

```
import { Typography } from "@material-ui/core";

const useStyles = makeStyles((theme) => ({
   root: {
      overflow: "auto",
      width: 350,
      maxHeight: 500,
      marginLeft: "25%",
      backgroundColor: theme.palette.background.paper,
   },
}));

export default function Keywords(props) {
   const classes = useStyles();

   return (
      <div>
         <h3>Lista de Testes</h3>
         <List className={classes.root}>
            {props.tests.map((test) => (
               <ListItem button={true} onClick={() => props.add(test.name)} key={test._id}>
                  <Tooltip placement="top-start" title={<Typography>{test.description}</
                      ↪ Typography>}>
                     <ListItemText primary={test.name} />
                  </Tooltip>
                     <AddCircleIcon color="primary" />
               </ListItem>
            ))}
         </List>
      </div>
   );
}
```

Listing B.25: NewPackage.js

```
import React, { useState } from "react";
import TextField from "@material-ui/core/TextField";
import SplitPane from "react-split-pane";
import Keywords from "./Keywords";
import Button from "@material-ui/core/Button";
import ButtonGroup from "@material-ui/core/ButtonGroup";
import Alert from "@material-ui/lab/Alert";
import Tooltip from '@material-ui/core/Tooltip';
import { Typography } from "@material-ui/core";
import { insertPackage } from "../../../api/api";

```

```
12
13 const conectores = [
14    {
15       simbol: "->",
16       tip: "NEXT"
17    },
18    {
19       simbol: "(",
20       tip: "BEGIN LOGICAL EXPRESSION"
21    },
22    {
23       simbol: "&",
24       tip: "AND"
25    },
26    {
27       simbol: "|",
28       tip: "OR"
29    },
30    {
31       simbol: ")",
32       tip: "END LOGICAL EXPRESSION"
33    },
34    {
35       simbol: "?",
36       tip: "IF"
37    },
38    {
39       simbol: ":",
40       tip: "ELSE"
41    },
42    {
43       simbol: ";",
44       tip: "END"
45    },
46 ]
47
48 export default function NewPackage(props) {
49    const [name, setName] = useState("");
50    const [description, setDescription] = useState("");
51    const [script, setScript] = useState("");
52    const [error, setError] = useState({active: false, description: ""});
53    const [nameRepetead, setNameRepetead] = useState(false);
54
55    const handleClickable = (simbol) => {
56       let reg = script.match(/->|\(|\)|&|\||\?|:|;|([A-Za-z]+([/ _-][A-Za-z]+)*)/g)
```

```
57      let regArray = script.length > 0 ? reg.map(r => r.match(/([A-Za-z]+([/ _-][A-Za-z]+)*)/g
            ↪ ) ? "keyword" : r) : []
58      let numOpen = regArray.filter(r => r === "(").length
59      let numClose = regArray.filter(r => r === ")").length
60      let numIf = regArray.filter(r => r === "?").length
61      let numElse = regArray.filter(r => r === ":").length
62      if(simbol === ";"){
63         if(regArray[regArray.length - 1] !== "keyword" || numOpen !== numClose || numClose
               ↪ !== numIf) return false
64      }
65      else if(simbol === "->"){
66         if(regArray[regArray.length - 1] !== "keyword" || numOpen !== numClose || numClose
               ↪ !== numIf || regArray.includes(";")) return false
67      }
68      else if(simbol === "("){
69         if(regArray[regArray.length - 1] === "keyword" || regArray[regArray.length - 1] === "
               ↪ ;" || numOpen !== numClose || numClose !== numIf || regArray.includes(";"))
               ↪ return false
70      }
71      else if(simbol === ")"){
72         if(regArray[regArray.length - 1] !== "keyword" || numOpen === numClose || regArray.
