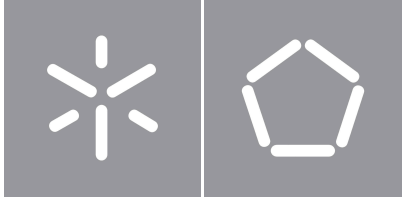
Universidade do Minho
Escola de Engenharia

Joel Salgueiro Martins

Aplicação FTTx BoM

outubro 2023



Universidade do Minho
Escola de Engenharia

Joel Salgueiro Martins

Aplicação FTTx BoM

Dissertação de Mestrado
Mestrado em Engenharia Informática

Trabalho efetuado sob a orientação de
António Costa
Helena Fernández

outubro 2023

Direitos de Autor e Condições de Utilização do Trabalho por Terceiros

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

Licença concedida aos utilizadores deste trabalho:

CC BY-SA

https://creativecommons.org/licenses/by-sa/4.0/

Declaração de Integridade

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

Universidade do Minho, Braga, outubro 2023

Joel Salgueiro Martins

Resumo

Atualmente com a massificação da Internet ao redor do mundo é inegável que a implantação das tecnologias de suporte à Internet para os seus consumidores tem de passar por um processo de planeamento antes da sua implantação, de modo a garantir máxima eficiência económica e máxima qualidade nos serviços de comunicação.

A tecnologia mais recente apta para a construção de novas redes fixas é a fibra ótica, capaz de transmitir dados recorrendo a sinais luminosos ao invés de sinais elétricos, garantindo elevadíssimas taxas de transferências de dados assim como confiabilidade e segurança acrescida.

O planeamento destas redes é de elevada importância para o sucesso do projeto de rede. Com este criam-se diversos resultados (um dos quais, a lista de materiais (**BoM**)) que permitem minimizar os custos envolvidos, tanto na implantação como na futura manutenção da rede, resultando em economias de dinheiro significativas a longo prazo.

Apesar das vantagens do planeamento destas redes, nas empresas, geralmente este é efetuado superficialmente, utilizando ferramentas genéricas que não automatizam a maioria das fases de planeamento e obrigam à intervenção manual que, devido à sua natureza minuciosa e ao erro humano, obriga a um prazo de execução prolongado que acaba por causar impactos negativos altamente significativos. Para mitigar este problema a Proef, empresa promotora desta dissertação, decidiu projetar quatro sistemas capazes de automatizar ao máximo cada uma das fases do planeamento, aspirando tornar este mais rápido e fiável.

A presente aplicação, *FTTx BoM*, incorpora-se na fase de construção da lista com todos os materiais necessários (**BoM**) e, visa automatizar a sua construção recorrendo apenas a informação dos documentos resultantes da fase de **Survey**, juntamente de regras de construção pré-definidas pela operadora financiadora que, tipicamente é um provedor de serviço de Internet (**ISP**).

O **BoM** desempenha um papel relevante na aprovação final por parte da operadora que financia o projeto e, atualmente, como a sua elaboração é feita manualmente, implica um período significativo frequentemente estendido devido a erros e inconsistências encontradas em fases avançadas, acarretando

prejuízos reputacionais e financeiros.

Palavras-chave Internet, fibra ótica, redes fixas, **BoM**, automatização.

Abstract

Nowadays, with the massification of the Internet around the world, it is undeniable that the implementation of Internet support technologies for consumers must go through a planning process prior to deployment, in order to guarantee maximum economic efficiency and maximum quality in communication services.

The latest technology suitable for building new fixed networks is optical fiber, which is capable of transmitting data using light signals rather than electrical signals, guaranteeing extremely high data transfer rates as well as increased reliability and security.

The planning of these networks is of great importance to the success of the network project. It creates a number of deliverables (one of which is the bill of materials (**BoM**)) that make it possible to minimize the costs involved, both in the deployment and future maintenance of the network, resulting in significant savings in the long term.

Despite the advantages of planning these networks, in companies it is generally carried out superficially, using generic tools that don't automate most of the planning phases and require manual intervention which, due to its meticulous nature and human error, results in a long lead time that ends up causing highly significant negative impacts. To mitigate this problem, Proef, the company behind this dissertation, decided to design four systems capable of automating each of the planning phases as much as possible, with the aim of making planning faster and more reliable.

The present application, *FTTx BoM*, is incorporated into the construction phase of the list with all the necessary materials (**BoM**) and aims to automate its construction using only information from the documents resulting from the **Survey** phase, together with construction rules predefined by the financing operator, which is typically an Internet service provider (**ISP**).

The **BoM** plays an important role in the final approval by the operator financing the project and, currently, as it is drawn up manually, it involves a significant period, often extended due to errors and inconsistencies found at later stages, leading to reputational and financial damage.

Key-words Internet, optical fiber, fixed networks, **BoM**, automation.

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

ACID Atomicity, Consistency, Isolation, and Durability. 31

API Application Programming Interface. 28–30, 35

ASCII American Standard Code for Information Interchange. 26

BoM Bill of Materials. iii–vi, viii, x, 4, 6–8, 11–17, 19, 20, 22, 24, 29, 33, 36–42, 44, 46, 47, 49–51, 61–63, 66, 71, 73

CAD Computer-aided design. 12, 13, 27, 28

CSV Comma Separated Values. 12, 13

DOM Document Object Model. 31

DWG Drawing Web Graphics. 25–29

DXF Drawing Exange Format. 25–29, 35, 40, 73

FAM Familiariedade. 35

FTTB Fiber to the Building. 3

FTTC Fiber to the Cabinet. 3

FTTH Fiber to the Home. 3

HTTP Hypertext Transfer Protocol. 31, 39, 48, 57

ISP Internet Service Provider. iii, v, 4

JSON JavaScript Object Notation. 32

JSX JavaScript XML. 31, 54

KMZ Keyhole Markup Language. 16

NoSQL Not Only Structured Query Language. 31–33, 59

PLC Programmable Logic Controller. 3, 9, 11, 43, 44, 50, 69

SQL Structured Query Language. 31

XLSX Microsoft Excel Open XML Spreadsheet. 12, 29, 35, 40, 43

XML Extensible Markup Language. 32

Glossário

Bottleneck Estado ou conjunto de estados que compromete o desempenho do sistema como um todo. 23–25

Domínio Conceito que abrange todos os atores e suas interações no processo ou área que se investiga. 5, 6, 19

FTTx Generaliza as diversas formas de implantação de Fibra Ótica até uma determinada infraestrutura (*Fiber to the X*). 3

Horizontal scaling Conceito que abrange o aumento na quantidade de máquinas a fornecerem um determinado serviço. Possui diversas formas de ser realizado. 24, 25, 35

Iot Generalização de dispositivos com capacidade de conexão à Internet (*Internet of Things*). 8

Mockup Modelo representativo que idealiza a interface projetada para fornecer funcionalidades estabelecidas por um conjunto de requisitos. 86

Schema Esquema tipicamente definido para uma base de dados que especifica, organiza e tipa os dados a serem guardados. 59

Survey Fase inicial do planeamento das redes de fibra ótica, essencial na identificação e levantamento de todas as informações existentes no terreno. Dela resultam documento como os mapas, sinóticos e tabelas de alocação da rede. iii, v, 10, 14, 15

Vertical scaling Conceito que abrange o aumento do poder computacional da própria máquina conforme a demanda por recursos, como processador, memória, armazenamento, largura de banda, etc. 23–25

xDSL Generaliza as diversas formas de implantação de Digital Subscriber Line. 2, 3

Zip Ficheiro de formato comum utilizado para comprimir um ou vários ficheiros numa só localização. 56

Parte I

Capítulo 1

Introdução

Este capitulo aborda a contextualização e motivação desta dissertação, assim como apresenta a sua empresa promotora, os objetivos pretendidos e a metodologia aplicada. Por último, apresenta a estruturação deste documento.

1.1 Contexto

Atualmente a Internet desempenha um papel fundamental na sociedade demonstrando importância desde o setor das telecomunicações até setores críticos como a saúde, a energia e as finanças [1] [2]. Desde o seu surgimento em Portugal que, ao longo dos anos, toda a sua infraestrutura tem passado por diversas expansões e evoluções. No começo, praticamente toda a infraestrutura se baseava na utilização de tecnologias da família *Digital Subscriber Line* (**xDSL**) [3], para assim fazer chegar a Internet aos seus utilizadores.

Com o passar dos anos, devido ao acréscimo constante de largura de banda necessitada pelos utilizadores, incapaz de ser acompanhada devido à reduzida largura de banda da tecnologia **xDSL** [4], levou os utilizadores a optarem pelas tecnologias de redes móveis que, efetivamente conseguiam aumentar a largura de banda substancialmente. No entanto, quando comparadas com as tecnologias **xDSL**, demonstravam-se instáveis, apresentando elevada latência, pelo que não foram consideradas tecnologias suficientemente capazes de substituir as tecnologias **xDSL**, impulsionando assim a investigação contínua por novas soluções [5].

Recentemente, surgiu a tecnologia fibra ótica, demonstrando-se capaz de oferecer elevadíssima largura de banda, altíssima estabilidade e reduzida latência, tendo sido na atualidade, definitivamente a melhor opção para substituir a tecnologia **xDSL** utilizada nas redes fixas [6]. Relativamente à Internet via rede móvel, com o surgimento da tecnologia 5G, apresentando características semelhantes à fibra ótica, continua a coexistir no mercado juntamente da fibra ótica [7].

Hoje, para fazer chegar a Internet a áreas sem qualquer infraestrutura já é utilizada a fibra ótica como tecnologia base que, para além de ser utilizada na expansão, também é utilizada na evolução das infraestruturas existentes [4] [8]. Na Figura 1 é representada a evolução da implantação da fibra ótica **FTTx** desde o seu surgimento, abstraindo os seus constituintes a: **provedor de Internet**, que tal como o nome indica, fornece a ligação à Internet; **juntas**, que representam pontos de divisão do sinal de Internet em vários outros; **PLC**'s, que representam os pontos imediatamente antes da ligação ao cliente; e **clientes**, que representa a habitação do utilizador final.

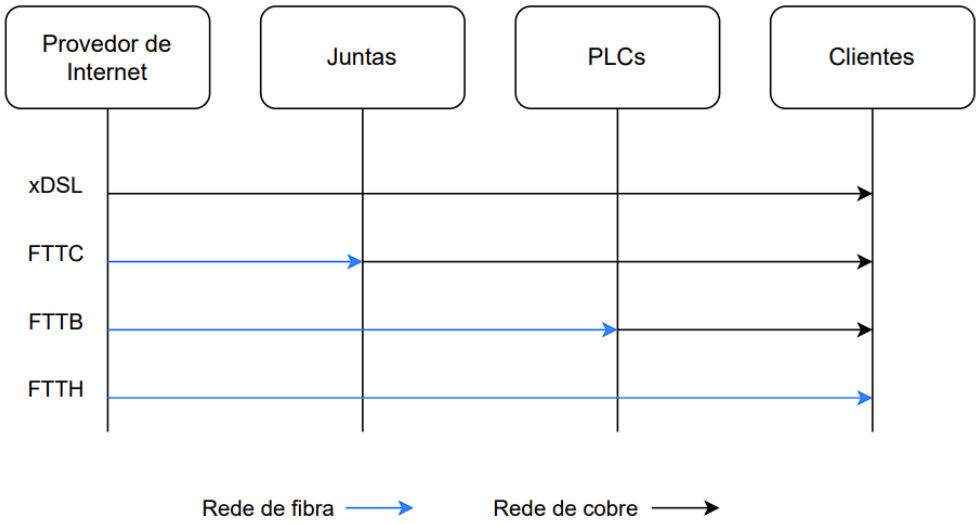


Figura 1: Evolução da infraestrutura xDSL em FTTx.

As primeiras redes fixas utilizavam tecnologias **xDSL** em toda a infraestrutura de suporte ao fornecimento de Internet, portanto, desde o Provedor até ao Cliente. Com o passar do tempo, esta infraestrutura foi sendo evoluída incrementalmente com a implantação de fibra ótica. Como primeira fase, *Fiber to the Cabinet* (**FTTC**), que implantava fibra ótica desde o Provedor até às Juntas, numa segunda fase, *Fiber to the Building* (**FTTB**), que implantava fibra ótica desde o Provedor até aos **PLC**'s, e numa terceira fase, a realidade hoje em dia, *Fiber to the Home* (**FTTH**), que implanta fibra ótica desde o Provedor até à casa do Cliente final [9].

1.2 Motivação

Sendo a fibra ótica uma tecnologia relativamente cara [10], antes do processo de implantação de uma rede utilizando-a, numa determinada área, é importante efetuar o processo de planeamento da rede [11], constituído pelo levantamento das infraestruturas existentes, determinação dos trajetos por onde e como

passar os cabos de fibra, além do cálculo do posicionamento dos componentes de rede, cabos e novas infraestruturas [6]. Consequentemente, é um processo demorado e requer várias etapas até à sua conclusão.

A aplicação resultante desta dissertação objetiva automatizar uma das fases constituintes do processo de planeamento, a criação da lista de materiais necessários, em inglês *Bill of Materias* (**BoM**).

A empresa promotora desta dissertação, a Proef, atua no âmbito da engenharia de redes fixas no setor das telecomunicações e fornece soluções de desenho e planeamento de redes para as diversas operadoras ou **ISP**'s. É precisamente para o planeamento destas redes que visa ser utilizada a aplicação resultante desta dissertação.

Atualmente, todas as fases do planeamento das redes fixas são feitas maioritariamente manualmente, pelo que, cada fase requer um intervalo de tempo significativamente demorado e, além disso, em cada fase existe uma alta probabilidade de ocorrerem erros que mais tarde acabam por ser detetados, criando problemas significativos que podiam ter sido evitados se o planeamento fosse altamente refinado, o que não acontece, pois implicaria um intervalo de templo ainda superior.

Devido às mesmas causas apresentadas, a Proef projetou a construção de quatro aplicações independentes (detalhadas no capítulo 2.4) que objetivam automatizar ao máximo as tarefas de planeamento.

1.3 Objetivos

Como mencionado anteriormente, com esta dissertação pretende-se uma aplicação capaz de automatizar o processo atual da construção do **BoM**, minimizando a intervenção humana. Para tal, a aplicação deverá através da informação contida nos documentos de projeto: sinótico, mapa e tabela de alocação da rede planeada, em conjunto com as regras de aplicação de materiais pré-definidas pela operadora promotora do projeto de planeamento, gerar a lista com todos os materiais necessários (**BoM**) para a sua implantação.

O **BoM** pretendido será constituído por todos os materiais necessários para a implantação da rede, associados às suas respetivas caraterísticas, fornecedoras e custos envolvidos, conforme o pretendido pela operadora promotora, além de estar conforme o *template* fornecido pela mesma.

Como resultados desta aplicação, é expectável um cálculo de orçamentos de implantação com fiabilidade acrescida, uma vez que não existirá intervenção humana, além da significativa redução de tempo exigido para a sua construção. Desta forma, com o tempo economizado, ainda durante a fase de planeamento poderão ser tomadas decisões para otimização de implantação impossíveis de serem tomadas sem a mesma.

1.4 Metodologia

Perante uma aplicação que visa automatizar um processo manual, é de elevada importância iniciar o seu desenvolvimento por uma fase de investigação e estudo do **Domínio** no qual o problema se incorpora, recorrendo a reuniões com as pessoas envolvidas no mesmo, de modo a perceber detalhadamente o problema e clarificar os conceitos envolvidos. Isto permitirá um levantamento de requisitos conciso.

Após o levantamento de requisitos é importante proceder à investigação e decisão das ferramentas de desenvolvimento adequadas de serem utilizadas, tendo em consideração o contexto de utilização atual e futuro, pretendido para a aplicação. Também é importante nesta fase, perceber a utilidade das soluções existentes no mercado.

Decididas as ferramentas e avaliadas as soluções existentes, é importante desenvolver uma arquitetura para a aplicação. Esta deverá ser constituída pela especificação de todos os seus componentes, interações entre os mesmos e funcionalidades a serem fornecidas ao utilizador. Também nesta fase, deverão ser desenvolvidos e validados os *mockups*.

Após desenvolvidas as etapas anteriores é importante definir as etapas de desenvolvimento, que consistem em agrupamentos específicos de funcionalidades que devem ser implementadas num intervalo agendado para a sua execução, sendo no final validadas em reuniões de acompanhamento efetuadas com as pessoas envolvidas no **Domínio**, de modo confirmar a boa execução da implementação e garantir a aplicação de uma metodologia ágil [12].

Para terminar, seguir-se-á a fase de testes, o lançamento da aplicação para ambiente de produção e a avaliação das funcionalidades desenvolvidas face aos requisitos inicialmente definidos. A Figura 2 ilustra a metodologia apresentada.

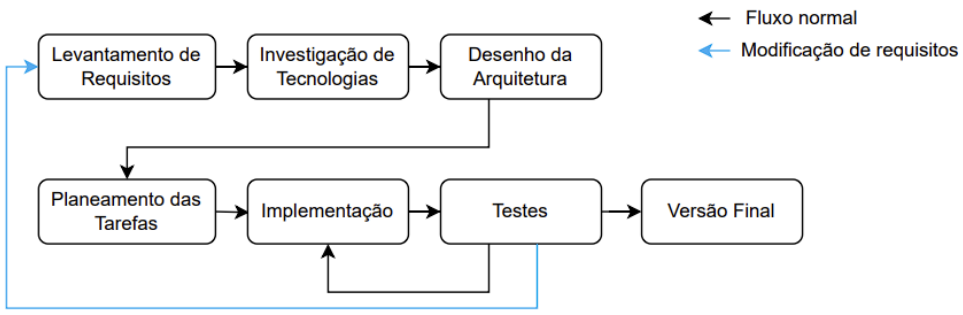


Figura 2: Metodologia.

Resumidamente, a metodologia a adotar consiste em: levantar os requisitos, investigar as tecnologias,

definir a arquitetura, planear as tarefas, implementar e testar. Entre a implementação e os testes existirão pelo menos tantas transições quanto o número de tarefas definidas, pois cada tarefa é constituída por um conjunto de funcionalidades que são implementadas, testadas e validadas antes de se prosseguir para uma nova tarefa. Após implementada a última tarefa, será revisto o levantamento de requisitos e verificado se tudo está conforme o pretendido, caso esteja, a aplicação passa para uma versão final, caso contrário, efetua-se novamente o ciclo de desenvolvimento.

1.5 Estrutura

A dissertação é constituída por sete capítulos, dos quais:

1. O atual contextualiza o domínio no qual se integra todo o propósito desta dissertação, assim como os motivos que levaram à sua necessidade e objetivos pretendidos.
2. O segundo apresenta a importância do **BoM**, o procedimento efetuado para a sua elaboração, as diversas soluções existentes no mercado para a resolução de problemas semelhantes e, por fim, justifica a necessidade do mesmo.
3. O terceiro apresenta os requisitos levantados do **Domínio**.
4. O quarto apresenta os problemas encontrados, detalha a arquitetura definida e as tecnologias selecionadas para implementação da aplicação.
5. O quinto apresenta detalhadamente a lógica da aplicação.
6. O sexto apresenta os testes efetuados e os resultados obtidos.
7. O sétimo aborda as conclusões e sugere um conjunto de tarefas para trabalho futuro.

Capítulo 2

Lista de Materiais (BoM), Soluções Disponíveis e Necessidade da Aplicação FTTx BoM

Ao longo deste capítulo será explicada a necessidade do **BoM** nos mercados globais, a sua contextualização nesta dissertação, assim como o seu processo de construção feito na Proef. Também serão apresentados os conteúdos relevantes do sinótico, mapa, tabela de alocação, regras e *templates* **BoM**. Posteriormente será feita a análise das aplicações e serviços existentes no mercado para a geração do **BoM** e para realização do planeamento de redes, terminando com a justificação da aplicação resultante desta dissertação e a apresentação das restantes aplicações projetadas para melhoramento de todo o ciclo de planeamento da Proef.

2.1 Lista de Materiais ou BoM

O conceito do **BoM** derivou da necessidade de organização das empresas, principalmente no setor da gestão de materiais e produção, por forma a conseguirem lidar com os mercados cada vez mais competitivos. O seu surgimento deu-se com a sistematização iniciada com a primeira revolução industrial e perante a cada vez mais globalização dos mercados, causando processos de produção cada vez mais complexos, hoje, perante quarta revolução, revela ainda mais importância [13] [14].

Atualmente, a elaboração do **BoM** é considerada uma tarefa de extrema importância para a concretização de qualquer finalidade, seja ela um produto ou uma infraestrutura. O **BoM** é composto pela especificação de todos os componentes, materiais, quantidades, fornecedores, respetivos custos unitários e totais de instalação ou construção [15].

Resumidamente, o **BoM** acompanha as seguintes vantagens:

1. Permite efetuar uma gestão eficiente dos materiais armazenados, mitigando armazenamentos excessivos ou em défice, além de permitir facilmente saber quais os materiais necessários para a

produção de determinada entidade. Também permite lidar com processos de produção complexos, auxiliando na coordenação de todos os constituintes da entidade a ser produzida. [16] [15].

2. Sendo a quarta revolução industrial caracterizada pela automação, **Iot** e análise de dados em tempo real, o **BoM** desempenha um papel fundamental quando integrado com sistema de gestão de produção e de planeamento.

3. Mitiga erros de produção, uma vez que é constituído por todos os materiais e seus respetivos detalhes, permitindo às entidades envolvidas nas linhas de produção atuarem com precisão.

4. Permite validar processos de produção, como a aprovação de orçamentos, a confirmação da existência de todos os materiais antes de se iniciar um processo de produção, e mesmo listar os materiais que já foram utilizados de modo a garantir que não existe omissão de nenhum.

Tipicamente, para as empresas de telecomunicações as principais vantagens com a utilização do **BoM** referem aos tópicos 1., 3. e 4. da lista anterior.

2.2 Atual Construção do BoM

Atualmente, a construção do **BoM** é realizada manualmente, revelando-se um processo demorado e suscetível a erros. Ao longo deste subcapítulo serão explicados os procedimentos e componentes dos projetos de planeamento de uma operadora específica, pois dependendo da operadora os documentos e organização do planeamento pode variar drasticamente.

O responsável aquando da sua construção procede à análise de três documentos, resultantes de fases prévias do planeamento. O primeiro documento, o **sinótico**, é constituído pela estrutura abstrata da rede de fibra ótica a ser construída, não possuindo informações relativamente à infraestrutura de suporte da mesma. Por outras palavras, contém informações da rede planeada abstratamente.