               ↪ includes(";")) return false
73      }
74      else if(simbol === "&"){
75         if(regArray[regArray.length - 1] !== "keyword" || numOpen === numClose || regArray[
               ↪ regArray.length - 2] === "|" || regArray.includes(";")) return false
76      }
77      else if(simbol === "|"){
78         if(regArray[regArray.length - 1] !== "keyword" || numOpen === numClose || regArray[
               ↪ regArray.length - 2] === "&" || regArray.includes(";")) return false
79      }
80      else if(simbol === "?"){
81         if(regArray[regArray.length - 1] !== ")" || regArray.includes(";")) return false
82      }
83      else if(simbol === ":"){
84         if(regArray[regArray.length - 1] !== "keyword" || numOpen !== numClose || numIf <=
               ↪ numElse || regArray.includes(";")) return false
85      }
86      return true
87   };
88
89   const handleChange = (event) => {
90      const { name, value } = event.target;
91      if (name === "name") setName(value);
92      if (name === "description") setDescription(value);
```

```
93        if (name === "script") setScript(value);
94     };
95
96     const handleClean = () => {
97        setName("");
98        setDescription("");
99        setScript("");
100    };
101
102    const handleAddKeyword = (name) => {
103       let newScript = script + " " + name + " ";
104       setScript(newScript);
105    };
106
107    const handleAddSimbol = (simbol) => {
108       if(handleClickable(simbol)){
109          let newScript = script + " " + simbol + " ";
110          setScript(newScript);
111       }
112    };
113
114    const handleCreate = async () => {
115       if(!props.packages.map(p => p.name).includes(name)){
116          let i = await insertPackage({
117             name: name,
118             description: description,
119             script: script,
120          });
121          if (i.errors) {
122             setError({active: true, description: i.errors});
123          } else {
124             setName("");
125             setDescription("");
126             setScript("");
127             props.handleRefresh();
128          }
129       }
130       else{
131          setNameRepetead(true)
132       }
133    };
134
135    return (
136       <SplitPane split="vertical" defaultSize={"50%"}>
137          <div>
```

```
138             <h3>Criar Novo Pacote</h3>
139             <TextField
140                label="Nome"
141                name="name"
142                variant="outlined"
143                value={name}
144                onChange={handleChange}
145             />
146             <br />
147             <br />
148             <TextField
149                label="Descrição"
150                name="description"
151                variant="outlined"
152                style={{ maxWidth: 500 }}
153                multiline
154                fullWidth={true}
155                rows={4}
156                value={description}
157                onChange={handleChange}
158             />
159             <br />
160             <br />
161             <TextField
162                style={{ maxWidth: 500 }}
163                label="Código"
164                name="script"
165                variant="outlined"
166                fullWidth={true}
167                multiline
168                rows={10}
169                value={script}
170                onChange={handleChange}
171             />
172             <p>
173                Conetores:
174                <ButtonGroup style={{ marginLeft: 10 }} color="primary">
175                   {conectores.map((c) => (
176                      <Tooltip title={<Typography>{c.tip}</Typography>}>
177                         <Button
178                            color={handleClickable(c.simbol) ? "primary" : "secondary"}
179                            onClick={() => handleAddSimbol(c.simbol)}
180                         >
181                            {c.simbol}
182                         </Button>
```

```
183                     </Tooltip>
184                  ))}
185               </ButtonGroup>
186            </p>
187            <Button
188               style={{ marginRight: 10 }}
189               disabled={name.length === 0 && description.length === 0 && script.length === 0}
190               variant="outlined"
191               color="primary"
192               onClick={handleClean}
193            >
194               Limpar
195            </Button>
196            <Button
197               style={{ marginLeft: 10 }}
198               disabled={!(name.length > 0 && description.length > 0 && script.length > 0)}
199               variant="outlined"
200               color="primary"
201               onClick={handleCreate}
202            >
203               Guardar
204            </Button>
205            {error.active ? (
206               <Alert
207                  style={{ marginTop: 10 }}
208                  severity="error"
209                  variant="filled"
210                  onClose={() => setError({active: false, description: ""})}
211               >
212                  O código não está correto!
213               </Alert>
214            ) : (
215               ""
216            )}
217            {nameRepetead ? (
218               <Alert
219                  style={{ marginTop: 10 }}
220                  severity="error"
221                  variant="filled"
222                  onClose={() => setNameRepetead(false)}
223               >
224                  Já existe um pacote com esse nome!
225               </Alert>
226            ) : (
227               ""
```

```
            )}
         </div>
         <div>
            <Keywords tests={props.tests} add={handleAddKeyword} />
         </div>
      </SplitPane>
   );
}
```

Listing B.26: EditPackage.js

```
import React, { useState } from "react";
import TextField from "@material-ui/core/TextField";
import SplitPane from "react-split-pane";
import Keywords from "./Keywords";
import Button from "@material-ui/core/Button";
import ButtonGroup from "@material-ui/core/ButtonGroup";
import Alert from "@material-ui/lab/Alert";
import Tooltip from '@material-ui/core/Tooltip';
import { Typography } from "@material-ui/core";
import { updatePackage } from "../../../api/api";

const conectores = [
   {
      simbol: "->",
      tip: "NEXT"
   },
   {
      simbol: "(",
      tip: "BEGIN LOGICAL EXPRESSION"
   },
   {
      simbol: "&",
      tip: "AND"
   },
   {
      simbol: "|",
      tip: "OR"
   },
   {
      simbol: ")",
      tip: "END LOGICAL EXPRESSION"
   },
   {
      simbol: "?",
      tip: "IF"
```

```
36     },
37     {
38        simbol: ":",
39        tip: "ELSE"
40     },
41     {
42        simbol: ";",
43        tip: "END"
44     },
45  ]
46
47  export default function EditPackage(props) {
48     const [name, setName] = useState(props.package.name);
49     const [description, setDescription] = useState(props.package.description);
50     const [script, setScript] = useState(props.package.code);
51     const [error, setError] = useState({active: false, description: ""});
52     const [nameRepetead, setNameRepetead] = useState(false);
53
54     const handleClickable = (simbol) => {
55        let reg = script.match(/->|\(|\)|&|\||\?|:|;|([A-Za-z]+([/ _-][A-Za-z]+)*)/g)
56        let regArray = script.length > 0 ? reg.map(r => r.match(/([A-Za-z]+([/ _-][A-Za-z]+)*)/g
              ↪ ) ? "keyword" : r) : []
57        let numOpen = regArray.filter(r => r === "(").length
58        let numClose = regArray.filter(r => r === ")").length
59        let numIf = regArray.filter(r => r === "?").length
60        let numElse = regArray.filter(r => r === ":").length
61        if(simbol === ";"){
62           if(regArray[regArray.length - 1] !== "keyword" || numOpen !== numClose || numClose
                 ↪ !== numIf) return false
63        }
64        else if(simbol === "->"){
65           if(regArray[regArray.length - 1] !== "keyword" || numOpen !== numClose || numClose
                 ↪ !== numIf || regArray.includes(";")) return false
66        }
67        else if(simbol === "("){
68           if(regArray[regArray.length - 1] === "keyword" || regArray[regArray.length - 1] === "
                 ↪ ;" || numOpen !== numClose || numClose !== numIf || regArray.includes(";"))
                 ↪ return false
69        }
70        else if(simbol === ")"){
71           if(regArray[regArray.length - 1] !== "keyword" || numOpen === numClose || regArray.
                 ↪ includes(";")) return false
72        }
73        else if(simbol === "&"){
```

```
74          if(regArray[regArray.length - 1] !== "keyword" || numOpen === numClose || regArray[
                ↪ regArray.length - 2] === "|" || regArray.includes(";")) return false
75       }
76       else if(simbol === "|"){
77          if(regArray[regArray.length - 1] !== "keyword" || numOpen === numClose || regArray[
                ↪ regArray.length - 2] === "&" || regArray.includes(";")) return false
78       }
79       else if(simbol === "?"){
80          if(regArray[regArray.length - 1] !== ")" || regArray.includes(";")) return false
81       }
82       else if(simbol === ":"){
83          if(regArray[regArray.length - 1] !== "keyword" || numOpen !== numClose || numIf <=
                ↪ numElse || regArray.includes(";")) return false
84       }
85       return true
86    };
87
88    const handleChange = (event) => {
89       const { name, value } = event.target;
90       if (name === "name") setName(value);
91       if (name === "description") setDescription(value);
92       if (name === "script") setScript(value);
93    };
94
95    const handleAddKeyword = (name) => {
96       let newScript = script + " " + name + " ";
97       setScript(newScript);
98    };
99
100   const handleAddSimbol = (simbol) => {
101      let newScript = script + " " + simbol + " ";
102      setScript(newScript);
103   };
104
105   const handleUpdate = async () => {
106      if(!props.packages.filter(p => p._id !== props.package._id).map(p => p.name).includes(
           ↪ name)){
107         if(props.package.name !== name || props.package.description !== description || props.
              ↪ package.script !== script){
108            let i = await updatePackage(props.package._id, {
109               name: name,
110               description: description,
111               script: script,
112            });
113            if (i.errors) {
```

```
114                setError({active: true, description: i.errors});
115             } else {
116                props.handleRefresh();
117             }
118          }
119          else {
120             props.handleRefresh();
121          }
122       }
123       else{
124          setNameRepetead(true)
125       }
126    };
127
128    return (
129       <SplitPane split="vertical" defaultSize={"50%"}>
130          <div>
131             <h3>Editar Pacote</h3>
132             <TextField
133                label="Nome"
134                name="name"
135                variant="outlined"
136                value={name}
137                onChange={handleChange}
138             />
139             <br />
140             <br />
141             <TextField
142                label="Descrição"
143                name="description"
144                variant="outlined"
145                style={{ maxWidth: 500 }}
146                multiline
147                fullWidth={true}
148                rows={4}
149                value={description}
150                onChange={handleChange}
151             />
152             <br />
153             <br />
154             <TextField
155                style={{ maxWidth: 500 }}
156                label="Código"
157                name="script"
158                variant="outlined"
```

```
159                fullWidth={true}
160                multiline
161                rows={10}
162                value={script}
163                onChange={handleChange}
164             />
165             <p>
166                Conetores:
167                <ButtonGroup style={{ marginLeft: 10 }} color="primary">
168                   {conectores.map((c) => (
169                      <Tooltip title={<Typography>{c.tip}</Typography>}>
170                         <Button
171                            color={handleClickable(c.simbol) ? "primary" : "secondary"}
172                            onClick={() => handleAddSimbol(c.simbol)}
173                         >
174                            {c.simbol}
175                         </Button>
176                      </Tooltip>
177                   ))}
178                </ButtonGroup>
179             </p>
180             <Button
181                style={{ marginRight: 10 }}
182                variant="outlined"
183                color="primary"
184                onClick={props.handleCancelEdition}
185             >
186                Cancelar
187             </Button>
188             <Button
189                style={{ marginLeft: 10 }}
190                disabled={!(name.length > 0 && description.length > 0 && script.length > 0)}
191                variant="outlined"
192                color="primary"
193                onClick={handleUpdate}
194             >
195                Guardar
196             </Button>
197             {error.active ? (
198                <Alert
199                   style={{ marginTop: 10 }}
200                   severity="error"
201                   variant="filled"
202                   onClose={() => setError(false)}
203                >
```

```
                  O código não está correto!