Os componentes de rede e os seus atributos que requerem análise no momento da elaboração do **BoM** são:

As **juntas**, responsáveis pela divisão do sinal recebido por várias fibras, possuem os seguintes atributos úteis:

- ID - Identifica a junta através da codificação de informações relativas à sub-área de suporte, sub-rede a que pertence e caraterísticas relativas ao próprio modelo.

- Tipo - Tipo do modelo.

- Tipo de instalação - Explicita a forma de instalação do equipamento, podendo ser em postes, fachadas ou condutas.

- Quantidade e tipo de *spliters* - Representa a quantidade de *spliters* necessários para as fusões (esta informação possuí mais detalhe na tabela de alocação que será explicada posteriormente).

Os **pontos de ligação ao cliente** (**PLC**'s) são semelhantes às juntas, contudo são colocados apenas em posições terminais cujas ligações que destes saem, seguem ou para outros **PLC**'s ou diretamente para habitações. Estes possuem os mesmos atributos relevantes que as juntas.

Os **cabos** são simbolizados pelas ligações entre os componentes acima apresentados, sendo constituídos pelos seguintes atributos:

- ID - Semelhante ao das juntas e **PLC**'s.

- Comprimento e capacidade - Representam a metragem necessária de cabo e, respetivamente, a quantidade de fibras óticas internas.

- Tipo de instalação - Tal como os componentes anteriores, a instalação pode ser feita em conduta, fachada ou poste (aéreo).

Visualmente, um sinótico apresenta as caraterísticas apresentadas na Figura 3.

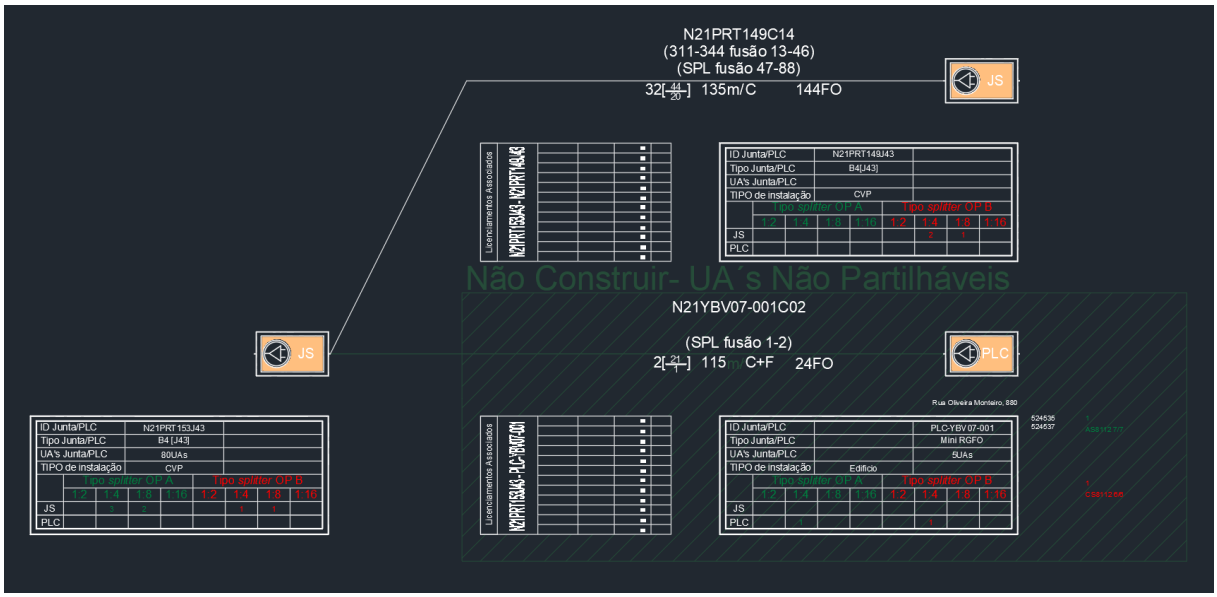


Figura 3: Sinótico.

Neste caso, existe uma junta raiz que dá suporte a outra junta e a um **PLC**. As ligações entre os componentes representam abstratamente os cabos de fibra ótica, não representando a infraestrutura de

suporte. Ainda sob estes existem apontamentos efetuados durante o **Survey**, juntamente das tabelas com os atributos apresentados anteriormente.

O segundo componente, o **mapa**, apesar de conter as informações do sinótico, acrescenta informações relativas à infraestrutura de suporte, mapeando a informação do sinótico no mapa cartográfico do terreno, acrescentando infraestruturas como: postes, tubos de subida, caixas de conduta e valas. Visualmente um mapa de rede possui as caraterísticas apresentadas na Figura 4.

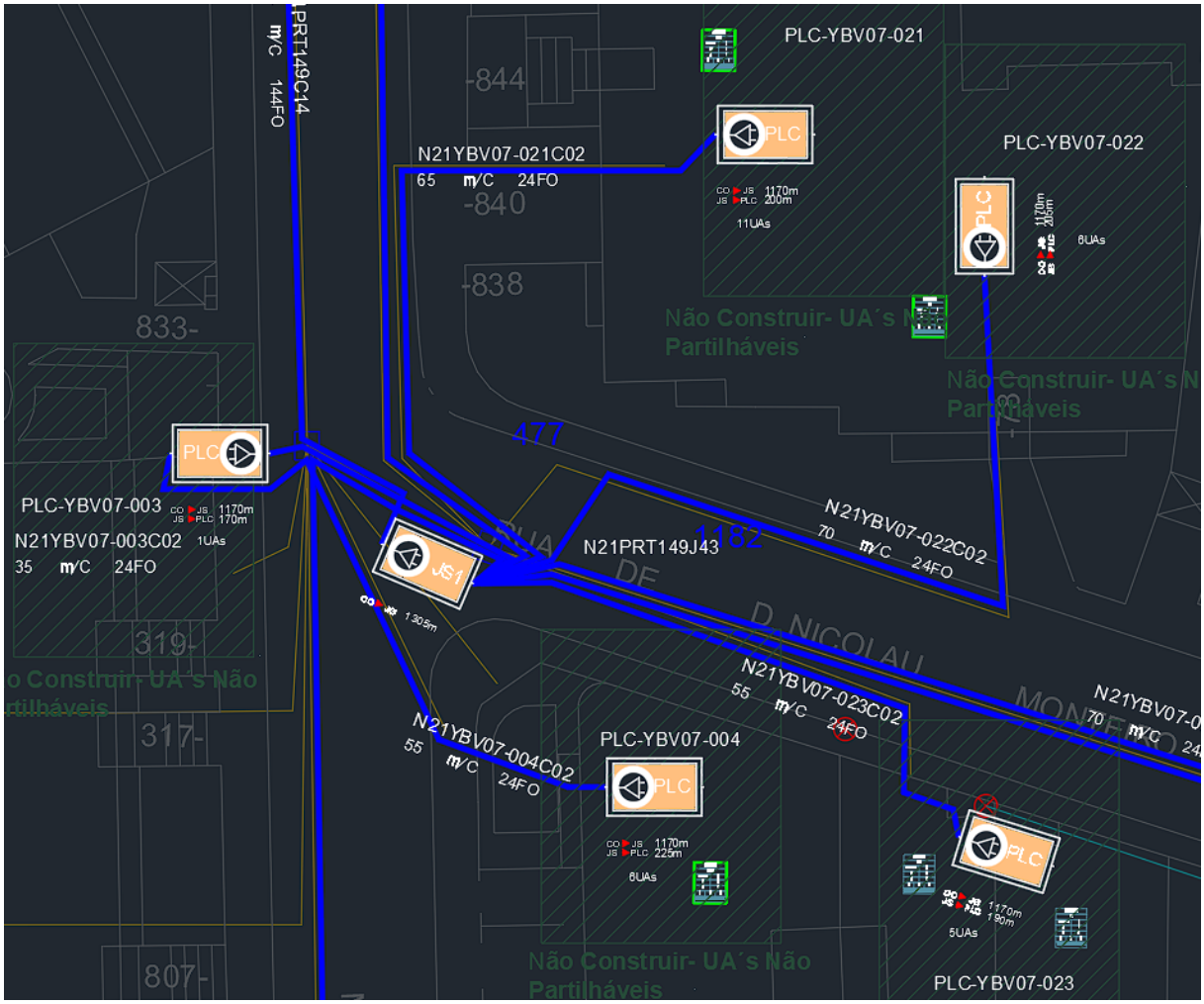


Figura 4: Mapa de rede.

Durante a análise do sinótico e do mapa de rede, é importante analisar os apontamentos representados sob a forma de uma grelha verde com um identificador "Não Construir - UA's Não Partilháveis". Este apontamento representa os componentes que já constam no terreno, cuja construção foi efetuada por outra operadora que autorizou a partilha do componente, pelo que não devem ser contabilizados para construção. Também é necessário verificar a rede a que os componentes pertencem, primária ou secundária, sendo esse processo efetuado pela verificação dos identificadores dos componentes. Por

último, é necessário verificar se os componentes da infraestrutura, no caso do mapa de rede, são todos necessários, uma vez que este contém toda a informação do terreno, seja ela relativa a componentes necessários ou não.

O terceiro componente, a **tabela de alocação**, possui informações relativas às tarefas que devem ser feitas nas juntas e/ou **PLC**'s que fazem parte da rede. Essas tarefas são tipicamente fusões entre cabos que devem ser contabilizadas no **BoM** conforme as suas caraterísticas de fusão e aplicação. Visualmente, a tabela de alocação apresenta as caraterísticas apresentadas na Figura 5.

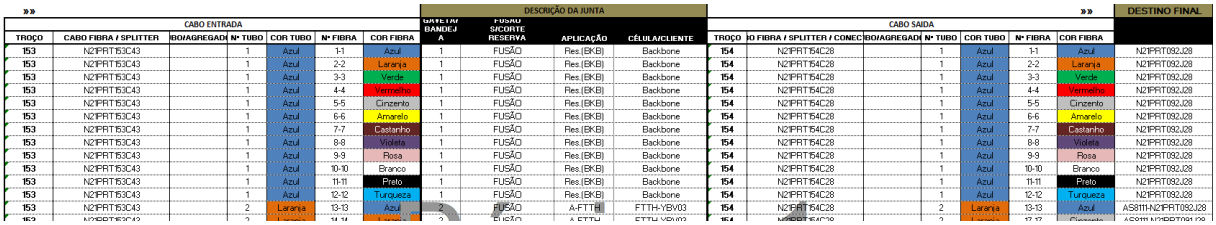


Figura 5: Tabela de alocação.

Para além da análise dos três constituintes do projeto de planeamento, o responsável tem de, recorrendo à sua própria experiência na elaboração de **BoM**'s, considerar as regras estabelecidas pela operadora, pois, à data, não existem documentos com a especificação das regras a serem aplicadas. Uma regra, tipicamente, é constituída por um material que deve ser aplicado sobre um componente, infraestrutura ou simplesmente que deve ser aplicado, por defeito, quando ocorrem determinadas situações. Estas regras podem diferir de operadora para operadora.

Como resultado obtém-se o *template* **BoM** preenchido com as devidas quantidades de material necessário para o projeto de planeamento em questão. A Figura 6 apresenta uma amostra do *template* em questão com as respetivas quantidades de material, representadas na coluna "#Q BOM".

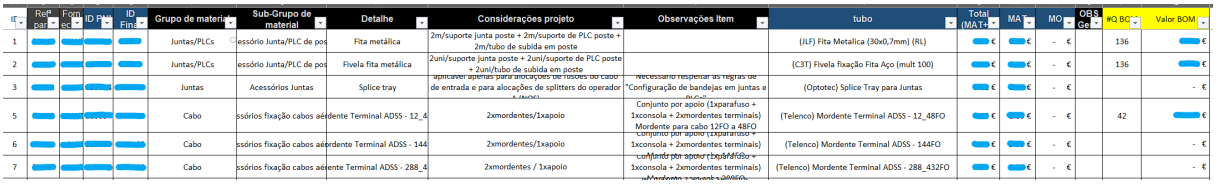


Figura 6: *Template* **BoM**.

2.3 Análise das Soluções Disponíveis

O universo das aplicações no mercado desenvolvidas com o intuito de apoiar a geração do **BoM**, para projetos de planeamento de redes de fibra ótica é consideravelmente reduzido. Aplicações independentes pensadas para a geração do **BoM** para projetos de redes, à data, não se encontram no mercado, no entanto, já existem aplicações capazes de extrair o **BoM** de projetos desenvolvidos em determinadas plataformas, como, por exemplo: *softwares* de desenho assistido por computador ou *computer aided design* (**CAD**).

Na Proef é utilizado o *software* de desenho **CAD** *AutoCad*, desenvolvido pela *AutoDesk*, para realizar o desenho dos sinóticos e mapas, logo, é importante tê-lo em consideração no momento da investigação das aplicações e serviços capazes da obtenção do **BoM** apresentadas seguidamente.

1. A aplicação *BoM List Generator*, desenvolvida pela *Jing Software*, consegue em associação exclusiva ao *AutoCad* gerar o **BoM** de um projeto desenvolvido internamente ao mesmo. Permite selecionar quais os atributos e propriedades dos blocos estáticos (entidades contentoras de informação no *AutoCad*) que se pretendem extrair. Também fornece suporte para propriedades dinâmicas (possivelmente a inserção de regras), no entanto, não deixa clara a definição das mesmas, pelo que não é possível avaliar a sua relevância para o projeto. Permite também selecionar vários projetos e unir toda a informação num único **BoM** gerado em formato **CSV**. É uma aplicação paga e não fornece nenhum tipo de versão experimental [17].

2. A aplicação *BoM Tools Pro*, desenvolvida pela *Micro Graphics*, é semelhante à anterior, acrescentando a possibilidade de exportar o **BoM** em formato *Microsoft Excel Open XML Spreadsheet* (**XLSX**), além de, segundo a própria desenvolvedora, possuir uma *interface* intuitiva capaz de possibilitar a verificação do **BoM** antes da sua exportação. É uma aplicação paga que fornece apenas uma versão de avaliação durante um período definido [18].

3. O serviço *BOM Management*, fornecido pela *OEMsecrets*, permite a criação do **BoM** manualmente, num ambiente *Web*, recorrendo a *templates* pré-definidos, permitindo a sua posterior exportação e importação. Apenas oferece a funcionalidade de construção do **BoM** manual, descartando a sua automatização, podendo ser útil para a substituição do *software* utilizado atualmente, o *Exell*. O seu uso não tem custos associados [19].

4. O serviço *IndaBOM*, desenvolvido por *Mike Kasparian*, é semelhante ao anterior, apenas acrescenta a vantagem do seu código ser *open-source*, pelo que, poderia ser considerada a sua modificação

por forma a automatizar a construção do **BoM** [20].

5. O serviço *OpenBoM*, desenvolvido e fornecido pela própria empresa *OpenBoM* oferece uma extensa gama de funcionalidades, além de possibilitar a sua integração com diversos *softwares* de desenho **CAD**. Apesar da extensa gama de funcionalidades, a de maior interesse para este projeto é a gestão do **BoM**, que possibilita a submissão de projetos realizados em **CAD** e o manuseamento dos seus materiais e componentes por meio de uma *interface* semelhante à do *Excel*. Além disso, como é um ambiente *Web*, tem a vantagem da fácil partilha de dados e colaborações com outros interessados. Contudo, é um serviço pago que fornece apenas uma versão de avaliação durante um período definido [21].

A informação apresentada encontra-se sintetizada na seguinte Tabela 1.

Tabela 1: Tecnologias relativas à geração do **BoM**.

Aplicação/Serviço	Ambiente de execução	Características destacadas
BoM List Generator	Desktop, exclusivo AutoDesk	Extração dos atributos dos blocos Customização de propriedades dinâmicas Geração do **BoM** em **CSV**
BoM Tools Pro	Desktop, exclusivo AutoDesk	Extração dos atributos dos blocos *Interface* intuitiva
BOM Manegement	Web	Ambiente colaborativo Fornecimento de *templates* **BoM**
IndaBOM	Web	Integra o Google Drive *Open Source*
OpenBOM	Web	Lida com softwares **CAD** Ambiente colaborativo

Apesar de não existirem aplicações para geração automática do **BoM** pensadas para o contexto das redes de fibra ótica, todas as investigadas ou gerariam o **BoM** incompleto ou obrigariam a alterar os

softwares já utilizados na Proef, sem oferecer uma garantia da capacidade de gerar o **BoM** completo e correto.

Apesar das soluções focadas na geração do **BoM** ainda estarem muito aquém do que se pretende, já existem no mercado serviços pensados para auxiliar o planeamento das redes de fibra ótica, capazes de realizar todo o processo, desde o **Survey** da área, até à obtenção do **BoM**. No entanto, são serviços pagos e pouco modulares, portanto, por exemplo: não permitem manipular documentos relativos ao resultado do **Survey**; não geram documentos de projeto como: mapa, sinótico e tabela de alocação, sendo documentos requeridos pela Proef para uso externo ao planeamento; não possibilitam a geração do **BoM** conforme as regras que se pretendem definir.

A seguir, tal como anteriormente, efetua-se a análise das aplicações e serviços existentes capazes de realizar todo o processo de planeamento.

1. O serviço *Weezie Fiber*, desenvolvido pela empresa *Weezie*, permite realizar o processo de **Survey**, possibilitando o registo das infraestruturas existentes, projeção da rede e geração de relatórios de inventário com os materiais e componentes necessários, similares ao **BoM** pretendido. Apesar de conseguir gerar relatórios de inventários, estes são muito básicos comparativamente ao que a Proef pretende, pelo que, seria um serviço relativamente útil apenas na fase do *survey*. É uma aplicação paga que também fornece uma versão de avaliação durante um período definido [22].

2. O serviço *Network Manager Telecom*, desenvolvido pela empresa *IQGEO* indica a possibilidade de realizar o processo de **Survey**, no entanto, não especifica a possibilidade de gerar **BoM**'s, permitindo apenas a utilização de serviços de terceiros, devendo ser possível a incorporação de outros serviços, como o primeiro, por exemplo [23].

3. O serviço *Vetro Fiber Map*, desenvolvido pela empresa *VETRO*, tal como a primeira, consegue realizar o processo de **Survey** e obter o inventário de materiais necessários, semelhante ao **BoM** pretendido, sendo este consideravelmente completo e possivelmente interessante para a Proef, no entanto, não especifica a possibilidade de criação de regras [24].

4. O serviço *Geospatial Network Inventory*, desenvolvido pela empresa *KSAVI GNI* não especifica a capacidade de realizar o processo de **Survey** na sua totalidade, apenas referencia o fornecimento alguns serviços de apoio ao planeamento, como automação de elementos de inventário e geração de documentação. Já para o **BoM** específica a capacidade da sua obtenção, no entanto, não deixa claro o conteúdo presente no mesmo [25].

5. O serviço *Comsof Fiber*, desenvolvido pela empresa *COMSOF*, também não especifica a capacidade de realizar o **Survey**, no entanto, menciona a capacidade de recorrendo a regras de desenho pré-definidas (exemplo: número de fibras e condutas subterrâneas que devem ser instaladas) calcular o **BoM** para toda a topologia da rede, podendo ser interessante para a Proef, no entanto, todas as funcionalidades estarão sempre limitadas às suportadas pelo serviço, além de não ser gratuito [26] [27].

A informação apresentada encontra-se sintetizada na seguinte Tabela 2.

Tabela 2: Tecnologias relativas ao planeamento de redes.

Aplicação/Serviço	Ambiente de execução	Características destacadas
Weezie Fiber	Web	Efetuação do **Survey** Geração de inventários *Interface* intuitiva
Network Manager Telecom	Desktop e Web	Integração com ferramentas de terceiros *Interface* intuitiva Geração de documentação financeira
Vetro Fiber Map	Web	Efetuação do **Survey** Geração de inventários
Geospatial Network Inventory	Web	Geração do **BoM** Extensão de funcionalidades via *plugins* Possível realização do **Survey**
Comsof Fiber	Web	Geração de **BoM** Possível criação de regras

2.4 Justificativa da aplicação FTTx BoM

Analisadas as aplicações e serviços existentes no mercado atual, justifica-se a aplicação resultante desta dissertação pela necessidade de um gerador do **BoM** automatizado, com foco no planeamento de redes. Deverá disponibilizar todas as funcionalidades requeridas tipicamente pelas empresas que atuam

no setor das telecomunicações, neste caso a Proef, como, por exemplo: a capacidade de customização de regras para a aplicação de materiais, a geração automática do **BoM** conforme o *template* pretendido pela operadora; tudo isto, recorrendo apenas aos documentos do projeto de planeamento.

Como explicado no capítulo anterior, esta aplicação pertence a um conjunto de quatro aplicações independentes que visam automatizar todo o processo de planeamento, de forma modular e independente. Cada uma consumirá e produzirá um resultado que poderá ser manipulado antes de seguir a linha de planeamento típica de um projeto de rede de fibra ótica. Este tipo de manipulação é impossível de se fazer nas aplicações e serviços existentes no mercado, e é essencial para a agilidade do projeto devido às constantes mudanças no próprio. As funcionalidades das quatro aplicações serão apresentadas a seguir.

A **FTTx Survey** é uma aplicação *mobile* pensada para utilização no terreno, aquando do levantamento das infraestruturas existentes, produzindo como resultado um ficheiro em formato **KMZ** onde constarão apontados todos os levantamentos efetuados. Recorrendo a este ficheiro, os responsáveis pelo planeamento construirão manualmente o mapa de rede, recorrendo à aplicação *AutoCad*, já apresentada. A construção do mapa é impossível de ser automatizada devido ao elevado número de dependências de terceiros relativamente às infraestruturas. Na Figura 7 ilustra-se o processo.

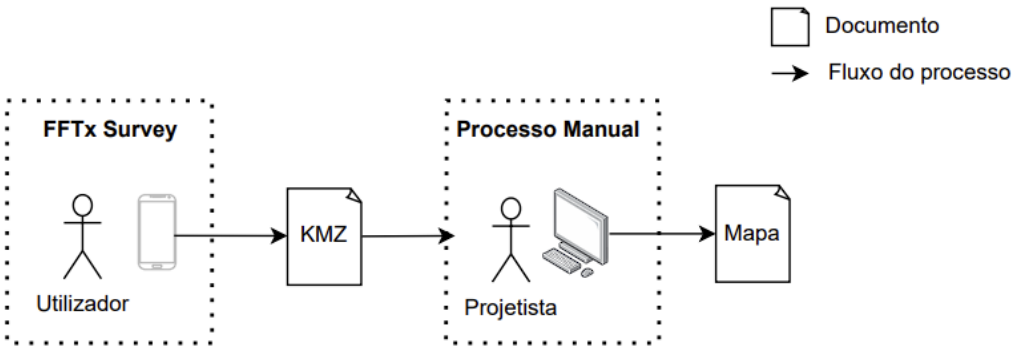


Figura 7: Processo de *survey*.

O mapa de rede resultante é em seguida utilizado pela aplicação **Fiber Networks Synoptics** que produzirá como resultado o sinótico e a tabela de alocação correspondentes, como ilustrado na Figura 8.

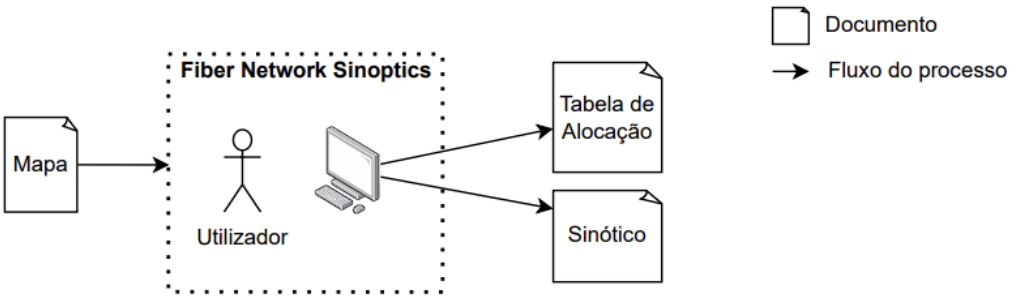


Figura 8: Processo de geração da tabela de alocação e sinótico.

A seguir, a aplicação resultante desta dissertação, **FTTx BoM**, recorre aos ficheiros resultantes, sinótico, mapa e tabela de alocação para gerar o respetivo **BoM** do planeamento de rede, como ilustrado na Figura 9.

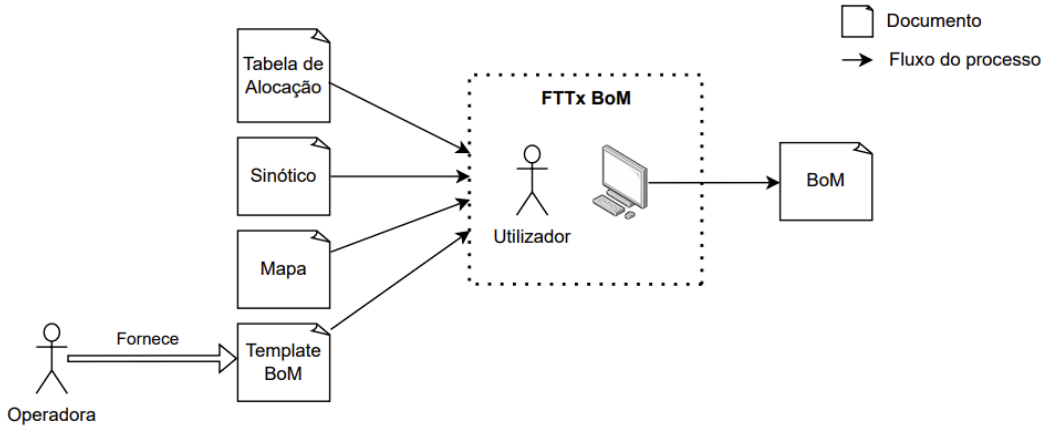


Figura 9: Processo de geração do BoM.

Por último, a aplicação **FTTx Check And Approval** recorre a todos os documentos resultantes do planeamento para, em conjunto com *Machine Learning*, identificar eventuais inconsistências, propor modificações e validar a sua execução. Ilustra-se na Figura 10.

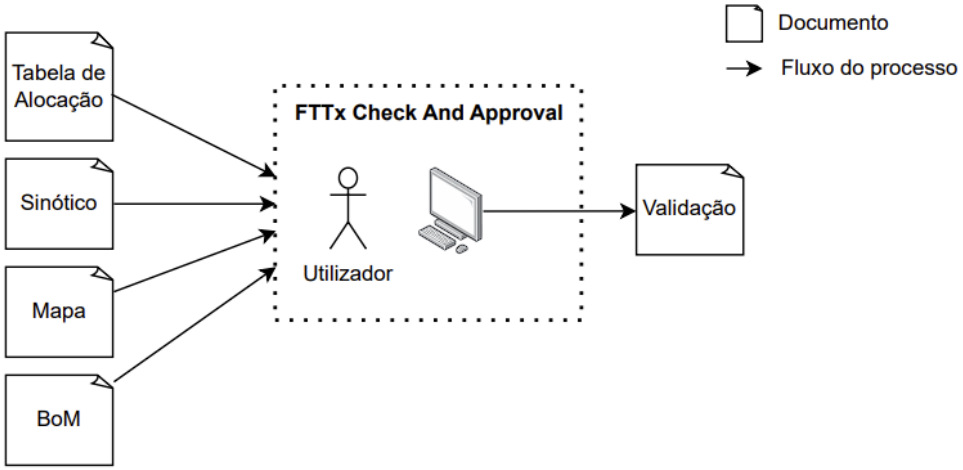


Figura 10: Processo de validação.

Capítulo 3

Requisitos

Ao longo deste capitulo apresentam-se todos os requisitos levantados do **Domínio** no qual o problema se insere, para a aplicação a ser desenvolvida [28]. Contudo, todas as informações necessárias relativas às funcionalidades e características a serem disponibilizadas serão resumidas ao longo do capítulo. Informações detalhadas encontram-se anexadas em apêndice.

3.1 Requisitos Funcionais

Os requisitos funcionais de uma aplicação especificam as funcionalidades a serem fornecidas ao utilizador final, assim como as respostas esperadas destas, além de desempenharem um papel fundamental na concretização da fase de planeamento. Os requisitos levantados agrupam-se conforme o seu relacionamento tal como na Figura 11, onde também estão apresentados os dois privilégios para utilização da aplicação, o **admin**, com acesso a todas as funcionalidades, e o **regular**, com acesso apenas às funcionalidades de autenticação e geração do **BoM**.

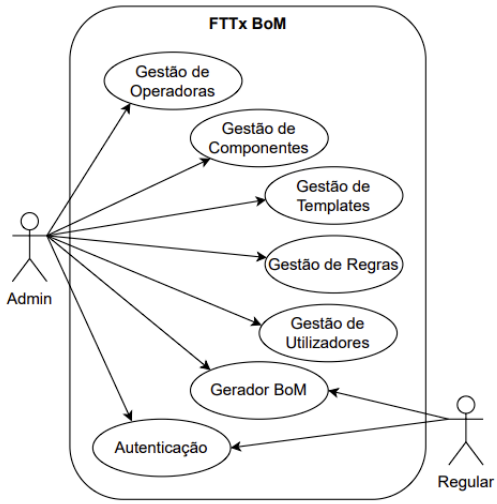


Figura 11: Agrupamentos de requisitos funcionais.

Os grupos caracterizam-se como:

- Gestão de Operadoras - Conjunto de funcionalidades relacionadas à gestão das operadoras, incluindo: criação, remoção, pesquisa e edição de operadoras.
- Gestão de Componentes - Conjunto de funcionalidades relacionadas à gestão dos componentes de rede, incluindo: edição do nome de atributos, mapeamento de atributos e configurações de reconhecimento. Estas funcionalidades colaboram no momento da extração da informação dos documentos a serem importados.
- Gestão de *Templates* - Conjunto de funcionalidades relacionadas à gestão de *templates*, incluindo: adição, pesquisa e remoção de *templates*.
- Gestão de Regras - Conjunto de funcionalidades relacionadas à gestão das regras de construção do **BoM**, incluindo: criação, remoção, pesquisa e edição de regras relativas à seleção dos componentes de rede onde devem ser aplicadas, assim como de sub regras relativas aos materiais a serem aplicados sobre os componentes.
- Gestão de Utilizadores - Conjunto de funcionalidades relacionadas à gestão de utilizadores, incluindo: pesquisa e remoção de utilizadores.
- Gerador BoM - Conjunto de funcionalidades relacionadas à geração do **BoM**, incluindo: submissão dos ficheiros de projeto, seleção da operadora, *template* **BoM**, tipo de **BoM** e tipo de rede.

- Autenticação - Conjunto de funcionalidades relacionadas à autenticação, incluindo: registo, *login* e *logout*.

A aplicação também deverá ser multi-plataforma e conseguir executar nos sistemas operativos: Windows, Mac e Linux. As configurações efetuadas sobre a aplicação têm de ser guardadas de modo a permitir o seu acesso por múltiplos utilizadores.

Na seguinte Figura 12 ilustra-se um modelo de domínio construído para representar e organizar os conceitos envolvidos na aplicação, permitindo uma visão global de todo o sistema e impulsionar a sua compreensão.

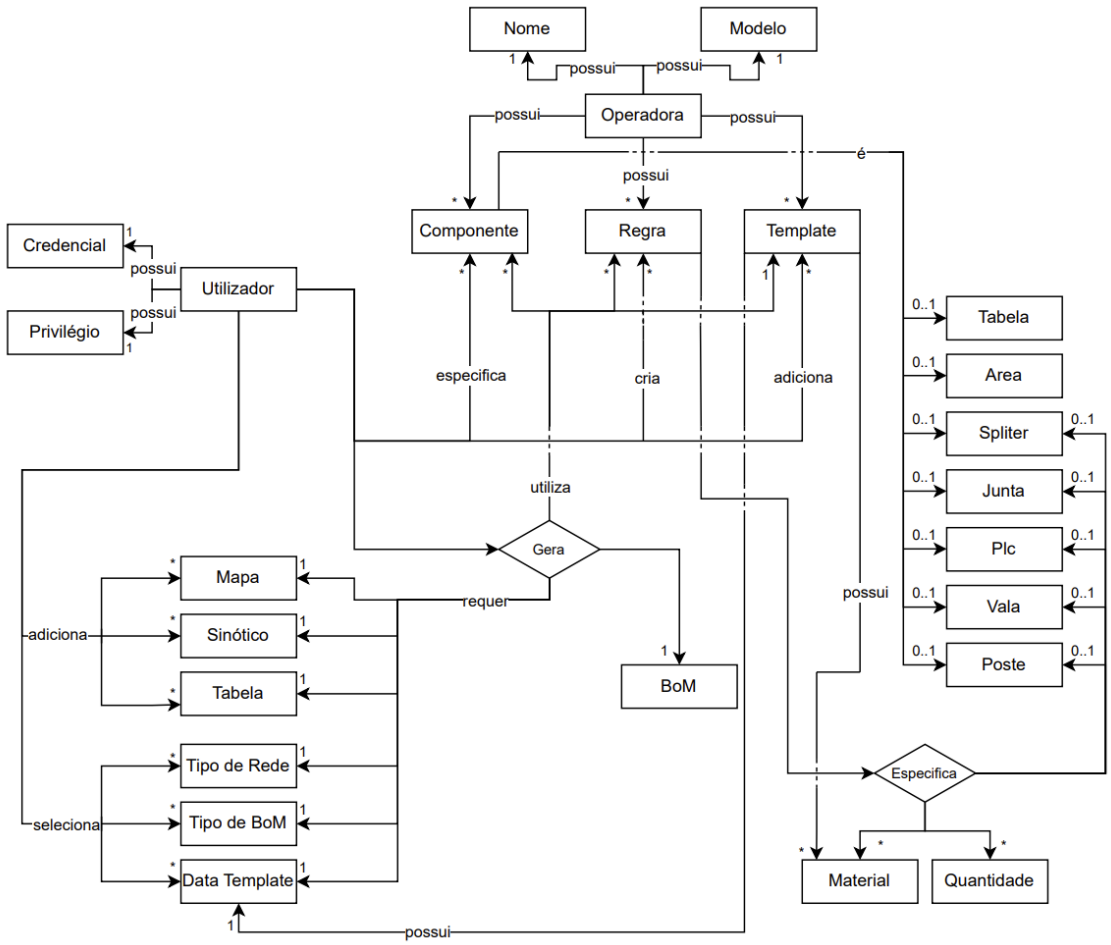


Figura 12: Modelo de domínio da aplicação.

3.2 Requisitos Não Funcionais

Os requisitos não funcionais estão relacionados às características que influenciam diretamente o utilizador e a qualidade da aplicação como um todo, logo, devem ser considerados ao longo do desenvolvimento.

A sua definição teve em consideração a finalidade e ambiente de utilização pretendido para a mesma. Estes podem ser divididos em dois sub-grupos conforme a sua interação, tal como na Figura 13.

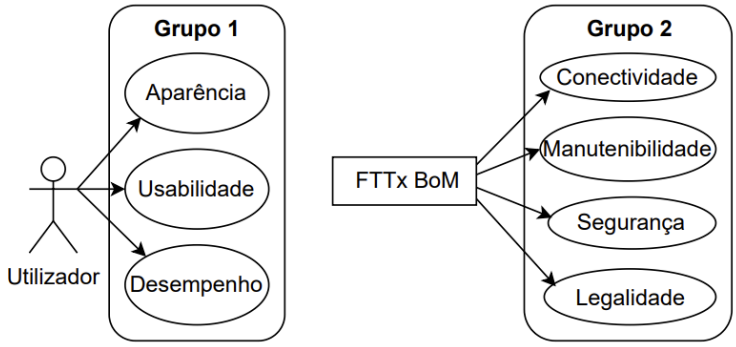


Figura 13: Requisitos não funcionais.

O grupo 1 é relativo às características que afetam diretamente o utilizador da aplicação, incluindo:

- Aparência e Usabilidade - Apesar de referidos, não foram especificados aspetos relativos à aparência e usabilidade da *interface*, uma vez que se trata de um protótipo com foco concretização de funcionalidades.
- Desempenho - Conjuntos de características relativas ao desempenho da aplicação. Incluindo intervalos de tempo máximos para a geração do **BoM**.

O grupo 2 é relativo às caraterísticas com influência na aplicação como um todo, influenciando o desenho da arquitetura.

- Conectividade - Exigência de conetividade à Internet para utilização da aplicação.
- Manutenibilidade - Conjunto de características relativas à forma como a arquitetura da aplicação deve ser desenhada, como, a utilização de componentes modulares (sem influência direta uns nos outros) que permitam uma simples leitura e compreensão do código. Desta forma, é garantida uma boa manutenibilidade a longo prazo.
- Legalidade - Todas as tecnologias utilizadas para o desenvolvimento deverão ser livres de licenciamento e *open-source*, para que desta forma sejam minimizados os custos envolvidos com o desenvolvimento e utilização da aplicação.
- Segurança - Conjuntos de características relativas à segurança e acesso aos dados guardados. Aplicar encriptação às credenciais dos utilizadores e garantir que o acesso aos dados é efetuado apenas por utilizadores autenticados e com o devido privilégio de acesso.

Capítulo 4

Arquitetura e Tecnologias

Ao longo deste capitulo apresentam-se os modelos arquiteturais considerados para a aplicação, juntamente das suas principais vantagens e desvantagens, terminando com a seleção e justificação do mais adequado. Por último, são apresentadas as tecnologias suscetíveis de serem utilizadas no futuro e durante o desenvolvimento.

4.1 Modelo Arquitetural

À medida que incrementalmente os requisitos evoluíram consideraram-se os seguintes três modelos arquiteturais, iniciando-se pelo modelo Standalone, inicialmente pensado como suficiente para a aplicação, e terminando com o modelo selecionado cliente-Servidor em duas camadas.

Standalone

Consiste na criação de uma aplicação de Desktop independente (*standalone*), onde o utilizador após a instalar na sua máquina, imediatamente a poderia utilizar. Este modelo apesar de revelar vantagens como a não necessidade de um servidor de remoto, impossibilita a persistência dos dados para mais de um utilizador (máquina), consequentemente, seria estritamente necessário proceder à configuração de todos os aspetos da aplicação sempre que se pretendesse a sua utilização numa nova máquina.

Possibilita **Vertical scaling** e não possui pontos de **Bottleneck** evidentes. Ilustra-se na Figura 14.

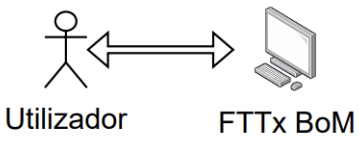


Figura 14: Modelo Standalone.

Cliente-Servidor

Outra opção passaria pela criação de uma aplicação *web* para hospedar num servidor centralizado, ao qual os utilizadores (clientes) acederiam remotamente por um *browser*, tal como um serviço remoto. Esta hipótese apesar de revelar vantagens, como facilidade de acesso e utilização, possui desvantagens como capacidade computacional necessária por parte do servidor, assim como a rapidez de utilização da aplicação estaria altamente dependente da largura de banda, uma vez que seria necessária a troca de ficheiros de projeto, com tamanho considerável, através da Internet sempre que se pretendesse geração de um **BoM**.

Possibilita realizar **Vertical scaling** e **Horizontal scaling**, no entanto, possui um possível ponto de **Bottleneck** devido à largura de banda, uma vez que a aplicação será utilizada por múltiplos utilizadores concentrados nos mesmos edifícios, onde só existirá um servidor. Ilustra-se na Figura 15.

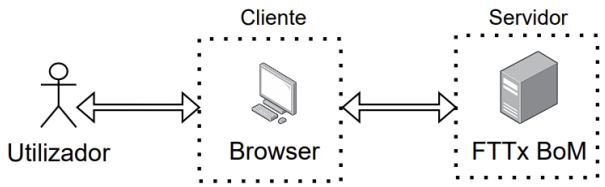


Figura 15: Modelo Cliente-Servidor.

Cliente-Servidor em Duas Camadas

Reunindo as vantagens de cada um dos modelos anteriores, definiu-se um modelo arquitetural cliente-servidor em duas camadas, uma camada de servidor (aplicação de Servidor) e uma camada de cliente (aplicação de Desktop). A primeira garantirá a persistência, a segurança e a gestão do acesso aos dados a múltiplos utilizadores. A segunda, possuirá a lógica de negócio responsável pelas funcionalidades que requerem poder computacional, como o reconhecimento da informação dos ficheiros, a aplicação das regras e a construção do **BoM**, além de fornecer a Interface para manusear as funcionalidades de toda a aplicação FTTx BoM.

Com este modelo arquitetural resolvem-se os dois problemas suscitados pelo modelo *standalone* e cliente-servidor, pois, teremos uma camada de servidor (centralizada) que mitiga a necessidade de configuração em novas máquinas e permite a utilização dos dados por múltiplos utilizadores e, teremos a lógica de negócio com mais exigência computacional na camada de cliente que mitiga a necessidade da troca de ficheiros pela Internet e consequentemente, minimiza a dependência por largura de banda,

além de fatores computacionais relacionados ao servidor.

Consequentemente, este modelo revela-se preparado para **Horizontal scaling**, **Vertical scaling** e mitiga a possível existência de **Bottleneck** causado por falta de largura de banda. O modelo arquitetural selecionado ilustra-se na Figura 16.

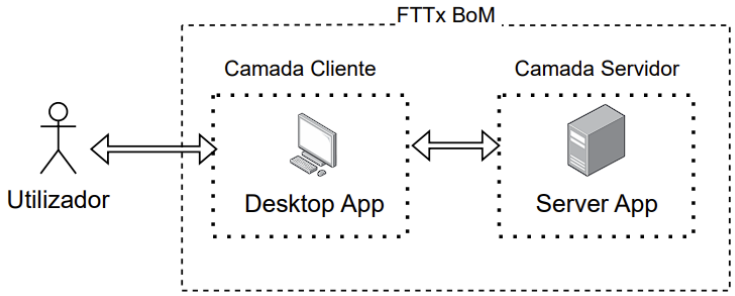


Figura 16: Modelo Cliente-Servidor em Duas Camadas.

4.2 Tecnologias

O presente capítulo explora os formatos dos ficheiros de entrada da aplicação e as tecnologias suscetíveis de se utilizarem na fase de desenvolvimento, assim como a sua seleção e justificação.

4.2.1 Ficheiros DXF e DWG

Como a aplicação a ser desenvolvida tem de conseguir receber como entrada o ficheiro relativo ao sinótico, mapa e tabela de alocação dos projetos de rede, é importante investigar quais os formatos utilizados por esses ficheiros e perceber quais as formas possíveis de os utilizar para selecionar a abordagem mais adequada.

Os formatos estipulados para os ficheiros relativos ao sinótico e mapa de rede, são o *Drawing Web Graphics* (**DWG**) e o *Drawing Exange Format* (**DXF**), pelo que, a aplicação deveria conseguir receber ambos como entrada.

Após investigados os formatos, verificou-se que o formato **DWG** é proprietário da *AutoDesk* e não foi desenhado para ser utilizado em aplicações *Open-Source*, apenas para ser utilizado exclusivamente em ambientes de desenvolvimento fornecidos pela própria *AutoDesk*. Possui o seu conteúdo representado em binário compactado [29], pelo que, a hipótese de construção de um interpretador para o mesmo é descartada, pois, mesmo que fosse possível fazê-lo, se tornaria inútil devido às constantes atualizações por parte da *AutoDesk*. Já o **DXF**, também desenhado pela *AutoDesk*, contrariamente ao **DWG**, é um formato

Open-Source cuja especificação está oficialmente publicada e o seu conteúdo encontra-se representado em carateres **ASCII** sem qualquer compactação [30] [31].

Para o formato **DXF**, sendo *Open-Source*, existe uma comunidade ativa que desenvolve e trabalha sobre ele, portanto, possui uma variedade de ferramentas para auxílio na sua interpretação. Contrariamente, para o formato **DWG**, devido à inexistência de comunidades ativas a desenvolverem e trabalharem em ferramentas *Open-Source* para o mesmo, existem apenas três soluções possíveis para o incorporar na aplicação.