               </Alert>
            ) : (
               ""
            )}
            {nameRepetead ? (
               <Alert
                  style={{ marginTop: 10 }}
                  severity="error"
                  variant="filled"
                  onClose={() => setNameRepetead(false)}
               >
                  Já existe um pacote com esse nome!
               </Alert>
            ) : (
               ""
            )}
         </div>
         <div>
            <Keywords tests={props.tests} add={handleAddKeyword} />
         </div>
      </SplitPane>
   );
}
```

Listing B.27: AddPackageButton.js

```
import React from "react";
import Button from "@material-ui/core/Button";
import Dialog from "@material-ui/core/Dialog";
import DialogActions from "@material-ui/core/DialogActions";
import DialogTitle from "@material-ui/core/DialogTitle";
import useMediaQuery from "@material-ui/core/useMediaQuery";
import { useTheme } from "@material-ui/core/styles";

export default function ResponsiveDialog(props) {
   const theme = useTheme();
   const fullScreen = useMediaQuery(theme.breakpoints.down("sm"));

   return (
      <Dialog
         disableBackdropClick={true}
         fullScreen={fullScreen}
         open={props.open}
      >
         <DialogTitle>
```

```
            {"Tem a certeza que pretende remover o pacote?"}
         </DialogTitle>
         <DialogActions>
            <Button
               onClick={() => props.setOpen(false)}
               color="primary"
               autoFocus
            >
               Não
            </Button>
            <Button onClick={props.handleDelete} color="primary" autoFocus>
               Sim
            </Button>
         </DialogActions>
      </Dialog>
   );
}
```

Listing B.28: DocumentationTests.js

```
import React, { useEffect, useState } from "react";
import SimpleTable from "./SimpleTable";
import { getTests } from "../../../api/api";

export default function Docs() {
   const [loading, setLoading] = useState(false);
   const [tests, setTests] = useState([]);

   useEffect(() => {
      setLoading(true);
      async function fetchData() {
         let t = await getTests();
         setTests(t);
         setLoading(false);
      }
      fetchData();
   }, []);

   return (
      <div>
         {loading ? (
            <h3 style={{ textAlign: "center" }}>A carregar dados...</h3>
         ) : (
               <SimpleTable tests={tests} />
         )}
      </div>
```

```
27     );
28 }
```

Listing B.29: DocumentationTable.js

```
1  import React from "react";
2  import Table from "@material-ui/core/Table";
3  import TableBody from "@material-ui/core/TableBody";
4  import TableCell from "@material-ui/core/TableCell";
5  import TableContainer from "@material-ui/core/TableContainer";
6  import TableHead from "@material-ui/core/TableHead";
7  import TableRow from "@material-ui/core/TableRow";
8  import Paper from "@material-ui/core/Paper";
9
10 export default function SimpleTable(props) {
11
12    return (
13       <TableContainer
14          style={{ margin: "1%", width: "98%", maxHeight: 820 }}
15          component={Paper}
16       >
17          <Table>
18             <TableHead>
19                <TableRow>
20                   <TableCell>
21                      <b>ID Teste</b>
22                   </TableCell>
23                   <TableCell align="center">
24                      <b>Módulo</b>
25                   </TableCell>
26                   <TableCell align="center">
27                      <b>Teste</b>
28                   </TableCell>
29                   <TableCell align="center">
30                      <b>Descrição</b>
31                   </TableCell>
32                   <TableCell align="center">
33                      <b>Parâmetro Por Defeito</b>
34                   </TableCell>
35                </TableRow>
36             </TableHead>
37             <TableBody>
38                {props.tests.map((test) => (
39                   <TableRow key={test._id}>
40                      <TableCell>{test.id}</TableCell>
41                      <TableCell align="center">{test.module}</TableCell>
```

```
                    <TableCell align="center">{test.name}</TableCell>
                    <TableCell align="center">
                       {test.description}
                    </TableCell>
                    <TableCell align="center">{test.defaultParam}</TableCell>
                 </TableRow>
              ))}
           </TableBody>
        </Table>
     </TableContainer>
  );
}
```

Listing B.30: SystemConfigurations.js

```
import React, { useEffect, useState } from "react";
import Grid from '@material-ui/core/Grid';
import TextField from "@material-ui/core/TextField";
import Button from "@material-ui/core/Button";
import { getConfigurations, getBackups, updateConfigurations, makeBackup, restoreBackup,
    ↪ getReports } from "../../../api/api";
import ExportCSV from "./ExportCSV"
import MakeBackup from "./MakeBackup"
import RestoreBackup from "./RestoreBackup"

export default function Configurations() {
   const [loading, setLoading] = useState(false);
   const [backups, setBackups] = useState([]);
   const [reports, setReports] = useState([]);
   const [configurations, setConfigurations] = useState({});
   const [userConfigurationID, setUserConfigurationID] = useState("");
   const [dbBinDir, setDbBinDir] = useState("");
   const [appDir, setAppDir] = useState("");
   const [backupSelected, setBackupSelected] = useState("");