A primeira, a mais simples e confiável, passa pela incorporação na aplicação da ferramenta proprietária da *AutoDesk*, *RealDWG* [32], que oferece precisamente a funcionalidade de leitura e escrita em ficheiros **DWG** e **DXF**, de forma estritamente fiável. No entanto, em contrapartida, a utilização desta ferramenta requer uma licença da *AutoDesk*. A segunda solução, mais complexa e menos fiável, passa pela utilização de um conversor externo para realizar a conversão do formato **DWG** para o formato **DXF**, uma vez que este é um formato possível de ser interpretado e utilizado por ferramentas *Open-Source*. Por último, a terceira solução seria recorrer a bibliotecas existentes capazes de interpretar o **DWG**, assim possibilitando o seu uso, no entanto, limitado às versões suportadas por essas bibliotecas.

Para avaliar a importância do formato **DWG** face ao **DXF**, foram investigadas as suas caraterísticas de diferenciação [33], tendo em mente o contexto de utilização na aplicação, relativamente ao:

- **Formato de armazenamento** - Como mencionado anteriormente, os dados são guardados de forma diferente, sendo o **DWG** em binário compactado, e o **DXF** em **ASCII** não compactado. Assim, concluí-se que o **DWG** é superior para a realização de tarefas relativas ao intercâmbio dos mesmos, principalmente quando o conteúdo guardado é muito extenso.

- **Tipo de desenho** - O formato **DWG** consegue lidar com desenhos tridimensionais e bidimensionais de forma fiável, enquanto o **DXF** apenas consegue lidar com desenhos bidimensionais de forma fiável e poderá apresentar limitações em certos aspetos gráficos, como, por exemplo: qualidade das cores. Ambos têm como principal objetivo guardar imagens construidas sob a forma de geometria.

- **Conversão mútua** - Tanto o formato **DWG** como o formato **DXF** podem ser convertidos um no outro, no entanto, quando se converte de **DWG** para **DXF**, para além da forma mais fiável ser recorrendo a alguma ferramenta da *AutoDesk*, poderá mesmo assim existir perda da informação relativa a recursos específicos da própria ferramenta. A perda de informação acentua-se quando a conversão é efetuada entre desenhos tridimensionais.

Considerando as caraterísticas apresentadas pelos sinóticos e mapas de rede, optou-se pela utilização exclusiva do formato **DXF** como formato nativo na aplicação. Como as caraterísticas exclusivas do formato **DWG** não se revelaram estritamente necessárias, este, poderá no futuro ser integrado recorrendo à utilização de um conversor apto de converter **DWG** em **DXF** e vice-versa. Com o intuito de avaliar esta possibilidade, investigaram-se os seguintes conversores:

1. Conversor *ODA File Converter*, desenvolvido pela *Open Desing Aliance*, uma organização sem fins lucrativos que fornece múltiplas soluções para desenvolvimento sobre **CAD**. Este conversor oferece a funcionalidade de conversão mútua entre **DWG** e **DXF**, e pode ser facilmente utilizado por *scripts Python*, recorrendo à biblioteca *dwg to dxf*. É um conversor gratuito [34].

2. Conversor em nuvem *DWG to DXF Converter*, desenvolvido e fornecido pela *Lunaweb GmbH*, uma organização com foco de mitigar a conversão de ficheiros, sejam eles ficheiros relativos a documentos, arquivos, imagens, áudios e até mesmo vídeos. Tal como o próprio nome indica, fornece a conversão mútua entre **DWG** e **DXF**. É um serviço gratuito [35].

3. Biblioteca *Aspose.CAD* desenvolvida pela empresa *Aspose* para *.NET*, oferece múltiplas funções de gestão de ficheiros, entre os quais, ficheiros **CAD**, para múltiplas plataformas de desenvolvimento em *.Net*. Fornece a conversão mútua entre **DWG** e **DXF**, no entanto, não suporta as versões recentes dos ficheiros. É uma biblioteca *open-source* [36].

4. Conversor *ZWCAD*, desenvolvido pela empresa *Ibercad*, uma organização que oferece múltiplos serviços para lidar com **CAD**. Fornece as mesmas funcionalidades do primeiro, no entanto, não existe nenhuma ferramenta que permita a sua utilização via *script*. Requer licença de utilização [37].

5. Conversor *DWG DXF Converter*, desenvolvido pela *AutoDWG*, uma empresa que, tal como a primeira, oferece múltiplos serviços para lidar com **CAD**, sendo o seu principal, a conversão entre ficheiros. Este conversor tem as mesmas características do anterior relativamente às suas funcionalidades. Requer licença de utilização [38].

A informação apresentada encontra-se sintetizada na seguinte Tabela 3.

Tabela 3: Tecnologias relativas ao planeamento de redes.

Conversor/Biblioteca	Ambiente de execução	Características destacadas
ODA File Converter	Desktop	Possui suporte da desenvolvedora Não requer licença de utilização Executável via *script*
DWG to DXF Converter	Web	Possui suporte da desenvolvedora Largamente utilizado Não requer licença de utilização
Aspose.CAD	Desktop em *.NET*	Suporta ficheiros **CAD** Biblioteca de programação Documentação detalhada
ZWCAD	Desktop	Semelhante ao ODA File Converter Requer licença de utilização Demonstra confiabilidade na conversão
DWG DXF Converter	Desktop	Semelhante ao ZWCAD Requer licença de utilização Possuí versão experimental

A existência de alguns conversores suportados por organizações fidedignas fortifica a possibilidade de integração de um destes na aplicação, garantindo a possibilidade de utilização do formato **DWG** no futuro, pois, sendo este formato constantemente atualizado pela *AutoDesk* [39] é de elevada importância existir suporte por parte da organização que fornece o conversor, caso contrário, rapidamente se tornaria inútil.

A seguir é efetuada uma análise das bibliotecas de programação mais utilizadas para lidar com **DWG** e/ou **DXF**.

1. *GNU LibreDWG* - Biblioteca distribuída pela *GNU*, capaz de lidar com a leitura e escrita de ficheiros em formato **DWG**. Encontra-se ainda numa fase beta não suportando as versões mais recentes do formato e garantindo apenas o bom funcionamento para as versões "R2000", referentes a versões do formato do ano 2000. À data, a biblioteca encontra-se sem suporte ao seu desenvolvimento.

2. *EZDXF* - Biblioteca do *Python*, capaz de lidar com a leitura e escrita de ficheiros em formato **DXF**, possuindo para isso uma **API** completa com documentação intuitiva e uma comunidade de desen-

volvimento relativamente grande quando comparada a outras bibliotecas com o mesmo objetivo. Possibilita também a incorporação por meio de *plugins* ao conversor apresentado anteriormente *ODA File Converter*.

Decidida a não implementação nativa do formato **DWG** na aplicação e tendo em consideração a documentação, comunidade e potencialidade da biblioteca *EZDXF*, esta será a escolhida para tratar da interpretação dos ficheiros **DXF**, portanto, recorrendo à linguagem *Python*. Posteriormente, em fases avançadas, se necessária a incorporação de um conversor para o formato **DWG**, a escolha desta biblioteca também trará como vantagem a capacidade de lidar com o *ODA File Converter*, apresentado anteriormente.

4.2.2 Ficheiros XLSX

Para além do sinótico e do mapa de rede, a aplicação terá de conseguir receber como entradas o ficheiro relativo à tabela de alocação e ao *template* **BoM**, cujos formatos são **XLSX**. Ora para este formato não existirá problema na sua interpretação devido à existência de uma diversidade de bibliotecas preparadas para lidar com ele, como:

1. *Pandas* - Biblioteca do *Python* apta para realizar múltiplas tarefas de manipulação e análise de dados, oferecendo para isso uma **API** completa. Também possui documentação detalhada e é amplamente utilizada em diversos setores.

2. *Openpyxl* - Biblioteca do *Python* com funcionalidades semelhantes à anterior, no entanto, com foco na manipulação de ficheiros **XLSX**, permitindo facilmente o acesso e modificação de células individuais.

3. *Xlsx* - Biblioteca do *NodeJS* semelhante à primeira e segunda.

Para lidar com a interpretação dos ficheiros **XLSX**, qualquer uma das bibliotecas poderá ser utiliza, pois, ambas possuem uma vasta comunidade de desenvolvimento e documentação detalhada. Prevê-se manipular ficheiros **XLSX** tanto no *Python* como no *NodeJS* em aspetos como: leitura e escrita de colunas e linhas, e criação de ficheiros bem formatados.

4.2.3 BackEnd e FrontEnd

A seguir, apresentam-se as tecnologias investigadas para o desenvolvimento da aplicação de Desktop e Servidor, assim como a justificação da sua escolha.

Aplicação de Desktop

A decisão das tecnologias deve considerar a possibilidade de construir uma aplicação *Cross-Platform* para ambientes de execução em Desktop, cuja arquitetura possa ser desenhada o mais modular e flexível possível. Para isso foram investigadas as seguintes tecnologias com foco no *BackEnd*:

- *NodeJS* - Ambiente de execução *open-source* e *cross-platform*, desenvolvido com o motor V8 da Google, capaz de executar *javascript* fora de um navegador comum, garantindo alto desempenho. A sua arquitetura é baseada em eventos assíncronos, sendo altamente escalável e adequado para o desenvolvimento de aplicações *web* aptas para responder a elevadas quantidades de pedidos em simultâneo. Possui uma comunidade ativa que fornece bibliotecas e *frameworks* que facilitam o desenvolvimento *web* e a construção de **API**'s de dados.

- *Python* - Linguagem orientada a objetos, interpretada, *open-source* e *cross-platform*. Possui alta legibilidade e facilidade de utilização, logo é frequentemente utilizada na resolução de problemas complexos, de pouca escalabilidade. Possui uma comunidade ativa que fornece bibliotecas e *frameworks* para diversas finalidades.

Decidiu-se a utilização do *NodeJs* como tecnologia base para o *BackEnd* pois possui todas as caraterísticas necessárias para o desenvolvimento de uma aplicação *web*, e a utilização do *Python* como tecnologia base para o desenvolvimento de um reconhecedor para os ficheiros de entrada, uma vez que possibilita a utilização da vantajosa biblioteca *EZDXF*.

Após decidida a utilização de *NodeJs* investigou-se a existência de *frameworks* para auxiliar o desenvolvimento do *BackEnd*, uma vez que se pretende uma aplicação para ambientes de execução em Desktop.

- *Electron* - *Framework open-source* e *cross-platform* focado para o desenvolvimento de aplicações de Desktop. Incorpora o Chromium, permitindo desenvolver *interfaces* recorrendo a tecnologias *web*. Estas características tornam o *Electron* uma escolha popular para o desenvolvimento de aplicações *web* para Desktop.

A *framework* selecionada foi o *Electron* devido a possuir incorporação do Chromium, disponibilizando uma diversidade de ferramentas de *debug* para o desenvolvimento da Interface, assim como possui uma curva de aprendizagem reduzida para a sua inicial utilização.

Por último, restava investigar a tecnologia a utilizar no *FrontEnd* para construir a Interface.

- *React* - Biblioteca apta para o desenvolvimento de *interfaces* modulares e reutilizáveis, recorrendo ao conceito de componentes. Possui uma linguagem de estruturação (**JSX**), que permite escrever código combinando elementos de *html* e de *javascript*, simplificando a manipulação do **DOM**. É amplamente utilizado na construção de aplicações *web* e possui diversas outras bibliotecas que cooperam com esta.

Rapidamente se decidiu a utilização do *React* devido às suas caraterísticas de manutenção, flexibilidade e documentação.

Aplicação de Servidor

A decisão das tecnologias para esta aplicação devem considerar a possibilidade de construção de uma aplicação de servidor capaz de receber e processar pedidos **HTTP**, assim como garantir a persistência e a segurança dos dados, e fornecer mecanismos de autenticação.

Decidida a utilização de *NodeJS* no *BackEnd* da aplicação de Desktop, decide-se também a sua utilização na aplicação de Servidor, optando-se pela *framework ExpressJS*.

- *ExpressJS* - *Framework* que fornece mecanismos capazes de criar *endpoints* e lidar com roteamento de pedidos **HTTP** feitos a estes. Utiliza o conceito de *middleware* quando existe a necessidade de aplicar processamentos aos pedidos antes de chegarem à respetiva rota de tratamento. Permite o acesso e a manipulação de parâmetros dos cabeçalhos HTTP. Altamente modular.

Por último, para perceber qual a forma de armazenamento mais adequada para a aplicação, foram investigados os dois paradigmas existentes de modo a avaliar qual o mais indicado tendo em consideração os dados que se pretendem guardar e ler, além da sua volatilidade ao longo do desenvolvimento [40].

- **SQL** - Paradigma para armazenamento de dados de forma estruturada que não permite armazenar dados sem que antes seja definida a sua estrutura, tipagem e relação. Dadas estas caraterísticas, são úteis no desenvolvimento de aplicação estáveis, cuja variabilidade dos dados seja reduzida, assim como, em antecedência ao desenvolvimento da aplicação, se conheçam detalhadamente os dados a armazenar. Este paradigma, devido a respeitar estritamente as propriedades **ACID**, é indicado para aplicações com transações de dados críticas. Altamente escalável verticalmente.

- **NoSQL** - Contrariando os princípios do anterior, este não requer a definição da estrutura, tipagem e relação dos dados, antes de poderem ser armazenados, portanto, é considerado útil no

desenvolvimento de aplicações voláteis, cuja estrutura dos dados armazenados apresente variabilidade ao longo do tempo. Este paradigma possui diversos sub-paradigmas, por isso, não possui uma linguagem de consulta própria, dependerá sempre do sub-paradigma. Altamente escalável horizontalmente.

Dadas as vantagens do paradigma **NoSQL** em conjunto com as características de volatilidade apresentadas pelos dados na fase inicial da aplicação, decidiu-se a utilização do paradigma **NoSQL**, restando apenas decidir qual o sub-paradigma a ser utilizado, analisado em seguida.

- Chave-Valor - Estrutura os dados sob a forma de pares chave-valor que se compõem por uma chave única e um valor associado, que pode ser outro par chave-valor. Esta organização permite realizar consultas eficientes quando efetuadas recorrendo a uma chave, no entanto, deixam de o ser quando efetuadas recorrendo a diversas chaves, ou seja, quando os dados armazenados possuem hierarquia. Desta forma, não é o sub-paradigma adequado para ser utilizado nesta aplicação, sendo útil apenas para armazenar dados simples, com poucas estruturas aninhadas, cuja consulta possa ser efetuada rapidamente recorrendo a uma chave [41].

- Documental - Estrutura os dados sob a forma de documentos em formatos específicos (tipicamente **JSON** ou **XML**) compostos por estruturas Chave-Valor, organizadas hierarquicamente, tipicamente para agrupar todos os dados relacionados. Devido a estas caraterísticas, rapidamente se obtém todos os dados pretendidos com uma consulta, além de permitir realizar consultas eficientes recorrendo a diversas chaves. Revela-se, devido às suas caraterísticas, o paradigma mais adequado para os dados previstos de serem guardados na aplicação [42].

- Colunar - Estrutura os dados sob a forma de famílias de colunas, em que, cada uma possui linhas com os dados guardados sob a forma de pares Chave-Valor, constituindo cada par uma coluna. Este sub-paradima tem semelhança paradigma SQL, no entanto, contrariamente a este, fornece flexibilidade nas colunas existentes, enquanto no SQL as colunas são estáticas. É normalmente utilizado em cenários de elevadas quantidades de dados onde cada família de colunas possui poucas linhas e muitas colunas, não sendo o caso dos dados previstos de serem utilizados na aplicação [43].

- Grafos - Estrutura os dados sob a forma de grafos, utilizando nodos para representar as entidades e arestas para as relações. Devido a estas caraterísticas, possibilita a aplicação de algoritmos de pesquisa nas consultas, consequentemente, torna-se um sub-paradigma altamente eficiente quando

utilizado com pequenas quantidades de dados altamente relacionados. Assim, este paradigma é descartado devido aos dados previstos de serem guardados na aplicação não apresentarem tais características. Tipicamente ferramentas de suporte a este sub-paradigma possuem uma *interface* gráfica que permite visualizar e compreender os dados [44].

Concluída a investigação, foi decidia a utilização do paradigma **NoSQL** com sub-paradigma Documental, principalmente devido à consulta de diversos dados relacionados em simultâneo no momento da geração de um **BoM**. A tecnologia escolhida para cumprir com este requisito foi o MongoDB, devido ao seu alto desempenho nas consultas e ampla comunidade de desenvolvimento.

4.2.4 Autenticação e Segurança

A autenticação é um assunto transversal às aplicações de Desktop e Servidor. A primeira fornecerá mecanismos que possibilitem a realização da autenticação e estabelecimento de uma sessão. Já a segunda realizará toda a lógica da autenticação, assim como garantirá que o acesso às suas rotas é efetuado por utilizadores autenticados com o privilégio de acesso adequado.

A tecnologia selecionada para realizar o pretendido segue:

- *Json Web Token* - Fornece mecanismos para autenticação e estabelecimento de sessões recorrendo a *tokens* constituídos por três componentes: o cabeçalho, constituído pelo tipo do *token* e o algoritmo de assinatura; a carga útil, constituída por informações úteis para identificação do utilizador autenticado; e a assinatura, gerada com base nos dois componentes anteriores e numa chave secreta definida no sistema que fornece o *token*.

Também, para garantir a segurança das credenciais dos utilizadores, as senhas no momento do registo serão encriptadas antes de serem guardadas na base de dados, recorrendo à tecnologia seguinte:

- *BCrypt* - Biblioteca criptográfica do *NodeJS* capaz de encriptar palavras de caracteres com elevado grau de segurança a ataques de força bruta, recorrendo a uma implementação do algoritmo de encriptação *Blowfish* [45].

A biblioteca *BCrypt* revela-se mais que suficiente para proteger as credenciais dos utilizadores da aplicação, uma vez que todos eles serão colaboradores da empresa e não existirá acesso externo.

4.2.5 Empacotamento e Implantação

Sendo a aplicação FTTx BoM (Desktop e Servidor) um protótipo, completado e melhorado ao longo do tempo, é importante definir um mecanismo que garanta uma simples implantação, tanto para a aplicação de Desktop como para a de Servidor, capaz de mitigar a necessidade de realizar os processos de instalação da aplicação e suas dependências.

A aplicação de Desktop, tal como o nome indica, será uma aplicação que executará como um processo num sistema operativo de uma máquina comum. Assim, para prevenir o utilizador da necessidade de executar comandos via linha de comandos para sua inicialização, será utilizada a biblioteca *Electron Packager* para empacotar a aplicação num ficheiro executável, uma vez que o seu desenvolvimento será feito em *Electron*.

Já para a aplicação de Servidor, sendo o servidor de implantação da Proef, existe a possibilidade deste estar a dar suporte a uma diversa gama de aplicações, portanto é necessário garantir o isolamento entre estas. Desta forma, será empacotada recorrendo à tecnologia *Docker* que utiliza ao conceito de contentores para isolar as aplicações e todas as suas dependências. A seguinte Figura 17 ilustra o conceito do Docker.

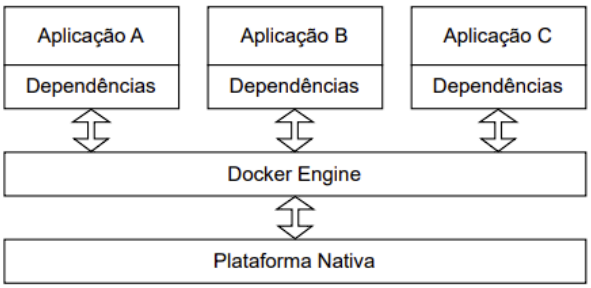


Figura 17: Docker.

A utilização do Docker apresenta como principais vantagens para esta aplicação:

- Isolamento - Os processos de cada contentor executam isoladamente dos processos de outros contentores, logo, por exemplo: o mau funcionamento de um, não comprometerá os restantes.
- Portabilidade - Os contentores executam sobre o motor (*Docker Engine*) *Cross-Platform*, portanto, é possível executar qualquer contentor de igual forma independentemente da plataforma subjacente. Também, graças a ferramentas do próprio Docker é possível recorrendo a um ficheiro de configuração, gerir a instalação e configuração dos contentores.

- Eficiência de recursos - Graças ao motor mencionado anteriormente, os contentores executam sobre a mesma plataforma subjacente, sendo vistos como um processo do sistema operativo, logo, consomem menos recursos e não requerem alocação de recursos em antecipado. Desta forma, tornam-se altamente eficientes quando comparados com a utilização de Máquinas Virtuais, neste caso.

- Escalabilidade - O Docker fornece ferramentas que tornam a gestão de contentores um processo rápido de ser efetuado, logo possibilita realizar **Horizontal scaling** de forma eficiente e ajustada à demanda. Esta vantagem revela ser uma mais-valia para o futuro da aplicação.

A informação apresentada encontra-se sintetizada na seguinte Tabela 4.

Tabela 4: Tecnologias relativas ao planeamento de redes.

Tecnologia	**FAM** %	Característica destaque	Utilização prevista
EZDXF	70	Manipulação de **DXF**	Reconhecedor - Desktop
Pandas Openpyxl Xlsx	60	Manipulação de **XLSX**	Reconhecedor - Desktop
Electron	70	Criação de aplicações Web Desktop	Núcleo - Desktop
React	60	Criação de Interfaces modulares	Interface - Desktop
Express	85	Criação de **API**'s	API - Server
MongoDB	70	Armazenamento sem modelagem	DB - Server
Json Web Token	70	Estabelecimento de sessão	Server e Desktop
BCrypt	90	Encriptação de dados	Server App
Electron Packager	80	Empacotamento de Electron Apps	Desktop
Docker	60	Empacotamento e implantação de Apps	Server

4.3 Arquitetura da aplicação de Desktop

A arquitetura projetada para a aplicação de Desktop ilustra-se na Figura 18.