   const [backupDir, setBackupDir] = useState("");
   const [backupDone, setBackupDone] = useState(false);
   const [restoreDone, setRestoreDone] = useState(false);
   const [unvalidPath, setUnvalidPath] = useState(false);
   const [noDiretories, setNoDiretories] = useState(false);
   const [dirNotFound, setDirNotFound] = useState(false);

   useEffect(() => {
      setLoading(true);
      async function fetchData() {
         let c = await getConfigurations();
         let b = await getBackups();
```

```
31          let r = await getReports()
32          setReports(r)
33          setUserConfigurationID(c.userConfigurationID)
34          setDbBinDir(c.dbBinDir)
35          setAppDir(c.appDir)
36          setBackupDir(c.backupDir)
37          setConfigurations(c)
38          setBackups(b);
39          setLoading(false);
40       }
41       fetchData();
42    }, []);
43
44    const handleChange = (event) => {
45       const { name, value } = event.target;
46       if (name === "userConfigurationID") setUserConfigurationID(value);
47       if (name === "dbBinDir") setDbBinDir(value);
48       if (name === "appDir") setAppDir(value);
49       if (name === "backupDir") setBackupDir(value);
50       if (name === "backupSelected") setBackupSelected(value);
51    };
52
53    const handleRestore = async () => {
54       if(appDir && dbBinDir && backupDir && backupSelected){
55          setLoading(true);
56          await restoreBackup(backupSelected)
57          let c = await getConfigurations();
58          let b = await getBackups();
59          setUserConfigurationID(c.userConfigurationID)
60          setDbBinDir(c.dbBinDir)
61          setAppDir(c.appDir)
62          setBackupDir(c.backupDir)
63          setConfigurations(c)
64          setBackups(b);
65          setBackupSelected("")
66          setLoading(false);
67          setRestoreDone(true)
68       }
69       else{
70          setUnvalidPath(true)
71       }
72    };
73
74    const handleBackup = async (options) => {
75       if(appDir && dbBinDir && backupDir){
```

```
76          setLoading(true);
77          let res = await makeBackup(options);
78          if(res === -1){
79             setDirNotFound(true)
80          }
81          else{
82             let b = await getBackups();
83             setBackups(b);
84             setBackupDone(true)
85          }
86          setLoading(false);
87       }
88       else{
89          setNoDiretories(true)
90       }
91    };
92
93    const handleGuardarConfiguracoes = async () => {
94       setLoading(true);
95       await updateConfigurations(configurations._id, {
96          userConfigurationID: userConfigurationID,
97          dbBinDir: dbBinDir,
98          backupDir: backupDir,
99          appDir: appDir
100      })
101      let c = await getConfigurations();
102      let b = await getBackups();
103      setUserConfigurationID(c.userConfigurationID)
104      setDbBinDir(c.dbBinDir)
105      setAppDir(c.appDir)
106      setBackupDir(c.backupDir)
107      setConfigurations(c)
108      setBackups(b);
109      setLoading(false);
110   };
111
112   const handleCancelar = () => {
113      setUserConfigurationID(configurations.userConfigurationID)
114      setDbBinDir(configurations.dbBinDir)
115      setAppDir(configurations.appDir)
116      setBackupDir(configurations.backupDir)
117   };
118
119   return (
120      <div style={{ textAlign: "center" }}>
```

```
121         {loading ? (
122            <h3>A carregar dados...</h3>
123         ) : (
124            <Grid container spacing={3}>
125               <Grid item xs={6}>
126                  <h2>Configurações do Sistema:</h2>
127                  <br />
128                  <br />
129                  <TextField
130                     style={{ width: 500 }}
131                     label="User Configuration ID"
132                     name="userConfigurationID"
133                     variant="outlined"
134                     value={userConfigurationID}
135                     onChange={handleChange}
136                  />
137                  <br />
138                  <br />
139                  <br />
140                  <TextField
141                     style={{ width: 500 }}
142                     label="MongoDB bin Directory"
143                     name="dbBinDir"
144                     variant="outlined"
145                     value={dbBinDir}
146                     onChange={handleChange}
147                  />
148                  <br />
149                  <br />
150                  <br />
151                  <TextField
152                     style={{ width: 500 }}
153                     label="Application Directory"
154                     name="appDir"
155                     variant="outlined"
156                     value={appDir}
157                     onChange={handleChange}
158                  />
159                  <br />
160                  <br />
161                  <br />
162                  <TextField
163                     style={{ width: 500 }}
164                     label="Backups Directory"
165                     name="backupDir"
```

```
166                     variant="outlined"
167                     value={backupDir}
168                     onChange={handleChange}
169                  />
170                  <br />
171                  <br />
172                  <br />
173                  <Button
174                     style={{marginRight: 10}}
175                     disabled={(configurations.backupDir === backupDir && configurations.
                           ↪ userConfigurationID === userConfigurationID && configurations.
                           ↪ dbBinDir === dbBinDir && configurations.appDir === appDir)}
176                     variant="outlined"
177                     color="primary"
178                     onClick={handleCancelar}
179                  >
180                     Cancelar
181                  </Button>
182                  <Button
183                     style={{marginLeft: 10}}
184                     disabled={(configurations.backupDir === backupDir && configurations.
                           ↪ userConfigurationID === userConfigurationID && configurations.
                           ↪ dbBinDir === dbBinDir && configurations.appDir === appDir)}
185                     variant="outlined"
186                     color="primary"
187                     onClick={handleGuardarConfiguracoes}
188                  >
189                     Guardar
190                  </Button>
191               </Grid>
192               <Grid item xs={6}>
193                  <br />
194                  <br />
195                  <ExportCSV reports={reports} />
196                  <br />
197                  <br />
198                  <MakeBackup
199                     handleBackup={handleBackup}
200                     noDiretories={noDiretories}
201                     setNoDiretories={setNoDiretories}
202                     backupDone={backupDone}
203                     setBackupDone={setBackupDone}
204                     dirNotFound={dirNotFound}
205                     setDirNotFound={setDirNotFound}
206                  />
```

```
                  <br />
                  <br />
                  <RestoreBackup
                     appDir={appDir}
                     dbBinDir={dbBinDir}
                     backupDir={backupDir}
                     backupSelected={backupSelected}
                     restoreDone={restoreDone}
                     setRestoreDone={setRestoreDone}
                     unvalidPath={unvalidPath}
                     setUnvalidPath={setUnvalidPath}
                     backups={backups}
                     handleChange={handleChange}
                     handleRestore={handleRestore}
                  />
               </Grid>
            </Grid>
         )}
      </div>
   );
}
```

Listing B.31: ExportCSV.js

```
import React, { useState } from "react";
import Paper from '@material-ui/core/Paper';
import Button from "@material-ui/core/Button";
import { CSVLink } from "react-csv";
import Grid from '@material-ui/core/Grid';
import DateFnsUtils from '@date-io/date-fns';
import {
 MuiPickersUtilsProvider,
 KeyboardDatePicker,
} from '@material-ui/pickers';

export default function ExportCSV(props) {
   const [reports, setReports] = useState(props.reports);
   const [reportsToExport, setReportsToExport] = useState(props.reports);
   const [initialDate, setInitialDate] = useState(null);
   const [finalDate, setFinalDate] = useState(null);

   const headers = [
      { label: "User ID", key: "id_user" },
      { label: "Date", key: "date" },
      { label: "Module", key: "module" },
      { label: "Name", key: "name" },
```

```
23      { label: "Message", key: "message" },
24      { label: "Execution Time", key: "runtime" },
25      { label: "Result Value", key: "resultValue" },
26      { label: "Result", key: "result" }
27    ]
28
29    const getFormatedDate = (tempDate, i) => {
30      let tmpDay = tempDate.getDate() + ""
31      let day = tmpDay.length === 1 ? "0" + tmpDay : tmpDay;
32      let tmpMonth = tempDate.getMonth() + 1 + "";
33      let month = tmpMonth.length === 1 ? "0" + tmpMonth : tmpMonth;
34      let year = tempDate.getFullYear() + "";
35      let date = "" + year + "" + month + "" + day
36      let tempHour = tempDate.getHours() + "";
37      let tempMin = tempDate.getMinutes() + "";
38      let tempSec = tempDate.getSeconds() + "";
39      let hour = tempHour.length === 1 ? "0" + tempHour : tempHour;
40      let min = tempMin.length === 1 ? "0" + tempMin : tempMin;
41      let sec = tempSec.length === 1 ? "0" + tempSec : tempSec;
42      i === 0 ? date += "000000" : (i === 1 ? date += "235959" : date += hour + min + sec)
43      return date;
44    };
45
46    const getDate = () => {
47      let tempDate = new Date();
48      let tmpDay = tempDate.getDate() + "";
49      let day = tmpDay.length === 1 ? "0" + tmpDay : tmpDay;
50      let tmpMonth = tempDate.getMonth() + 1 + "";
51      let month = tmpMonth.length === 1 ? "0" + tmpMonth : tmpMonth;
52      let year = tempDate.getFullYear() + "";
53      let tempHour = tempDate.getHours() + "";
54      let tempMin = tempDate.getMinutes() + "";
55      let tempSec = tempDate.getSeconds() + "";
56      let hour = tempHour.length === 1 ? "0" + tempHour : tempHour;
57      let min = tempMin.length === 1 ? "0" + tempMin : tempMin;
58      let sec = tempSec.length === 1 ? "0" + tempSec : tempSec;
59      let date = "" + year + "" + month + "" + day + "" + hour + "" + min + "" + sec;
60      return date;
61    };
62
63    const handleClick = (event, done) => {
64      let reportsAux = []
65      reports.filter(rep => {
66        if(initialDate !== null && finalDate !== null){
67          let begin = getFormatedDate(initialDate, 0)
```

```
68              let end = getFormatedDate(finalDate, 1)
69              let tempDate = new Date(parseInt(rep.date.split(" ")[0].split("-")[0]), parseInt(
                    ↪ rep.date.split(" ")[0].split("-")[1]) - 1, parseInt(rep.date.split(" ")[0].