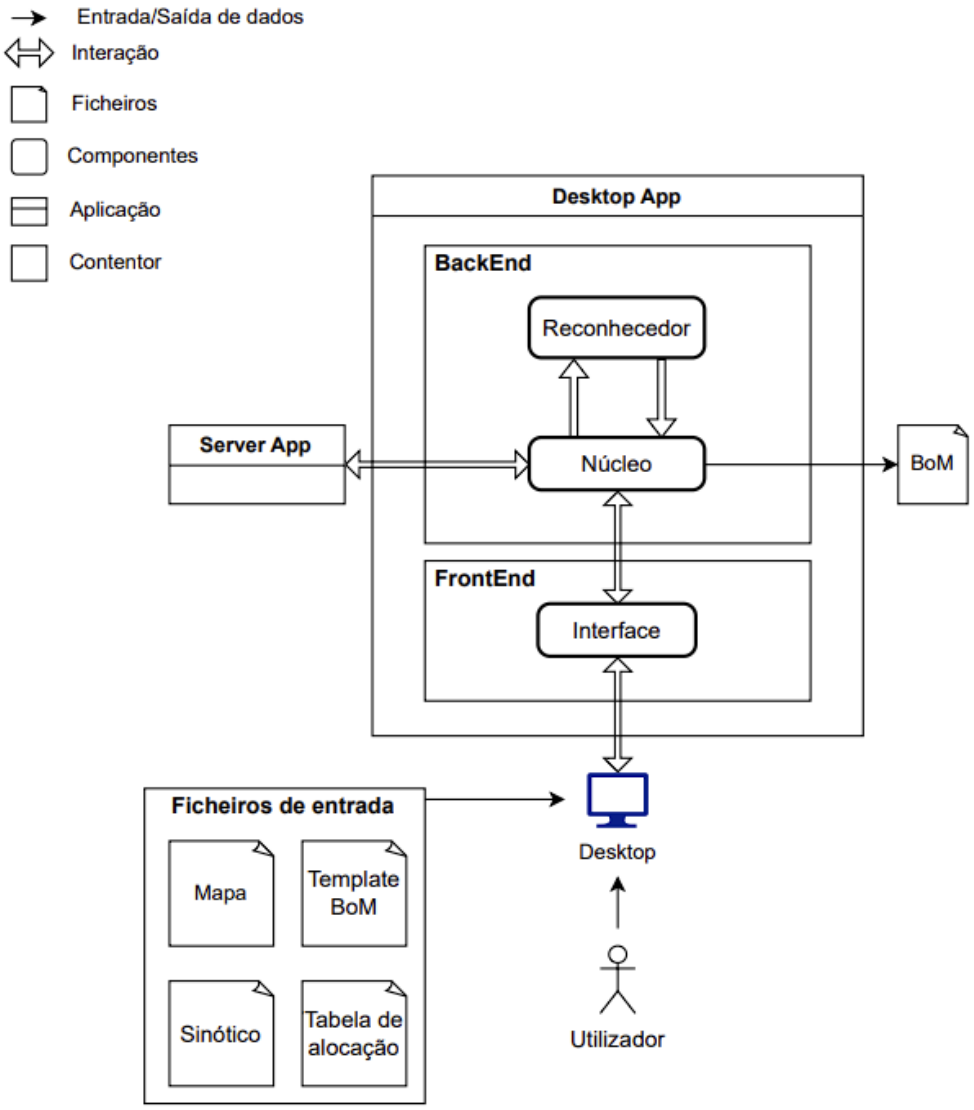


Figura 18: Arquitetura da aplicação de Desktop.

A arquitetura é composta por três componentes principais, dois deles pertencentes ao *BackEnd* e o restante ao *FrontEnd*. O primeiro componente, o *Reconhecedor*, pertencente ao *BackEnd* foca-se nas tarefas de reconhecimento de informação dos ficheiros a serem utilizados pela aplicação, tais como: sinóticos, mapas, tabelas de alocação e *templates* **BoM**. A extração de informação será tratada pelo Reconhecedor conforme o tipo de ficheiro em questão, produzindo como resultado um ficheiro a ser utilizado pelo segundo componente.

O Reconhecedor revela um papel importante na incorporação de novas operadoras na aplicação FTTx BoM. Este deverá possuir tantas versões quanto o número de operadoras que a aplicação FTTx BoM suportar, isto porque, cada operadora organiza os projetos de planeamento distintamente, podendo possuir ficheiros diferentes em quantidade, organização e estruturação interna, consequentemente, a

extração da sua informação será sempre distinta. Cada operadora que se pretenda inserir na aplicação terá de ser analisada detalhadamente, de modo a desenvolver um Reconhecedor específico para a própria, capaz de extrair a informação útil dos ficheiros.

Para minimizar a necessidade de modificar os restantes componentes, os Reconhecedores deverão sempre produzir como resultado um ficheiro com uma estrutura pré-definida, onde conste o conteúdo necessário para a geração do **BoM**. Isto conseguir-se-á sempre atingir, pois todas as operadoras têm em comum os aspetos utilizados para se decidirem quais os materiais a serem aplicados sobre os componentes, independentemente de como os ficheiros de projeto estejam organizados, a informação essencial estará sempre contida neles.

O segundo componente, o Núcleo, também pertencente ao *BackEnd*, tal como o nome indica, será o Núcleo da aplicação, responsável por invocar a execução do Reconhecedor quando necessária a extração de informação de algum ficheiro, responsável pela interação com a aplicação de Servidor e pela geração do **BoM**, além de fornecer o suporte ao *FrontEnd*.

O terceiro componente, a Interface, pertencente ao *FrontEnd*, disponibilizará ao utilizador a Interface gráfica com as funcionalidades previstas para a aplicação, já apresentadas no capítulo anterior. A Interface será também responsável por carregar os ficheiros de projeto do sistema de arquivos para a aplicação, no momento da sua submissão para o Núcleo.

Lógica de funcionamento

O utilizador submeterá os ficheiros, *template* **BoM**, mapa, sinótico e tabela de alocação através da Interface. Estes ficheiros serão recebidos e armazenados pelo Núcleo numa diretoria especifica para armazenamento dos mesmos, conhecida pelo Reconhecedor. Os ficheiros têm diferentes tipos e objetivos na aplicação, sendo tratados de acordo.

O Reconhecedor, o componente que realiza o reconhecimento da informação destes ficheiros, deverá estar preparado para lidar com cada um dos diferentes tipos. No momento da sua invocação será informado de qual o tipo de ficheiro que lhe será submetido, pelo que, saberá como proceder perante ele. Também, será informado do nome do ficheiro para saber exatamente qual o ficheiro a processar. O resultado do Reconhecedor será guardado numa diretoria especifica da qual o Núcleo terá conhecimento, podendo consumir o resultado logo que o Reconhecedor termine a sua execução.

O *template* **BoM** é utilizado de duas formas distintas na aplicação. Primeiramente, como é nele que constam todos os materiais que a operadora correspondente pode utilizar, é importante extraí-los e guardá-los associados à respetiva operadora, garantindo assim que a sua utilização na elaboração das

regras possa ser feita sem recurso à sua nova importação. Devido ao *template* possuir atualizações mensais dos materiais que nele constam, no momento da associação dos mesmos à respetiva operadora é necessário indicar a data do mesmo. Desta forma, o utilizador poderá criar regras recorrendo a materiais de *templates* com diferentes datas, assim como, poder escolher qual o pretendido no momento da geração do **BoM**.

Também, como o *template* possui diversas informações desnecessárias e não padronizáveis, guardá-lo numa base de dados seria pouco eficiente e implicaria a mudança de esquema constantemente, uma vez que, de operadora para operadora a sua estrutura pode variar crucialmente, ou mesmo entre versões do próprio *template*. Assim, este será guardado no sistema de arquivos da aplicação de Servidor associado à sua respetiva data, e enviado à aplicação de Desktop sempre que for solicitada a geração de um **BoM**. Isto evitará a necessidade de submeter o *template* à aplicação de Desktop sempre que se pretender gerar um **BoM**.

No momento da geração do **BoM**, o Núcleo saberá exatamente a localização dos materiais no *template* da operadora, graças ao mapeamento feito no momento da submissão do *template*. Portanto, será apenas necessário acrescentar a informação relativa às quantidades necessárias de cada material e exportar o **BoM**. O mapeamento feito no momento da adição do *template* à operadora, especifica quais são os atributos-chave com informação relevante para a criação das regras e identificação da posição do material no documento.

A geração do **BoM** será feita em duas fases principais, a fase de extração de informação, onde é invocado o Reconhecedor para realizar a tarefa de reconhecimento, e a fase da aplicação das regras definidas, realizada pelo próprio Núcleo.

4.4 Arquitetura da aplicação de Servidor

Sendo esta uma aplicação independente é necessário também projetar a sua arquitetura, ilustrada na Figura 19.

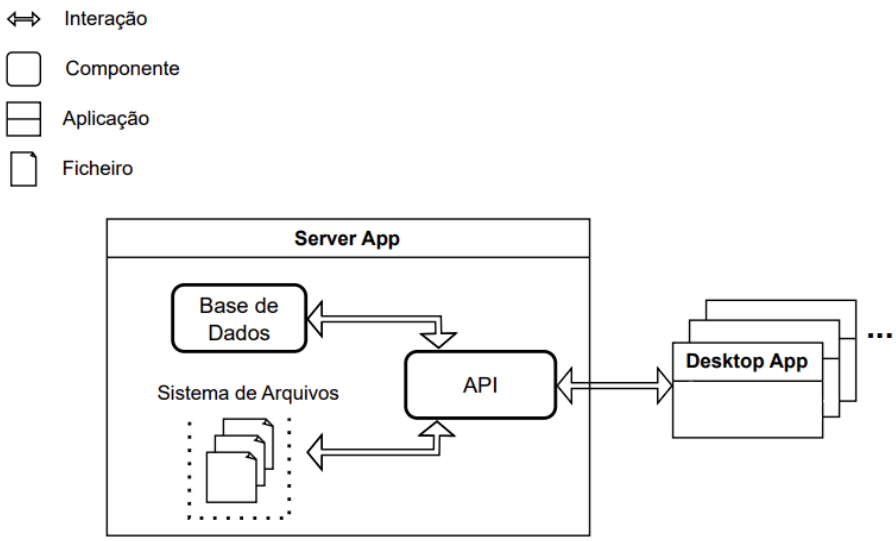


Figura 19: Arquitetura da aplicação de Servidor.

A arquitetura constituí-se por dois componentes e pelo sistema de arquivos. O primeiro componente, a Base de Dados, será responsável pelo armazenamento de todos os dados relacionados a operadoras, como a configuração dos componentes, *templates*, materiais, regras e utilizadores registados na aplicação.

O segundo componente, a API, realizará a gestão dos dados, o processo de autenticação, o controlo de concorrência e a verificação do privilégio de acesso aos dados armazenados. Esta possuirá um conjunto de rotas pré-estabelecidas às quais as aplicações de Desktop deverão efetuar pedidos **HTTP** para assim obterem os dados pretendidos.

O sistema de arquivos guardará os *templates* **BoM** das operadoras registadas na aplicação, tal como dito anteriormente, para evitar a recorrente submissão por parte dos utilizadores sempre que pretendam gerar um **BoM**.

Capítulo 5

Desenvolvimento

Ao longo deste capitulo serão detalhados os componentes de ambas as aplicações, assim como os seus passos de desenvolvimento recorrendo às tecnologias decididas e decisões tomadas no capítulo anterior.

5.1 Aplicação de Desktop

O desenvolvimento iniciou-se pela aplicação de Desktop, especificamente pelo componente Reconhecedor, devido ao desafio inicial constar na extração da informação dos ficheiros. Numa primeira etapa do sinótico, mapa e tabela de alocação, uma vez que a informação destes documentos são a base do propósito da aplicação, e numa segunda etapa, do *template* **BoM**.

Após desenvolvida a primeira versão do Reconhecedor partiu-se para o desenvolvimento do Núcleo, tendo como objetivo inicial estabelecer um mecanismo capaz de invocar o Reconhecedor e importar os resultados por ele produzidos. Seguido-se o desenvolvimento incremental de ambos os componentes, Núcleo e Reconhecedor, conforme os conjuntos de tarefas planeados inicialmente. A Interface foi desenvolvida aquando da implementação das funcionalidades no Núcleo, de modo a facilitar a execução de testes manuais durante o desenvolvimento.

5.1.1 Componente Reconhecedor

Tal como decidido anteriormente o Reconhecedor foi desenvolvido em *Python*, cuja arquitetura se apresenta na Figura 20, constituída por dois subcomponentes principais, o PythonDXF e o PythonXLSX. O primeiro efetua as tarefas de reconhecimento dos ficheiros relativos ao sinótico e mapa, que constam em formato **DXF**, recorrendo à utilização da biblioteca *EZDXF*. O segundo efetua as tarefas de reconhecimento dos ficheiros relativos ao *template* **BoM** e tabela de alocação, que constam em formato **XLSX**, pelo que, requerem a utilização da biblioteca *Pandas* e *Openpyxl*.

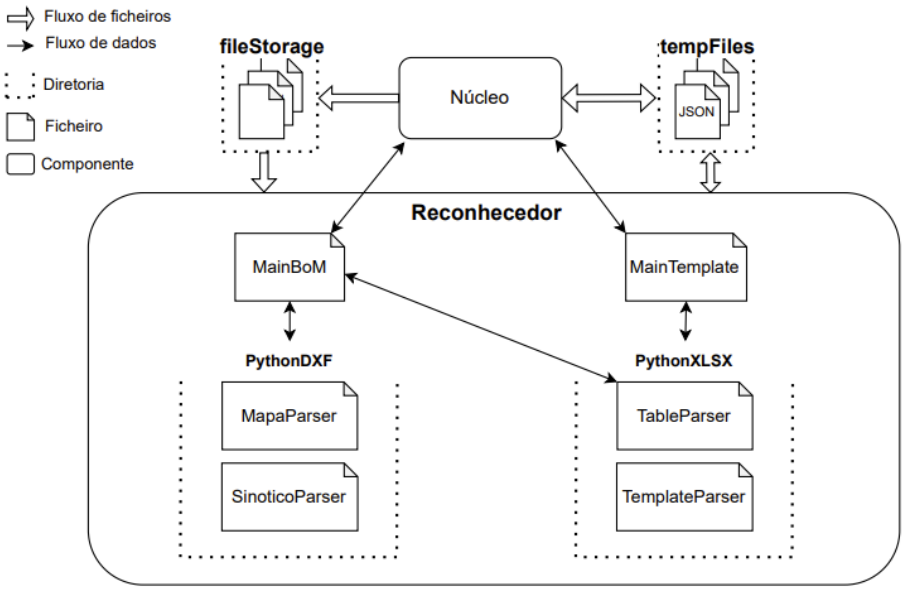


Figura 20: Arquitetura do Reconhecedor.

O Reconhecedor possui dois pontos de entrada, o MainBoM, invocado quando se pretende a extração de informação dos ficheiros para construção do **BoM**, e o MainTemplate, invocado quando se pretende a extração dos materiais contidos no *template* **BoM**. Por possuírem finalidades distintas possuem argumentos de entrada distintos em tipo e quantidade.

Os ficheiros de projeto recebidos pelo Núcleo são guardados na diretoria *fileStorage* para posterior acesso pelo Reconhecedor. Na diretoria *tempFiles* é colocado pelo Núcleo o ficheiro com as informações auxiliares para reconhecimento, e também colocados os resultados produzidos pelo Reconhecedor.

Extração da informação para o BoM

O primeiro passo do processo de geração de um **BoM** é feito imediatamente antes do processo de invocação do Reconhecedor, consistindo numa exportação feita pelo Núcleo, de um ficheiro para a diretoria *tempFiles* onde são colocadas as informações necessárias para o reconhecimento dos documentos da respetiva operadora. Estas informações incluem os nomes dos atributos a reconhecer entre outros auxiliares de reconhecimento abordados posteriormente. A figura 21 ilustra o conceito do mapeamento de atributos, implementado para permitir um certo grau de variabilidade entre projetos da mesma operadora.

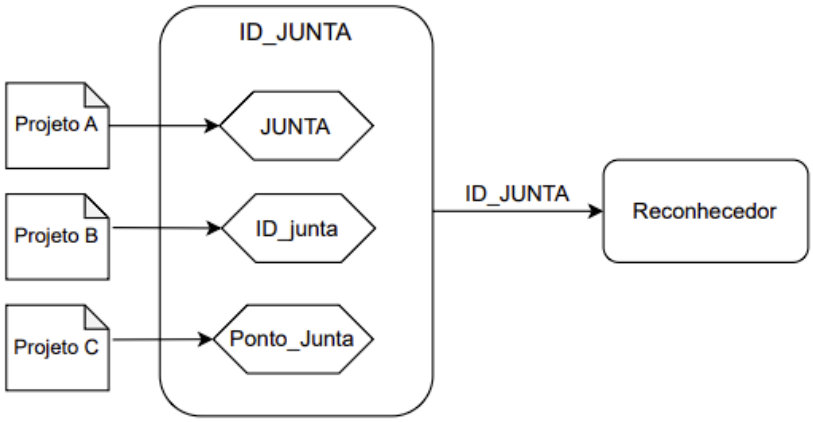


Figura 21: Mapeamento de atributos.

No exemplo ilustrado existem três projetos, em que cada um representa com um nome diferente o atributo relativo à identificação da Junta (JUNTA, ID_JUNTA, PONTO_JUNTA). No momento da geração do **BoM** é passado ao reconhecedor o atributo ID_JUNTA com a identificação respetiva para o projeto em questão. Com este mecanismo elimina-se a necessidade de modificar o Reconhecedor sempre que há alguma variação no nome dado ao atributo.

Como dito anteriormente, em conjunto com o mapeamento de atributos são passados ao Reconhecedor auxiliares para o reconhecimento dos componentes, como:

- Componentes descartáveis - Características como: nomes atribuídos a identificadores (*layers*) de componentes do terreno não pertencentes ao projeto de rede.
- Tabela de alocação - Características como a quantidade de linhas de cabeçalho da tabela a ignorar, quantidade de colunas e identificação das que revelam importância para o **BoM**.
- Auxiliares de deteção - Configurações para ajuda no reconhecimento, como: especificação de distâncias entre a posição de determinados componentes no sinótico, utilizada para a deteção de cabos partilháveis.
- Contabilização de componentes - Configuração das situações que ocorrem internamente ao projeto e que implicam a contabilização de algum componente, como, por exemplo: tubos de subida, sempre que existe uma transição do tipo de instalação de conduta para fachada, no caso dos cabos.

A lógica de negócio implementada organiza-se da seguinte forma:

- MainBoM - Recebendo como argumentos o: sinótico, mapa e tabela de alocação, começa por importar as informações de reconhecimento do ficheiro exportado pelo Núcleo e, em seguida, invoca o SinoticoParser, de onde obtém componentes como: juntas, **PLC**'s e *spliters*. A seguir, invoca o MapaParser, de onde obtém componentes como: postes, cabos e valas. Após estas duas extrações a própria MainBoM procede à organização da informação numa única estrutura de dados, normalizando os dados conforme o tipo e estrutura pretendida. A seguir, invoca o TableParser, pertencente ao PythonXLSX, uma vez que estamos perante um ficheiro agora em formato **XLSX**, obtendo como resultado as fusões pertencentes às juntas e **PLC**'s. Por último, é feita a deteção da quantidade de tubos de subida, uma vez que estes são determinados pelo número de transições de cabos de conduta para fachada. Terminada a extração, é então exportado o ficheiro com a estrutura de dados já normalizada para posterior importação pelo Núcleo.

- SinoticoParser - Este começa a extração verificando se está perante um componente do tipo: junta ou **PLC**, pois, apesar de terem os mesmos atributos, é importante distinguir se é um ou outro. Seguidamente são extraídos os atributos: id, tipo, instalação, *spliters* internos, identificador do cabo de fornecimento de sinal, além de verificado o tipo de rede a que o componente pertence, e se está contido em alguma grelha de representação de componentes para não construção. No final é passado à MainBoM o resultado obtido.

- MapaParser - Este extrai apenas a informação que não consta no sinótico, portanto: extrai os postes e valas pertencentes à rede, verificando tal através dos seus identificadores; informações complementares para os cabos, como o tipo, capacidade e comprimento utilizado em cada tipo de instalação (conduta, fachada ou aéreo). No final, o resultado é passado à MainBoM.

- TableParser - Este extrai as fusões pertencentes a cada junta ou **PLC**, constituídas por: tipo de fusão, tipo de aplicação e célula de localização.

Como dito na secção 2.2, o desenvolvimento da aplicação teve foco nos projetos de uma operadora específica, no entanto, era um requisito a aplicação suportar mais do que uma operadora. Ora, desenvolver a aplicação de modo a suportar qualquer tipo de operadora sem uma lógica previamente desenvolvida, era impossível de ser cumprido no intervalo de tempo alocado para a realização desta dissertação. Assim, a decisão arquitetural tomada, já apresentada, para mitigar a necessidade de modificações significativas na arquitetura da aplicação, com a inserção de novas operadoras, foi a utilização de uma estrutura de dados intermédia.

Esta estrutura de dados intermédia normalizada, possui todas as informações relevantes para o **BoM** e é o produto final do Reconhecedor, encontrando-se divida em três secções principais não relacionadas: global, postes e valas:

- Global - Subestrutura com a lista de todas as juntas e **PLC**'s, contendo toda a informação relacionada, como:
  - tipoComponente - Indica se estamos perante um componente do tipo junta ou **PLC**.
  - secundaria - Indica se o componente pertence ou não à rede secundária.
  - tipo - Indica o tipo de equipamento utilizado para o componente.
  - instalacao - Indica o tipo de instalação do componente.
  - cabo - Subestrutura que possui informação relativamente ao cabo que dá sinal a este componente.
    - secundaria - Indica se o cabo pertence ou não à rede secundária.
    - capacidade - Indica a capacidade do cabo.
    - tipos - Subestrutura que possui informação relativamente aos tipos de instalação e comprimento de cabo.
      - metrosF - Comprimento de cabo instalado em fachada.
      - metrosC - Comprimento de cabo instalado em conduta.
      - metrosA - Comprimento de cabo aéreo.
  - fusões - Subestrutura que possui informação relativamente às fusões efetuadas no componente.
    - tipo - Indica o tipo de fusão.
    - aplicacao - Indica a forma de aplicação.
    - celula - Indica célula de aplicação.
    - quantidade - Indica a quantidade de fusões com as características de cima.
  - spliters - Subestrutura que possuí informação relativamente aos *spliters* contidos no componente.
    - tipo - Indica o tipo do *spliter*.
    - quantidade - Indica a quantidade de *spliters* do tipo acima.

  - construir - Indica se o componente é ou não para construir, portanto, se é ou não um componente partilhável.
  - construirCabo - Indica se o cabo é ou não para construir, portanto, se é ou não um cabo partilhável.

- Postes - Subestrutura com a lista de todos os postes e informação relacionada, como:
  - tipo - Indica o tipo de poste.
  - altura - Indica a altura do poste.
  - construirPoste - Indica se o poste é ou não para construir, portanto, se é ou não um poste partilhável.

- Valas - Subestrutura com a lista de todas as valas e informação relacionada, como:
  - id - Identifica o nome atribuído às valas.