                    ↪ split("-")[2]), parseInt(rep.date.split(" ")[1].split(":")[0]), parseInt(
                    ↪ rep.date.split(" ")[1].split(":")[1]))
70              let current = getFormatedDate(tempDate, 2)
71              if(current >= begin && current <= end) return true
72           }
73           else if(initialDate !== null){
74              let begin = getFormatedDate(initialDate, 0)
75              let tempDate = new Date(parseInt(rep.date.split(" ")[0].split("-")[0]), parseInt(
                    ↪ rep.date.split(" ")[0].split("-")[1]) - 1, parseInt(rep.date.split(" ")[0].
                    ↪ split("-")[2]), parseInt(rep.date.split(" ")[1].split(":")[0]), parseInt(
                    ↪ rep.date.split(" ")[1].split(":")[1]))
76              let current = getFormatedDate(tempDate, 2)
77              if(current >= begin) return true
78           }
79           else if(finalDate !== null){
80              let end = getFormatedDate(finalDate, 1)
81              let tempDate = new Date(parseInt(rep.date.split(" ")[0].split("-")[0]), parseInt(
                    ↪ rep.date.split(" ")[0].split("-")[1]) - 1, parseInt(rep.date.split(" ")[0].
                    ↪ split("-")[2]), parseInt(rep.date.split(" ")[1].split(":")[0]), parseInt(
                    ↪ rep.date.split(" ")[1].split(":")[1]))
82              let current = getFormatedDate(tempDate, 2)
83              if(current <= end) return true
84           }
85           return false
86        }).map(rep => rep.results.map(test =>
87           reportsAux.push({
88              date: rep.date,
89              id_user: rep.id_user,
90              module: test.module,
91              name: test.name,
92              message: test.message,
93              runtime: test.runtime,
94              resultValue: test.resultValue,
95              result: test.result === "success" ? "Passou" : (test.result === "inconclusive" ? "
                    ↪ Inconclusivo" : "Falhou"),
96           })
97        ))
98        setReports(props.reports)
99        setReportsToExport(reportsAux)
100       setInitialDate(null)
101       setFinalDate(null)
102       done()
```

```
103     }
104
105     return (
106        <Paper style={{maxWidth: 700, backgroundColor: "#d6f0d6"}}>
107           <br/>
108           <h3>Exportar relatórios de execução no formato CSV:</h3>
109           <MuiPickersUtilsProvider utils={DateFnsUtils}>
110              <Grid container justify="space-around">
111                 <KeyboardDatePicker
112                    margin="normal"
113                    id="date-picker-initial"
114                    label="Data Inicial"
115                    format="yyyy/MM/dd"
116                    value={initialDate}
117                    onChange={setInitialDate}
118                 />
119                 <KeyboardDatePicker
120                    margin="normal"
121                    id="date-picker-final"
122                    label="Data Final"
123                    format="yyyy/MM/dd"
124                    value={finalDate}
125                    onChange={setFinalDate}
126                 />
127              </Grid>
128           </MuiPickersUtilsProvider>
129           <br />
130           <CSVLink
131              data={reportsToExport}
132              headers={headers}
133              asyncOnClick={true}
134              onClick={handleClick}
135              filename={"reports_v" + getDate() + ".csv"}
136              style={{textDecoration: "none"}}
137           >
138              <Button
139                 variant="outlined"
140                 color="primary"
141              >
142                 Download CSV
143              </Button>
144           </CSVLink>
145           <br />
146           <br />
147        </Paper>
```

```
148     )
149 }
```

Listing B.32: MakeBackup.js

```
1  import React, { useState } from "react";
2  import Button from "@material-ui/core/Button";
3  import Alert from "@material-ui/lab/Alert";
4  import Paper from '@material-ui/core/Paper';
5  import FormControl from '@material-ui/core/FormControl';
6  import FormControlLabel from '@material-ui/core/FormControlLabel';
7  import Checkbox from '@material-ui/core/Checkbox';
8
9  export default function MakeBackup(props) {
10    const [options, setOptions] = useState({
11       reports: true,
12       schedules: true,
13       packages: true,
14       configurations: true,
15    });
16
17    const handleChange = (event) => {
18       if([options.reports, options.schedules, options.packages, options.configurations].filter
          ↪ ((v) => v).length > 1 || !options[event.target.name])
19          setOptions({ ...options, [event.target.name]: event.target.checked });
20    };
21
22    const handleBackup = async () => {
23       props.handleBackup(options)
24    };
25
26    return (
27       <div>
28          <Paper style={{maxWidth: 700, backgroundColor: "#d6f0d6"}}>
29             <br />
30             <h3>Efetuar cópia de segurança ao sistema:</h3>
31             <p>(Escolher opções a incluir no backup, pelo menos tem que escolher uma)</p>
32             <FormControl>
33                <FormControlLabel
34                   control={<Checkbox color="primary" checked={options.reports} onChange={
                         ↪ handleChange} name="reports" />}
35                   label="Relatórios"
36                />
37                <FormControlLabel
38                   control={<Checkbox color="primary" checked={options.schedules} onChange={
                         ↪ handleChange} name="schedules" />}
```

```
39                label="Agendamentos"
40             />
41          </FormControl>
42          <FormControl>
43             <FormControlLabel
44                control={<Checkbox color="primary" checked={options.packages} onChange={
                       ↪ handleChange} name="packages" />}
45                label="Pacotes de Testes"
46             />
47             <FormControlLabel
48                control={<Checkbox color="primary" checked={options.configurations} onChange
                       ↪ ={handleChange} name="configurations" />}
49                label="Configurações"
50             />
51          </FormControl>
52          <br />
53          <br />
54          <Button
55             variant="outlined"
56             color="primary"
57             onClick={handleBackup}
58          >
59             Efetuar Backup
60          </Button>
61          <br />
62          <br />
63          {
64             props.noDiretories ?
65             <div>
66                <Alert
67                   style={{ margin: 10 }}
68                   severity="error"
69                   variant="filled"
70                   onClose={() => props.setNoDiretories(false)}
71                >
72                   Tem de preencher todas diretorias para efetuar uma cópia de segurança.
73                </Alert>
74                <br />
75             </div> : ""
76          }
77          {
78             props.dirNotFound ?
79             <div>
80                <Alert
81                   style={{ margin: 10 }}
```

```
                        severity="error"
                        variant="filled"
                        onClose={() => props.setDirNotFound(false)}
                    >
                        A diretoria de backups não foi encontrada.
                    </Alert>
                    <br />
                </div> : ""
            }
        </Paper>
        {
            props.backupDone ?
            <div>
                <br />
                <Alert
                    style={{ margin: 10, maxWidth: 650 }}
                    severity="success"
                    variant="filled"
                    onClose={() => props.setBackupDone(false)}
                >
                    A cópia de segurança foi efetuada com sucesso!
                </Alert>
            </div> : ""
        }
    </div>
  )
}
```

Listing B.33: RestoreBackup.js

```
import React from "react";
import Paper from '@material-ui/core/Paper';
import Button from "@material-ui/core/Button";
import Alert from "@material-ui/lab/Alert";
import InputLabel from '@material-ui/core/InputLabel';
import MenuItem from '@material-ui/core/MenuItem';
import FormControl from '@material-ui/core/FormControl';
import Select from '@material-ui/core/Select';

export default function RestoreBackup(props) {
  return (
    <div>
        <Paper style={{maxWidth: 700, backgroundColor: "#d6f0d6"}}>
            <br />
            <h3>Carregar backup e restabelecer cópia de segurança do sistema:</h3>
```

```
16            <p>(Insira o nome da diretoria correspondente àversão do backup que quer carregar)
                  ↪ </p>
17            <FormControl variant="outlined" style={{ width: 300, textAlign: "left" }}>
18               <InputLabel>Backup</InputLabel>
19               <Select
20                  value={props.backupSelected}
21                  name="backupSelected"
22                  onChange={props.handleChange}
23                  label="Backup"
24               >
25                  <MenuItem value="">Nenhum</MenuItem>
26                  {props.backups.map(b => <MenuItem value={b}>{b.split(".")[0]}</MenuItem>)}
27               </Select>
28            </FormControl>
29            <br />
30            <br />
31            <Button
32               variant="outlined"
33               color="primary"
34               onClick={props.handleRestore}
35            >
36               Carregar Backup
37            </Button>
38            <br />
39            <br />
40            {
41               props.unvalidPath ?
42               <div>
43                  <Alert
44                     style={{ margin: 10 }}
45                     severity="error"
46                     variant="filled"
47                     onClose={() => props.setUnvalidPath(false)}
48                  >
49                     Tem que selecionar um dos backups disponiveis! Se não existe nenhum pode
                           ↪ ser porque a diretoria especificada não contém nenhum backup ou
                           ↪ não estão preenchidas todas diretorias necessárias.
50                  </Alert>
51                  <br />
52               </div> : ""
53            }
54         </Paper>
55         {
56            props.restoreDone ?
57            <div>
```

```
58                <br />
59                <Alert
60                   style={{ margin: 10, maxWidth: 650 }}
61                   severity="success"
62                   variant="filled"
63                   onClose={() => props.setRestoreDone(false)}
64                >
65                   A cópia de segurança foi carregada com sucesso e o sistema foi atualizado!
66                </Alert>
67             </div> : ""
68          }
69       </div>
70    )
71 }
```