Extração dos materiais utilizáveis

- MainTemplate - Este é consideravelmente mais simples que a MainBom. Recebe como argumento apenas o *template* e invoca o TemplateParser para extrair os dados do *template*, não realizando nenhum tratamento de dados após a extração, sendo o próprio Núcleo que, conforme o mapeamento de atributos realizado utiliza os dados necessários.

- TemplateParser - De forma semelhante ao TableParser, extrai os materiais e organiza-os numa estrutura que exporta no final, posteriormente tratada pelo Núcleo.

5.1.2 Componente Núcleo

O Núcleo é responsável por gerir e interligar todos os componentes da aplicação de Desktop, assim como interagir com a aplicação de Servidor. A sua arquitetura é ilustrada na Figura 22.

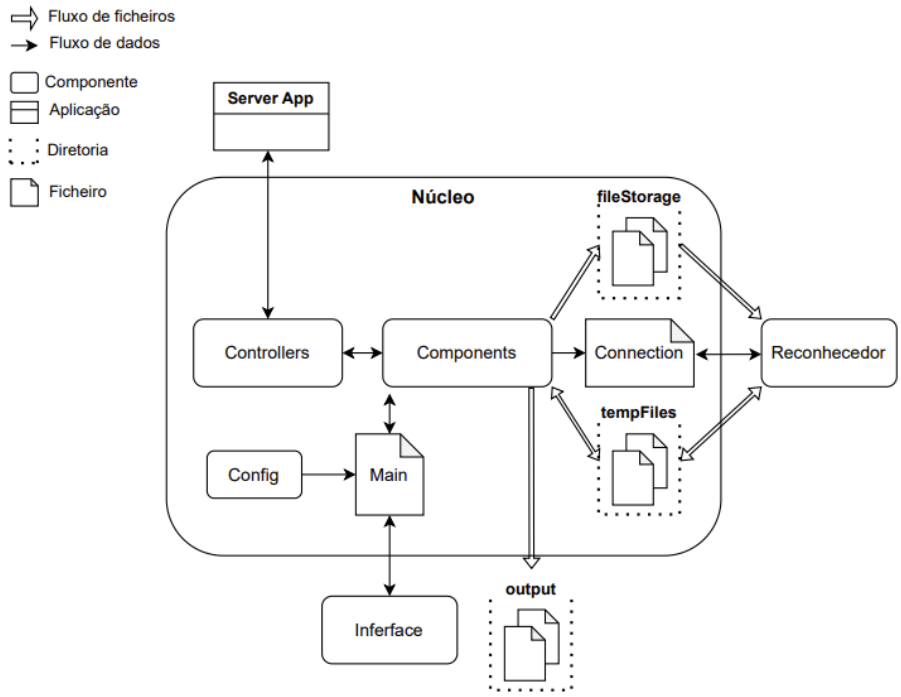


Figura 22: Arquitetura do Núcleo.

A tecnologia escolhida para desenvolver o Núcleo foi o *Electron*, portanto, toda a execução inicia pela Main, responsável por: criar o processo principal da aplicação; interagir com o sistema operativo; executar a lógica de negócio desenvolvida; gerir o processo de renderização responsável pela execução do componente Interface.

A Main utiliza um ficheiro de configuração (Config) onde se encontra especificado o endereço e porta para interação com a aplicação de Servidor. A Main interage com os Components, invocando o método do módulo correspondente ao componente específico para tratamento das solicitações efetuadas na Interface. Estes métodos, por sua vez, podem precisar de informações complementares e, para isso, recorrem aos Controllers para as obter. Os Controllers efetuam os pedidos à aplicação de Servidor.

Todos os Components recorrem ao módulo Connection para invocarem o Reconhecedor quando precisam de extrair informação de algum ficheiro. Para isso, os ficheiros a reconhecer são colocados na diretoria *fileStorage* e os resultados recebidos na diretoria *tempFiles*. O **BoM** após gerado é colocado na diretoria *output*.

Os módulos dos Components que possuem a lógica da aplicação de Desktop ilustram-se na Figura 23.

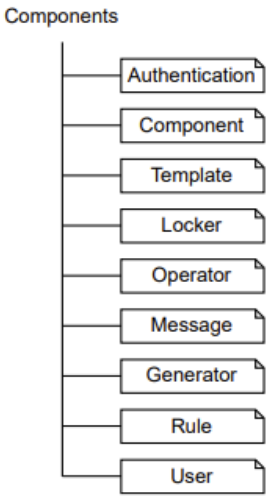


Figura 23: Componentes.

Abstratamente, a descrição da funcionalidade geral de cada módulo:

- Authentication - Responsável pelas funcionalidades de autenticação e estabelecimento de sessão da aplicação, como *login*, *logout* e registo.
- Operator - Responsável pelas funcionalidades relativas à gestão das operadoras, como a criação e remoção de operadoras. Assim como o carregamento dos materiais que podem ser utilizados na elaboração das regras.
- Component - Responsável pelas funcionalidades de configuração de componentes de rede (existentes nos ficheiros de projeto).
- Template - Responsável pela gestão dos *templates* e materiais das operadoras.
- Rule - Responsável pela gestão das regras das operadoras.
- User - Responsável pela gestão dos utilizadores.
- Locker - Responsável pela verificação da possibilidade de acesso a funcionalidades que não podem ser acedidas por mais do que um utilizador em simultâneo.
- Generator - Responsável pelas funcionalidades de geração do **BoM**, como receber os ficheiros de projeto, consumir o seu resultado, aplicar regras e exportar o **BoM** segundo o *template* pretendido.

- Message - Responsável pelo armazenamento da referência para a janela principal do processo de renderização. Desta forma, este módulo pode ser utilizado dentro de outros módulos para efetuar o envio de mensagens espontâneas do *BackEnd* para o *FrontEnd*.

Os Controllers, como dito, implementam os métodos responsáveis pela interação com a aplicação de Servidor e são constituídos pelos módulos apresentados na Figura 24.

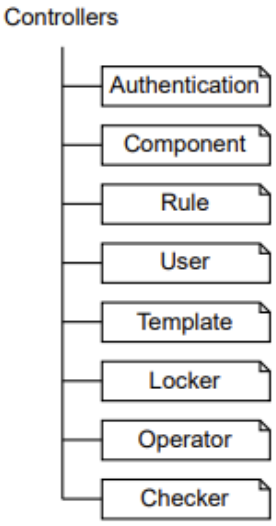


Figura 24: Controllers.

Todos os módulos objetivam na realização de pedidos **HTTP** à aplicação de Servidor, possuindo funcionalidades semelhantes, apenas agrupados conforme os seus contextos de utilização, à exceção do módulo Checker que é um *middleware* utilizado por todos os outros, na verificação das respostas recebidas da aplicação de Servidor.

Interação com o Reconhecedor

O Reconhecedor é invocado pelo módulo Conection que, possuí métodos para fazer a invocação dos respetivos pontos de entrada do Reconhecedor (MainBoM e MainTemplate), como ilustrado na Figura 25.

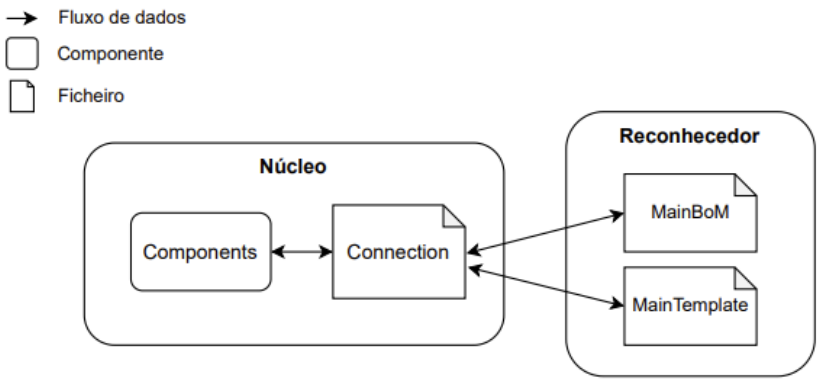


Figura 25: Conexão do Núcleo ao Reconhecedor.

Qualquer um dos Components pode invocar o Reconhecedor quando é necessário extrair informação de algum ficheiro, ocorrendo em dois casos: o primeiro, feito pelo módulo Generator quando é necessário gerar um **BoM**, o segundo, ao inserir um *template* **BoM** a uma operadora.

Interação com a aplicação de Servidor

Todas as interações com a aplicação de Servidor são realizadas através dos Controllers. Todas as respostas do servidor, antes de serem consumidas, são verificadas pelo módulo Checker que confirma se e resposta foi obtida tal como esperado, caso contrário, ajusta a Interface de acordo. Esta verificação é efetuada recorrendo ao código de status do cabeçalho das respostas e conforme o código, efetuado o procedimento adequado.

Interação com a Interface

A interação com a Interface é feita recorrendo ao mecanismo do *Electron*, ipcMain, que coloca o processo principal à escuta por pedidos em canais específicos, pré-definidos para invocar o método de tratamento correspondente. O retorno dos métodos de tratamento é diretamente enviado como resposta para a Interface.

Geração do BoM

Antes da geração do **BoM**, são especificados: a operadora; o *template* **BoM**; o tipo de **BoM** e o tipo de rede; e selecionados os documentos de projeto, sinótico, mapa e tabela de alocação. Após tal, e como dito anteriormente, a geração do **BoM** procede-se em duas fases, a extração da informação dos ficheiros (mapa, sinótico e tabela de alocação) e a aplicação das regras.

A extração da informação dos ficheiros inicia-se pela sua receção e armazenamento na diretoria *fileStorage*. Após tal, é pedido ao Servidor a gestão de componentes efetuada para a operadora, guardado o resultado num ficheiro especifico na diretoria *tempFiles* e então invocado o Reconhecedor. Após o término deste, é possível proceder à aplicação das regras, portanto, a segunda fase. Para tal, é pedido ao Servidor a gestão de regras efetuada para a operadora e importado o resultado do Reconhecedor para uma estrutura de dados interna ao Núcleo.

Esta estrutura de dados contém todo o conteúdo resultante do Reconhecedor e, portanto, contém toda a informação necessária para a geração do **BoM**. Assim sendo, para cada elemento da estrutura é verificada a existência de regras que possam ser aplicadas e, caso haja, portanto, caso as características do componente de rede coincidam com as características de aplicação da regra, são aplicados os materiais especificados na regra. Esta aplicação consiste na associação dos materiais a um multiplicador que especifica a quantidade de componentes iguais. Por exemplo:

- *Spliters* - Cada junta/**PLC** possui a lista com os seus *spliters*, constituídos pelo seu tipo e a sua respetiva quantidade. Desta forma, significa que a quantidade de material necessária calcula-se pela quantidade especificada na regra a aplicar sobre o *spliter* do tipo, multiplicada pela quantidade de *spliters* necessários. O mesmo acontece para as fusões.

- Cabo - Um cabo está obrigatoriamente associado a uma junta ou **PLC**, o que significa em antemão que só existe um cabo e, portanto, que os materiais a serem aplicados sobre este só o sejam uma vez. No entanto, existe uma característica diferenciadora na aplicação das regras aos cabos, a aplicação ao metro, que consiste em aplicar sobre o cabo, os materiais especificados da regra, de x em x metros, consequentemente, a quantidade de material necessária, calcula-se pela multiplicação da quantidade especificada na regra pelo número de metros total dividido pelo x.

- Junta - Cada junta é representada num elemento da estrutura de dados, logo, a quantidade de material necessária calcula-se pela multiplicação da quantidade especificada na regra por 1. O mesmo acontece para os **PLC**'s, tubos de subida, postes e valas.

Após o cálculo das quantidades é pedido ao servidor o *template* **BoM** e o mapeamento de atributos correspondente, para ser preenchida a coluna especifica para as quantidades de material necessárias e, por fim, exportado o **BoM** para a diretoria *output*.

Procedimento de autenticação

A utilização da aplicação requer autenticação obrigatória, caso contrário, não é possível aceder às suas funcionalidades. Como dito anteriormente, a tecnologia utilizada para efetuar a autenticação foi o *Json Web Token*, portanto, ao efetuar o *login*, ou o registo na primeira utilização, é criado um *token* na aplicação de Servidor e enviado para a aplicação de Desktop para ser utilizado em todos os posteriores pedidos à aplicação de Servidor. Assim, esta saberá se tem ou não autorização para lhes responder e mesmo se o utilizador tem privilégio para aceder ao que solicita.

O *token* quando recebido pela aplicação de Desktop é colocado numa variável global que constitui o cabeçalho de todos os pedidos efetuados pelos Controllers. Quando efetuado o *logout*, o *token* é eliminado, obrigando a efetuar novamente o *login*.

Controlo de concorrência

Funcionalidades que modificam dados guardados na base de dados têm de ser restritas a um só utilizador em simultâneo, para mitigar a criação de inconsistências. Desta forma, o acesso a uma operadora é exclusivo a um utilizador, ou seja, sempre que um utilizador aceder a uma operadora, essa operadora é bloqueada para escrita para qualquer outro utilizador, sendo apenas possibilitada a leitura, ou seja, apenas a geração de **BoM**'s.

Como dito, este bloqueio é efetuado no momento do acesso a uma operadora, portanto, no momento em que o utilizador tenta aceder a uma operadora, é verificada a disponibilidade dessa operadora e se possível aceder, a correspondente é bloqueada para utilizador, caso contrário, é informado que a operadora a que acede está bloqueada por outro utilizador.

Para esta funcionalidade funcionar corretamente, as *interfaces* estão organizadas de acordo com uma hierarquia especifica, de modo a ser enviado à aplicação de Servidor o pedido de desbloqueio da operadora quando acedida uma funcionalidade que não exija o seu prévio bloqueio. Mais detalhes serão explicados na próxima secção.

Este mecanismo cria dois problemas identificados, o bloqueio permanente de uma operadora por um utilizador, e o *login* feito em mais do que uma aplicação de Desktop pelo mesmo utilizador.

O primeiro problema surge quando um utilizador bloqueia uma operadora e por algum motivo de erro (exemplo: desconexão da Internet) não a desbloqueia, pois nas condições normais, a aplicação no seu término envia à aplicação de Servidor o pedido de desbloqueio da operadora. Numa situação de erro, a operadora manter-se-ia bloqueada infinitamente, logo, para evitar esta ocorrência, foi definido um limite de temporal a que uma operadora pode ficar bloqueada por um utilizador, ultrapassado esse limite, a

operadora é desbloqueada.

O segundo problema surge quando o mesmo utilizador efetua *login* em duas aplicações de Desktop distintas, o que possibilitaria a modificação da mesma operadora em simultâneo. Para evitar esta ocorrência foi definida uma lista de utilizadores autenticados, verificada sempre que um utilizador faz *login*, de modo, a impedir o *login* de um utilizador que já esteja autenticado. Esta solução cria um problema tal como o anterior, ou seja, a autenticação permanente caso um utilizador autenticado por algum motivo de erro (exemplo: desconexão da Internet) não faça *logout*, pois a aplicação no seu término envia à aplicação de Servidor o pedido de *logout*. Para resolver este caso foi implementada a mesma solução anterior.

5.1.3 Componente Interface

Este componente foi desenvolvido aquando das funcionalidades do Núcleo, portanto, sempre que desenvolvida uma funcionalidade no Núcleo, a correspondente Interface para a sua utilização também foi desenvolvida. Apenas após desenvolvidas todas as funcionalidades, foram desenvolvidos os aspetos não funcionais como melhoramentos de usabilidade e aparência.

A tecnologia utilizada foi o *React*, portanto a Interface é uma aplicação *FrontEnd* independente que corre num processo dedicado à renderização e utiliza o *ipcRenderer* do *Electron* para efetuar pedidos ao próprio *BackEnd*. A arquitetura da Interface ilustram-se na Figura 23.

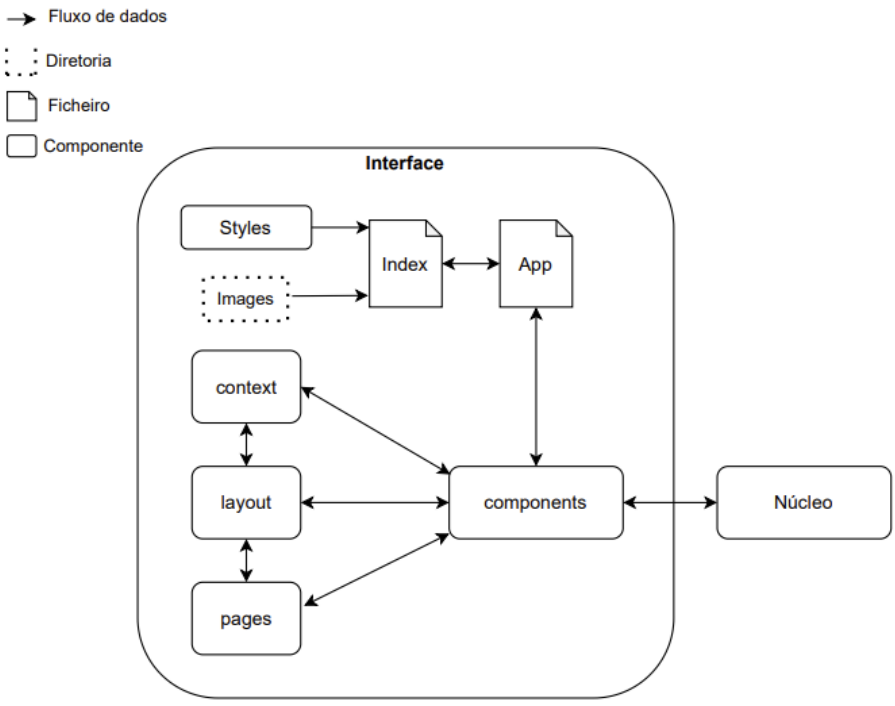


Figura 26: Arquitetura da interface.

Tal como numa aplicação *React* comum, a execução inicia pelo módulo Index, onde o módulo App é incorporado, juntamente dos módulos do componente Styles e imagens a utilizar. O módulo App é a raiz da aplicação, onde estão incorporados todos os módulos dos Components e respetivas rotas de acesso. No Index são também especificados os contextos de sessão e bloqueio que, devem ser renderizados antes da App em si, pois, fornecerão informações ao próprio módulo App. De forma semelhante acontece no módulo App, onde é especificado o contexto de histórico, mais uma vez, para renderizar antes de qualquer outro módulo, pois o histórico consiste na receção de mensagens do *BackEnd* que, podem ser recebidas em qualquer um dos módulos, a qualquer momento. Qualquer um dos Components pode efetuar pedidos ao Núcleo e encontram-se agrupados em três sub-componentes: Context, Layout e Page.

O Context possui os módulos que fornecem contexto aos módulos do sub-componente Page, como: autenticação, mensagens recebidas do Núcleo e situação de bloqueio de operadoras.

O Layout possui módulos que otimizam a construção das páginas da Interface, portanto, aspetos comuns a todas as páginas modularizados, como o cabeçalho e o rodapé.

O Page possui todas as páginas da Interface, pelo que, devido ao elevado número delas, encontra-se também dividido em sub-componentes conforme o seu contexto, tal como ilustra a Figura 27.

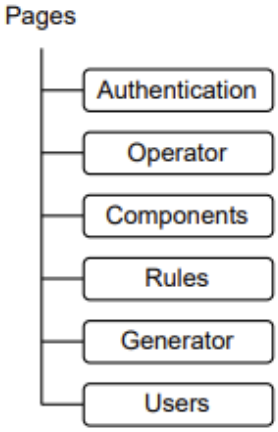


Figura 27: Sub-componentes do Page.

Cada sub-componente possui o conjunto dos módulos que constituem as páginas da Interface agrupadas pelo seu contexto de utilização. Todas elas possuem alguma semelhança quanto à sua implementação, sendo constituídas por:

- Variáveis - Utilizadas para preencher o conteúdo da página a ser apresentada, e também, para guardar informação inserida pelo utilizador nos formulários.
- UseEffets - Utilizados para disparar os pedidos de informação ao Núcleo e inserir nas respetivas

variáveis. Estes métodos executam maioritariamente uma única vez no momento em que a página é renderizada, salvo exceções em que executam devido a alguma modificação efetuada sobre uma variável.

- Métodos - Implementam funcionalidades para botões de submissão, remoção e seleção, formulários de pesquisa, entre outras.
- **JSX** - Implementa a estrutura da página, preenchendo-a quando necessário com informação das variáveis.

O componente Style possui os módulos que fornecem as características de estilísticas da Interface e a diretoria Images as imagens que dela fazem parte.

É importante denotar que a Interface foi o componente da aplicação com menor prioridade durante o desenvolvimento, pelo que, ainda existe muito desenvolvimento a poder ser feito sobre a mesma, como otimizações para melhor manutenibilidade da mesma, assim como exploração das potencialidades da tecnologia selecionada. Durante o seu desenvolvimento, para tornar cada módulo o mais independente possível, aspetos comuns não foram modularizados, devido à realização de tal tarefa, requerer bastante tempo e certeza de que estes aspetos não se mudariam a curto prazo. Assim, o desenvolvimento de otimizações na Interface foram consideradas de mínima prioridade e mais detalhes sobre o trabalho futuro serão apresentados no Capítulo 7.

Procedimento de autenticação

Quando a aplicação inicia é imediatamente apresentada a página com as funcionalidades de autenticação para que o utilizador a efetue. Após efetuada, ou seja, após obtido o *token* da aplicação de Servidor, o utilizador passa a estar autenticado e todas as páginas e correspondentes funcionalidades são liberadas, permitindo o seu acesso. A permissão do seu acesso é sempre verificada antes de ser apresentada a página a que se acede, consistindo em verificar se o utilizador continua autenticado, pois, por exemplo: o tempo de *login* poderá ter sido excedido, disparando o *logout* automático do utilizador na aplicação de Servidor. Se a verificação falhar, portanto, se por algum motivo o utilizador deixar de estar autenticado, a Interface carrega automaticamente a página com as funcionalidades de autenticação e impede o acesso a qualquer outra.

Controlo de concorrência

Para facilitar o impedimento do acesso às páginas com funcionalidades que manipulam dados de operadoras, foi importante organizar a Interface de uma forma hierárquica, de modo a obrigar o utilizador a obter o bloqueio da operadora a que acede, antes de poder aceder à página da operadora, onde constam as funcionalidades de manipulação de dados. De forma semelhante ao procedimento anterior, se por algum motivo o utilizador perder o bloqueio da operadora, é automaticamente carregada a página do topo de hierarquia, onde consta a lista com todas as operadoras, para este poder selecionar novamente a operadora pretendida e obter o bloqueio da mesma se esta estiver livre.

Reações a mensagens do Núcleo

A Interface para receber as mensagens do Núcleo, colocar-se à escuta num canal específico para tal, no qual o Núcleo colocará as mensagens, para isso, utilizado o *ipcRenderer*. Como qualquer uma das páginas pode gerar mensagens no Núcleo, para mitigar a repetição de código, foi utilizado o mecanismo de contextos fornecido pelo *React*, de modo a possibilitar guardar as mensagens (constituindo o histórico de procedimentos efetuados e de erros surgidos) e também reagir conforme o recebido, uma vez que existem três tipos de mensagens:

- Informação - Objetivam em informar se a funcionalidade utilizada foi ou não concluída com êxito.
- Erro - Tal igual à anterior, mas para casos de erro ou má utilização.
- Redirecionamento - Este tipo de mensagens são as que exigem reatividade imediata por parte da Interface, constituindo uma prevenção para a utilização de determinadas funcionalidades impedidas de serem utilizadas. São utilizadas nas seguintes situações:
  - Suponhamos que foi efetuado pela aplicação de Servidor o *logout* do utilizador por inatividade. Acontece que a aplicação de Desktop apenas saberá que o utilizador não estará autenticado assim que for utilizada uma funcionalidade que necessite de realizar um pedido à aplicação de Servidor, sendo o Núcleo o componente a ser informado. Desta forma, para evitar que o utilizador continue a ter acesso à Interface como um utilizador autenticado, é enviada uma mensagem do Núcleo para a Interface, para que esta efetue o redirecionamento para a página de autenticação e bloqueie novamente as funcionalidades.
  - De forma análoga acontece o mesmo processo para o bloqueio das operadoras.

5.1.4 Implantação

A implantação da aplicação é feita recorrendo ao *electron packager*, resultando na criação um pacote com todo o código e suas dependências. O pacote inclui um ficheiro executável que serve de ponto de entrada para a aplicação e que deve ser utilizado pelo utilizador para executar a mesma. O pacote é transferido na sua totalidade para a máquina do utilizador por meio de um **Zip**.

5.2 Aplicação de Servidor

O desenvolvimento da aplicação de servidor iniciou-se com o surgimento da necessidade de guardar dados, tendo incrementalmente evoluído conforme as funcionalidades implementadas na aplicação de Desktop. É composta por dois componentes, a API e a base de dados.

5.2.1 Componente API

A API, responsável por tratar os pedidos efetuados pelas aplicações de Desktop, possui a sua arquitetura conforme a Figura 28.

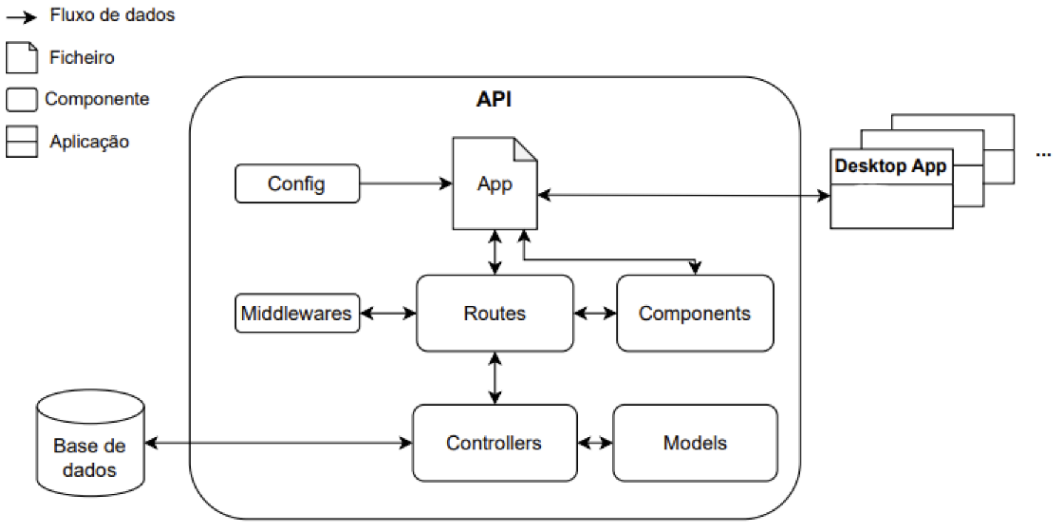


Figura 28: Arquitetura da API.

Uma vez escolhido o *ExpressJS* para implementar a API, a sua execução inicia-se pelo módulo App, responsável por: invocar o módulo para estabelecimento da ligação à base de dados; configurar as rotas para os respetivos módulos do componente Routes; e por iniciar a escuta por pedidos na porta especificada.

Os pedidos efetuados pelas aplicações de Desktop ao chegarem à App são redirecionados para tratamento conforme o nome da rota e método **HTTP** utilizado no pedido. Como dito, o tratamento é efetuado pelos módulos internos ao componente Routes, ilustrados na Figura 29.

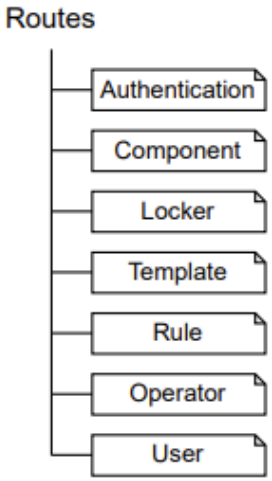


Figura 29: Módulos do Routes.

Cada módulo tem a sua responsabilidade, sendo:

- Authentication - Tratamento dos pedidos de *login*, registo e *logout*.
- Component - Tratamento dos pedidos relativos à gestão de componentes.
- Locker - Tratamento dos pedidos relativos ao controlo de concorrência.
- Template - Tratamento dos pedidos relativos à gestão de *templates*.
- Rule - Tratamento dos pedidos relativos à gestão das regras.
- Operator - Tratamento dos pedidos relativos à gestão de operadoras.
- User - Tratamento dos pedidos relativos à gestão de utilizadores.

De forma análoga à aplicação de Desktop, todos os módulos recorrem aos Controllers para efetuar as leituras e escritas na base de dados.

Apenas após ter sido desenvolvida uma API capaz de suportar todas as funcionalidades da aplicação de Desktop foram implementados os mecanismos de autenticação, controlo de concorrência, além de protegidas as rotas necessárias de acordo, recorrendo a *middlewares* que executam e verificam se o pedido pode ou não prosseguir para tratamento.

Lógica de autenticação

A autenticação é utilizada como mecanismo de segurança para o acesso aos dados guardados na base de dados, sendo garantida pela utilização de um *middleware* que é executado sempre que é acedida alguma das rotas disponibilizadas, à exceção das rotas de Registo e *login*. Este *middleware* verifica se o acesso está a ser feito por um utilizador autenticado, portanto, com *login* efetuado através da aplicação de Desktop.

- Registo do utilizador - Um registo é constituído pelo nome de utilizador, palavra-chave e privilégio de acesso. No momento do registo a aplicação de Servidor verifica se o nome de utilizador é válido (não existe outro registo com o mesmo nome), caso seja, verifica se está perante um registo de um utilizador administrador ou regular, validando no caso do administrador, se a palavra-chave de permissão de registo está correta. Após estes passos, prossegue com a encriptação da palavra-chave e posterior inserção na base de dados. Por fim, efetua o *login* automático, gerando o *token* com a identificação do utilizador gerada pela base de dados, iniciando o temporizador para o *logout* automático e enviando o *token* à aplicação de Desktop.

- *Login* do utilizador - O *login* verifica a existência do utilizador com o nome utilizado, caso exista, compara a sua palavra-chave com a utilizada e se coincidir, inicia o temporizador para *logout* automático, gera o *token* e envia à aplicação de Desktop.

- Temporizador de *logout* - Este temporizador não é definido pelo tempo de expiração atribuído ao *token* e sim pela definição de um temporizador para remoção do utilizador da lista de utilizadores autenticados. Desta forma, o *middleware* na verificação da autenticação observará que o utilizador não está autenticado apesar de possuir um *token* válido. Este temporizador é renovado sempre que é feita uma interação com o servidor.

Controlo de concorrência

Mesmo que um utilizador malicioso tente aceder forçadamente às funcionalidades de uma operadora, será impossibilitado, pois todas as rotas que suportam estas funcionalidades são bloqueadas para o utilizador detentor do bloqueio da operadora. Este bloqueio é efetuado por um *middleware* que verifica se o nome do utilizador detentor do bloqueio é o mesmo do utilizador que acede às suas funcionalidades, caso não o seja a funcionalidade é barrada.

- Bloqueio de operadora - O bloqueio é efetuado explicitamente quando alguma aplicação de Desktop

solicita o acesso a uma operadora. Nesse momento, é verifica a disponibilidade da operadora e confirmada a inexistência de alguma operadora já bloqueada por este, só então sendo bloqueada para o utilizador solicitador. Nesse momento é também definido o temporizador de bloqueio.

- Temporizador de bloqueio - Quando uma operadora fica bloqueada para um certo utilizador, esse bloqueio é temporário e renovado conforme o utilizador efetua modificações sobre a operadora. Caso o utilizador bloqueie a operadora e não efetue modificações na mesma, ao fim de um intervalo de tempo a operadora é desbloqueada, obrigando o utilizador a voltar a obter o bloqueio da mesma.

Este controlo de concorrência é efetuado nas funcionalidades que modificam os dados das operadoras, ou seja, qualquer funcionalidade que apenas consulte os dados da operadora, não necessita do bloqueio da operadora.

5.2.2 Componente Base de Dados

Este componente foi implementado recorrendo ao *MongoDB*, portanto, utilizando um paradigma **NoSQL** orientado a documentos, crucial para lidar com as diversas modificações que foram surgindo ao longo do desenvolvimento da aplicação de forma rápida e precisa, devido às suas vantagens.

O **Schema** final da base de dados possui duas entidades principais, a operadora e o utilizador. O utilizador apenas armazena os dados de autenticação dos utilizadores, já a operadora armazena todos os dados relacionados às operadoras, pelo que é mais complexa. A figura 30, representa abstratamente a estrutura dos dados das operadoras.

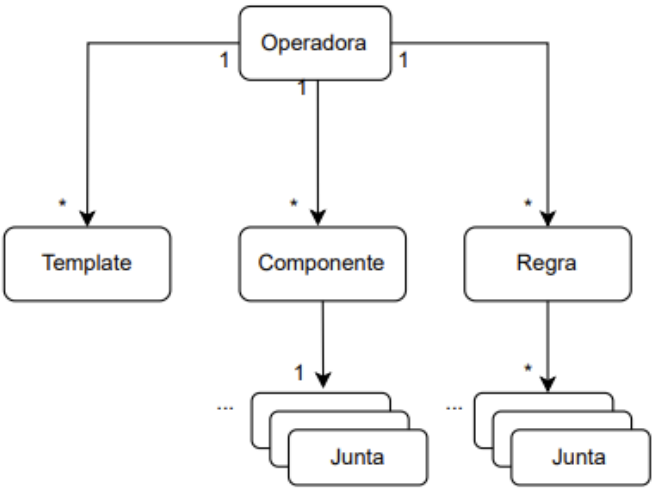


Figura 30: Estrutura dos dados da entidade operadora.

A operadora é constituída pelas três entidades seguintes:

- Template - Coleção de documentos com informações relativas aos *templates* introduzidos na operadora em questão. Cada documento representa um *template* e possui informações como: a data do *template*; o mapeamento dos atributos pré-definidos na aplicação com os respetivos nomes das colunas do *template*; a lista com todos os materiais possíveis de utilizar nas regras; e o nome atribuído ao correspondente ficheiro do *template*, guardado no sistema de arquivos.

- Componente - Coleção de documentos pré-definidos, que podem ser: Junta, Plc, Poste, Cabo, Area, Spliter, Tabela de alocação, Tubo e Vala. Cada um destes documentos armazenam dados como: o mapeamento entre atributos pré-definidos requeridos pela aplicação para reconhecimento da informação dos documentos e respetivos nomes utilizados pela operadora, assim como outras características adicionais mencionadas no final da secção 5.1.1.

- Regra - Coleção de documentos também pré-definidos, que podem ser: Junta, Plc, Poste, Cabo, Area, Spliter, Tabela de alocação, Tubo e Vala. Cada um destes documentos possui as características do componente no qual a regra se aplica, além da lista de sub-regras com os respetivos materiais e quantidades a serem aplicadas.

5.2.3 Implantação

A sua implantação no servidor é feita com recurso ao *Docker* e ao *Docker Compose*, tal como previsto. Após ser criada e publicada no *DockerHub* a imagem da API, bastará utilizar o *Docker Compose* para implantar e executar automaticamente a aplicação de Servidor.

Capítulo 6

Testes e Resultados

Durante o desenvolvimento de toda a aplicação utilizou-se um projeto de rede especifico de uma operadora específica. Só em fase avançadas se iniciou a utilização de outros projetos efetuados para a mesma operadora, iniciando-se também o desenvolvimento do Reconhecedor para uma nova operadora. Desta forma, a maioria do desenvolvimento foi feito tendo como base o projeto selecionado, pelo que, em seguida abordarei com principal ênfase os testes e resultados efetuados sobre este.

6.1 Reconhecimento dos ficheiros de projeto

O funcionamento correto da aplicação requer a configuração dos componentes, que engloba o mecanismo implementado para permitir a variabilidade de nomes além de outros aspetos abordados anteriormente, como, por exemplo: identificação dos componentes a não construir, identificação de redes, etc. Esta configuração possibilita o correto funcionamento do Reconhecedor independentemente dos nomes atribuídos e é obrigatória de ser feita antes da utilização do Gerador **BoM**.

Como dito anteriormente, o Reconhecedor foi desenvolvido com foco nos ficheiros de projeto de uma operadora particular, pelo que, na sua fase inicial todos os atributos necessários de serem reconhecidos já eram conhecidos, devido à prévia análise da estrutura dos documentos. Estes atributos foram colocados diretamente no código do Reconhecedor e então desenvolvida a lógica do Reconhecedor para extrair a informação útil destes.

O Reconhecedor foi assim testado no projeto fornecido e melhorado incrementalmente até reconhecer toda a informação necessária para a decisão dos materiais a serem aplicados. A confirmação de que todos os componentes eram reconhecidos foi realizada manualmente, comparativamente lado a lado com os documentos de projeto, revelando-se um processo demorado e exigente.

Por diversas razões, colocar os atributos a reconhecer diretamente no código e não permitir a sua variabilidade tornar-se-ia uma solução inviável a longo prazo, portanto, construiu-se o mecanismo capaz de

permitir variabilidade no nome dado a estes atributos possibilitando a sua variabilidade futura, apesar do seu significado para o Reconhecedor ser o mesmo. Foi então que surgiu o conceito falado anteriormente, a configuração de componentes.

A configuração de componentes pré-define um conjunto de variáveis (atributos) necessárias de serem reconhecidas dos ficheiros, dando ao utilizador a possibilidade de modificar o nome atribuído. Após implementação deste mecanismo, procedeu-se novamente aos testes do Reconhecedor até obter a garantia de que os resultados coincidiam com a prévia versão.

6.2 Criação e aplicação das regras

A configuração das regras engloba a seleção dos componentes do projeto e quais os materiais a serem aplicados sobre estes. Portanto, foi definitivamente a funcionalidade mais exigente de ser testada, uma vez que exigia saber precisamente quais as regras de construção do **BoM** e comparar lado a lado **BoM**'s gerados automaticamente com **BoM**'s realizados manualmente.

Devido à inexistência de documentação que especifique as regras, foram criadas versões beta da aplicação e passadas aos projetistas da própria empresa de modo a testarem a capacidade desta em fornecer os mecanismos pretendidos para a criação de regras. Conforme as necessidades dos projetistas foram implementados e melhorados os mecanismos de criação e aplicação das regras.

6.3 Autenticação e concorrência

Estes dois mecanismos foram implementados já numa fase avançada, de modo a estabelecer medidas de segurança no acesso aos dados e também prevenir a realização de inconsistências devido à manipulação dos mesmos por mais do que um utilizador.

6.4 Páginas da Interface

As *interfaces* desenvolvidas para fornecer as funcionalidades ao utilizador serão a seguir apresentadas, para um utilizador detentor do privilégio de administrador e, portanto, com acesso a todas as funcionalidades.

Imediatamente após iniciada a aplicação de Desktop é apresentada a página da Figura 31, onde são disponibilizadas as funcionalidades de *login*, registo e histórico que, nesta altura, não possuí informações.

Figura 31: Página de autenticação.

Após realizado o registo ou *login*, a funcionalidade da página é automaticamente modificada para fornecer o *logout* quando acedida e o utilizador reencaminhado para a página principal, apresentada na Figura 32.

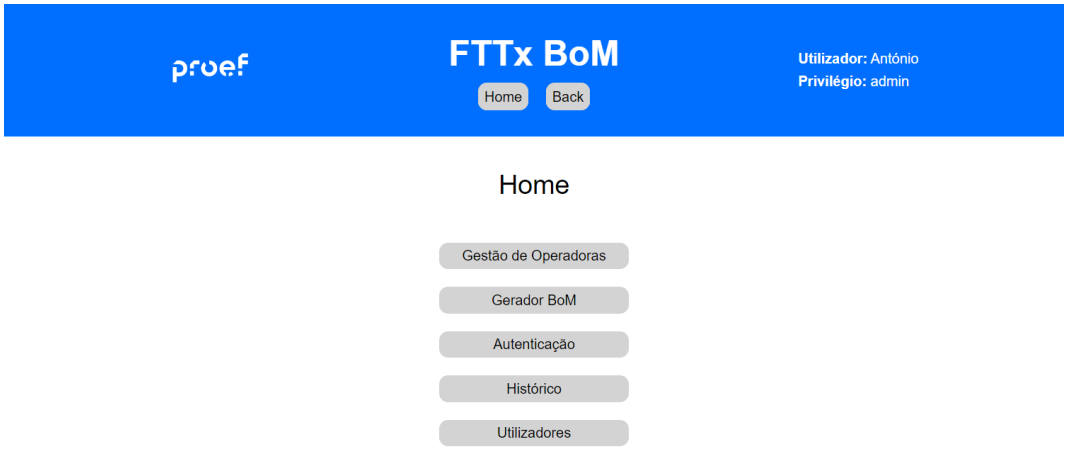


Figura 32: Página principal.

Esta não apresenta nenhuma funcionalidade, apenas agrupa as rotas para as funcionalidades numa só página, dos quais:

- Gestão de operadoras - Possui as funcionalidades relacionadas às operadoras.
- Gerador BoM - Gerador do **BoM**.

- Autenticação - Possui todas as funcionalidades relacionadas à autenticação, neste caso, como o utilizador já está autenticado é-lhe apresentado o *logout*.
- Histório - Possui o registo das mensagens recebidas do Núcleo.
- Utilizadores - Possui as funcionalidades relacionadas aos utilizadores.

A gestão de operadoras possui a página da Figura 33. Nela são listadas as operadoras registadas no sistema, juntamente de uma barra de pesquisa e de um botão para adição de novas operadoras. É precisamente nesta página que, quando acedida uma operadora é verificada a possibilidade de acesso, ou seja, verificado a possibilidade de bloqueio da mesma e caso seja, apresentada a página da operadora.

Figura 33: Página da gestão de operadoras.

Se pretendida a adição de uma nova operadora a página da Figura 34 é apresentada.

Figura 34: Página da adição de operadoras.

Aqui o utilizador escolhe o nome a atribuir à operadora e seleciona o seu modelo, à data apenas disponível o modelo para a operadora base deste projeto (encontra-se coberta a azul por razões de privacidade

impostas pela Proef). Este modelo é, por outras palavras, a identificação do Reconhecedor que deve ser usado para tratar os documentos da respetiva operadora. Como ainda só existe um Reconhecedor, apenas existe um modelo de operadora.

Referindo novamente a Figura 33, supondo que foi permitido o acesso à operadora_1, é imediatamente carregada a página seguinte, apresentada na Figura 35.

Figura 35: Página da operadora.

Nesta página é possível aceder às funcionalidades que manipulam diretamente os dados da operadora, ou seja, os componentes de rede, os *templates* inseridos e as regras criadas, além da permitir apagar a operadora. Na gestão de componentes é apresentada a seguinte página, correspondente à Figura 36.

Figura 36: Página da gestão de componentes.

Cada um dos botões redireciona para uma página com a estrutura necessária de ser preenchida,

conforme o tipo de componente. Também é aqui feita a configuração da tabela de alocação, que apesar de não ser um componente de rede, é necessária para a geração do **BoM**. Na Figura 37, 38 e 39 são apresentadas as páginas já preenchidas, respetivamente para as juntas, os cabos e tabela de alocação.

Figura 37: Página da gestão de componentes: cabo.

Figura 38: Página da gestão de componentes: junta.

Figura 39: Página da gestão de componentes: tabela de alocação.

Referindo novamente a Figura 35, mencionando agora a gestão de *templates*, temos a seguinte página da Figura 40.

Figura 40: Página da gestão de *templates*.

Nesta página são listadas as datas dos *templates* inseridos, onde cada uma é um botão que redireciona para a página onde constam todos os seus materiais, sendo cada um, um botão, tal como apresenta a Figura 41. Além disso, é associado a cada data o respetivo botão para apagar. No topo encontra-se o botão para adicionar um novo *template* que, quando pressionado redireciona para a página da Figura 42. Na Figura 42 o formulário para adição já se encontra devidamente preenchido.

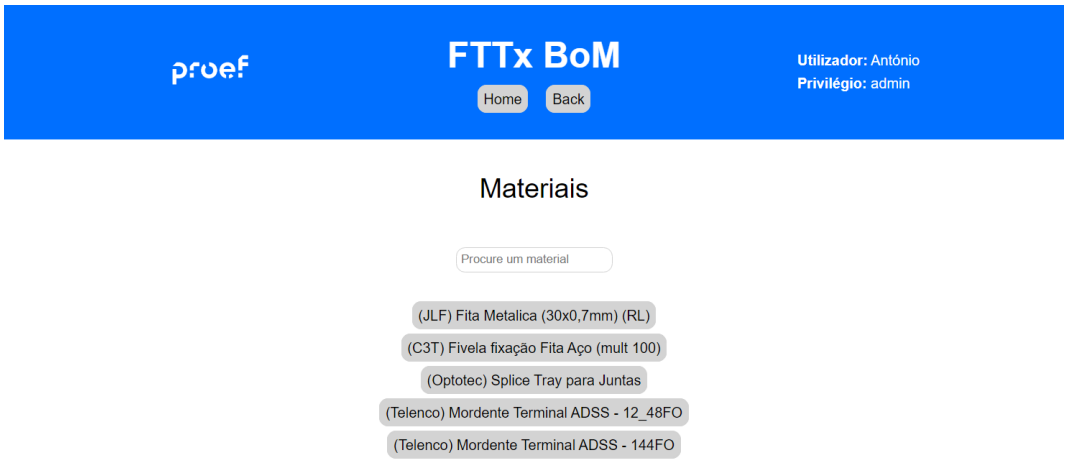


Figura 41: Página do *template*.

Cada um dos botões da página da Figura 41 redireciona para a página onde constam todas as regras que utilizam o respetivo material.

Figura 42: Página da adição de *template*.

O formulário com título "Mapeamento de atributos do template"apenas é apresentado após selecionado o *template* e pressionado o botão "processar template"da Figura 42. Este botão submeterá o *template* ao Núcleo de modo a este invocar o Reconhecedor para a extração de informação. Com o término da extração, o Núcleo retorna à Interface as colunas existentes no *template*, para esta preencher as opções das caixas de seleção. Desta forma é possível efetuar o mapeamento das colunas com o respetivo significado requerido.

Referindo novamente a Figura 35, mencionando agora a gestão das regras, temos a seguinte página

da Figura 43, nela listam-se todas as regras criadas, subdivididas conforme o componente de rede de aplicação. No caso apresentado, encontram-se criadas três regras, uma para juntas, outra para **PLC**'s e outra para cabos. Cada uma apresenta as respetivas caraterísticas do componente de rede ao qual devem ser aplicadas.

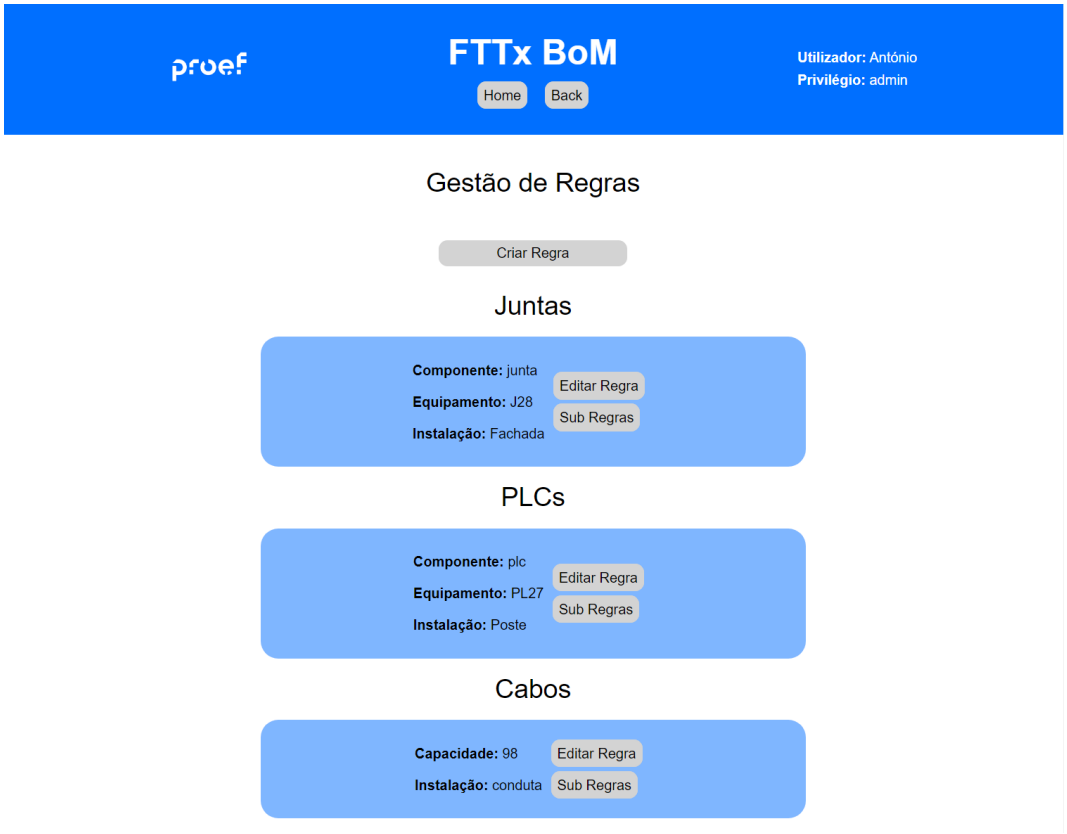


Figura 43: Página da gestão de regras.

Quando pressionado o botão "Criar Regra"é redirecionada uma página semelhante à da gestão de componentes, tal como a Figura 44, onde se apresentam todos os componentes de rede, cada um com uma especificação distinta. Como podemos observar na Figura 43, a especificação de uma junta (Componente, Equipamento, Instalação) difere da especificação de um cabo (Capacidade, Instalação). Cada regra criada (representada por um retângulo azul) possui sub-regras que especificam quais os materiais, respetivas quantidades e formas de aplicação, a serem aplicados sobre o componente de rede especificado pela regra, tal como na Figura 45.

Figura 44: Página da criação de regra.

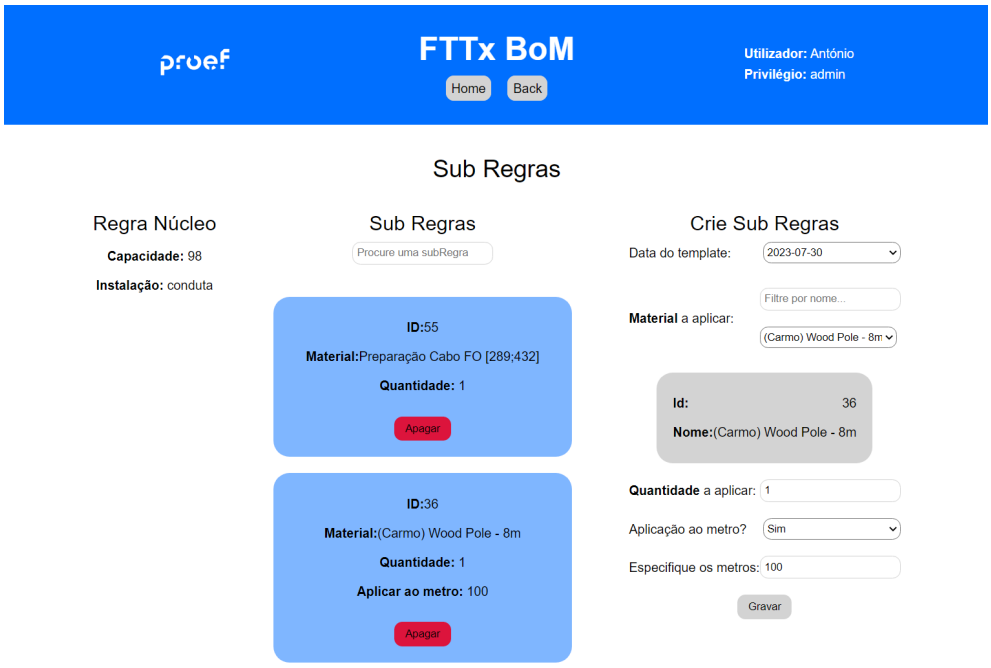


Figura 45: Página de sub-regras para o cabo.

Esta página (Figura 45) é composta por três colunas:

- Primeira - Apresenta a regra principal (regra núcleo), ou seja, qual o componente de rede onde as sub-regras vão ser aplicadas.
- Segunda - Lista as sub-regras criadas, ou seja, o material selecionado, a quantidade respetiva e a forma de aplicação, que poderá neste caso ser ou não ao metro, pois estamos perante uma regra principal que especifica um componente do tipo cabo. No caso apresentado, já existem duas sub-regras, uma que aplica a quantidade de 1 unidade de material com id 55 e outra que aplica a quantidade de 1 unidade de material com id 36 a cada 100 metros de cabo existente. Estas sub-regras serão aplicadas sempre que surja um cabo com capacidade 98 instalado em conduta.

- Terceira - Formulário para criação de sub-regras, onde é requisito obrigatório selecionar a data do *template* de materiais a utilizar, o material do respetivo, a quantidade a ser aplicada e a forma de aplicação.

Referindo novamente a Figura 32, resta gerador BoM, o histórico e os utilizadores. O primeiro, apresentado na Figura 46, tal como o nome indica, é o gerador para os **BoM**'s, possuindo o formulário para os três ficheiros de projeto (sinótico, mapa e tabela) e quatro caixas de seleção, para a operadora, data do *template*, tipo de **BoM** e tipo de Rede.

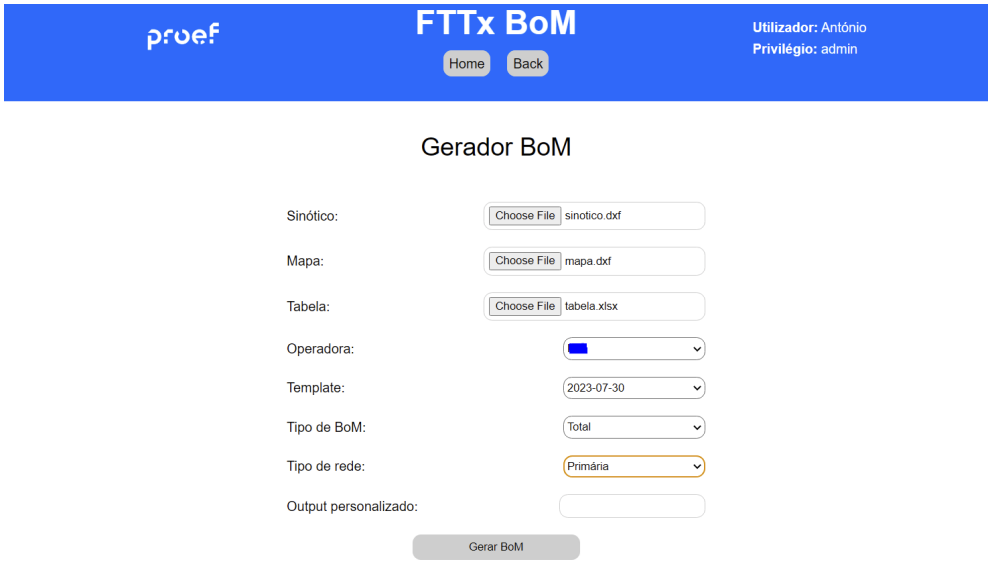


Figura 46: Página do gerador Bom.

O segundo é responsável por apresentar as mensagens recebidas do *BackEnd*. Na seguinte Figura 47 encontram-se as mensagens emitidas pelo *BackEnd* a um utilizador que se autenticou para gerar um **BoM**.

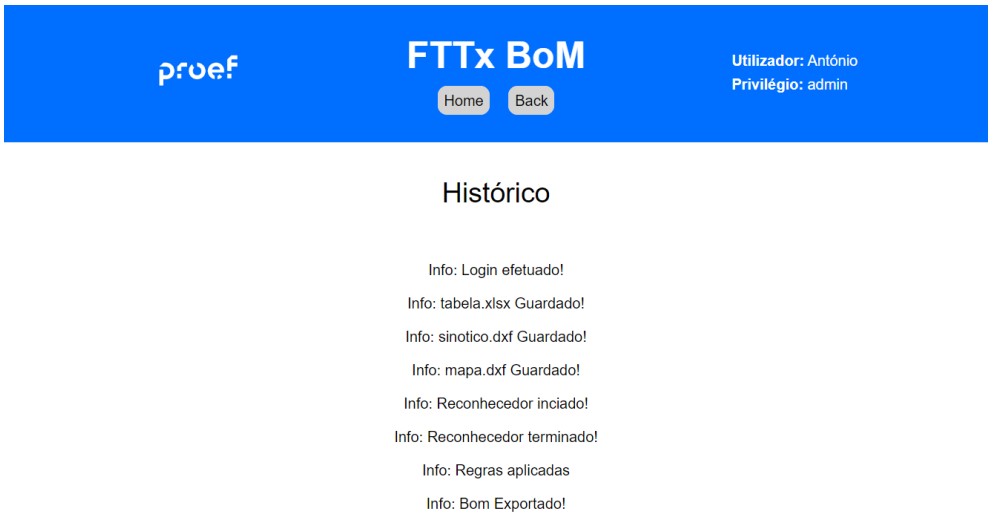


Figura 47: Página do histórico.

O terceiro é a gestão de utilizadores, portanto, a página onde estão listados todos os utilizadores registados na aplicação, associados ao seu respetivo botão de remoção, tal como a Figura 48.

Figura 48: Página da gestão de utilizadores.

Capítulo 7

Conclusões e trabalho futuro

Como conclusão desta dissertação é possível afirmar que o principal objetivo foi atingido, gerar **BoM**'s recorrendo à informação dos documentos de projeto e a regras definidas dinamicamente. Ao longo do desenvolvimento, os requisitos levantados inicialmente foram iterativamente modificados, resultando na adição de novas funcionalidades e consequentemente melhoramento da aplicação, até ao ponto em que, agora, a podemos chamar serviço *FTTx BoM*. Com a realização desta dissertação foram exploradas as técnicas de manipulação de ficheiros **DXF**, aprofundados os conhecimentos em *NodeJS*, especificamente nas *frameworks BackEnd*: *Express* e *Electron*; mas também *FrontEnd* com a biblioteca *React*, entre outros detalhes que resultaram no aumento da experiência em engenharia de *software*.

Ainda existe muito trabalho a ser realizado sobre o serviço *FTTx BoM*, desde melhoramentos de Interface, até melhoramentos como: da capacidade de reconhecimento dos ficheiros de projeto; dos mecanismos fornecidos para a criação de regras, de modo a permitir criar regras mais complexas; dos mecanismos de autenticação, uma vez que os implementados são consideravelmente simples, devido à aplicação ser pensada para uso interno à empresa; até à adição de novas operadoras com a criação do Reconhecedor correspondente e, possivelmente no futuro, a construção de um mecanismo apto para reconhecer os ficheiros de qualquer operadora, sem a sua prévia análise. Juntamente, devido à exigência dos testes manuais, o desenvolvimento de um sistema de testes automáticos para futuros Reconhecedores seria algo de alta importância.

Certamente o serviço criado apresenta elevado potencial, pois, para além de conseguir construir o **BoM** para qualquer projeto, calcula-o num intervalo de tempo insignificante, quando comparado ao intervalo de tempo requerido pela sua construção manual e ainda, garante elevadas taxas de precisão, permitindo às empresas utilizadoras do serviço poupar horas de trabalho, anteriormente, gastas por projetistas com a construção manual do **BoM**.

Bibliografia

[1] Isabel Tootill. *O papel da internet no processo de Globalizacao*. Acedido em: 2022-12-22, Url: https://www.academia.edu/35679459/O_papel_da_internet_no_processo_de_Globalizacao. 2016 (ver p. 2).

[2] Débora Albuquerque. *A importância das telecomunicações para o crescimento econômico*. Acedido em: 2022-12-22, Url: https://economiadeservicos.com/2017/02/09/a-importancia-das-telecomunicacoes-para-o-crescimento-economico/. 2017 (ver p. 2).

[3] António Carlos Martins Oliveira. *Redes de acesso em fibra ótica : situação atual e evolução*. Acedido em: 2022-12-22, Url: https://repositorio.ipl.pt/handle/10400.21/8847. 2017 (ver p. 2).

[4] *Evolução dos acessos à Internet em Portugal*. Acedido em: 2022-12-22, Url: https://www.anacom.pt/streaming/estudoSpeedtest2015velocidadesBL.pdf?contentId=1374020&field=ATTACHED_FILE. 2015 (ver pp. 2, 3).

[5] Blair Campbell. *Rural Internet Options: Satellite vs. DSL*. Acedido em: 2022-12-23, Url: https://getinternet.com/rural-internet-satellite-vs-dsl/. 2022 (ver p. 2).

[6] Emanuel dos Santos Nunes. *Projeto de Redes de Fibra Ótica*. Acedido em: 2022-12-14, Url: https://iconline.ipleiria.pt/bitstream/10400.8/2450/1/Emanuel%20dos%20Santos%20Nunes-%20Mestrado%20em%20Eng.Eletrot%C3%A9cnica.pdf. 2016 (ver pp. 2, 4).

[7] *Fiber Internet vs 4G/5G- What's the difference? Which is better?* Acedido em: 2022-12-10, Url: https://www.zajil.com/fiber-internet-vs-4g-5g/. 2022 (ver p. 2).

[8] James Farmer et al. *FTTx Networks: Technology Implementation and Operation*. Morgan Kaufmann, 2016 (ver p. 3).

[9] Krzysztof Borzycki. «FTTx Access Networks: Technical Developments and Standardization». Em: *Broadband Communications Networks-Recent Advances and Lessons from Practice*. IntechOpen, 2017 (ver p. 3).

[10] António José Couto Rebelo da Costa. *Departamento de Electrónica, Telecomunicações e Informática*. Acedido em: 2022-12-7, Url: https://www.zajil.com/fiber-internet-vs-4g-5g/. 2012 (ver p. 3).

[11] Hadrien Louchet et al. *Impact of strategic decisions on the planning of FTTx-Networks: A case study*. Acedido em: 2022-12-7, Url: https://ieeexplore.ieee.org/document/6145200/authors#authors. 2011 (ver p. 3).

[12] Pekka Abrahamsson et al. *Agile software development methods: Review and analysis*. 2017 (ver p. 5).

[13] Cláudia Alexandra Coimbra Serrão. *Codificação, Bill of Materials e monitorização da produção na indústria da automação*. Acedido em: 2022-12-8, Url: https://comum.rcaap.pt/handle/10400.26/11726. 2015 (ver p. 7).

[14] Maicon Boettcher. *Revolução Industrial - Um pouco de história da Indústria 1.0 até a Indústria 4.0*. Acedido em: 2022-12-8, Url: https://www.linkedin.com/pulse/revolu%C3%A7%C3%A3o-industrial-um-pouco-de-hist%C3%B3ria-da-10-at%C3%A9-boettcher/?originalSubdomain=pt. 2015 (ver p. 7).

[15] Guilherme J. *BOM (Bill of Materials) - Um Componente essencial para estruturação e programação de materiais nas organizações*. Acedido em: 2022-12-8, Url: https://www.linkedin.com/pulse/bom-bill-materials-um-componente-essencial-para-e-1-aguiar-barbosa/?trk=public_profile_article_view. 2018 (ver pp. 7, 8).

[16] Marin Rusanescu. *Material requirements planning, inventory control system in industry*. Acedido em: 2022-12-14, Url: https://www.proquest.com/docview/1510594772. 2014 (ver p. 8).

[17] Jing Software. *Jing Software: BOM List Generator*. 2022. URL: https://apps.autodesk.com/MEP/en/Detail/Index?id=5226499155434189626 (acedido em 15/12/2022) (ver p. 12).

[18] Micro Graphics. *Micro Graphics: BOM Tools Pro*. 2022. URL: https://apps.autodesk.com/INVNTOR/en/Detail/?id=9090551896012244131 (acedido em 15/12/2022) (ver p. 12).

[19] OEMsecrets. *OEMsecrets: BOM Management*. 2022. URL: https://bomtool.oemsecrets.com/ (acedido em 15/12/2022) (ver p. 12).

[20] Mike Kasparian. *IndaBOM*. 2022. URL: https://indabom.com/ (acedido em 15/12/2022) (ver p. 13).

[21] Openbom. *Openbom: BOM Management Services*. 2022. URL: https://www.openbom.com/bom-management-capability (acedido em 15/12/2022) (ver p. 13).

[22] Weezie. *Weezie Fiber: FTTx intelligence and collaboration*. 2022. URL: https://weezie.io/ (acedido em 23/12/2022) (ver p. 14).

[23] IQGEO. *IQGEO: Network Manager Telecom*. 2022. URL: https://www.iqgeo.com/product/network-manager-telecom (acedido em 23/12/2022) (ver p. 14).

[24] VETRO. *VETRO: VETRO FiberMap*. 2022. URL: https://vetrofibermap.com/products/fibermap/ (acedido em 23/12/2022) (ver p. 14).

[25] KSAVI GNI. *KSAVI GNI: Geospatial Network Inventory*. 2022. URL: https://ksavinetworkinventory.com/network-inventory/ (acedido em 23/12/2022) (ver p. 14).

[26] Comsof. *Comsof Fiber: Fiber optic network planning & design software*. 2022. URL: https://www.comsof.com/fiber (acedido em 23/12/2022) (ver p. 15).

[27] Jonathan Spruytte et al. «Joint-Rollout of FTTH and Smart City Fiber Networks as a Way to Reduce Rollout Cost». Em: *2019 CTTE-FITCE: Smart Cities & Information and Communication Technology (CTTE-FITCE)*. 2019, pp. 1–5. DOI: 10.1109/CTTE-FITCE.2019.8894814 (ver p. 15).

[28] João M. Fernandes e Ricardo J. Machado. *Requirements in engineering projects*. Springer International Publishing, 2016. URL: https://link.springer.com/10.1007/978-3-319-18597-2 (ver p. 19).

[29] *Arquivos DWG*. Acedido em: 2022-11-24, Url: https://www.adobe.com/pt/creativecloud/file-types/image/vector/dwg-file.html (ver p. 25).

[30] *Arquivos DXF*. Acedido em: 2022-11-24, Url: https://www.adobe.com/creativecloud/file-types/image/vector/dxf-file.html (ver p. 26).

[31] Han Zhang e Xueqing Li. «Data extraction from DXF file and visual display». Em: *HCI International 2014-Posters' Extended Abstracts: International Conference, HCI International 2014, Heraklion, Crete, Greece, June 22-27, 2014. Proceedings, Part I 16*. Springer. 2014, pp. 286–291 (ver p. 26).

[32] Autodesk. *Autodesk: RealDWG OEM*. 2023. URL: https://aps.autodesk.com/developer/overview/realdwg-oem (acedido em 05/01/2023) (ver p. 26).

[33] *DWG vs. DXF: Which Is Best?* Acedido em: 2022-11-24, Url: https://www.adobe.com/creativecloud/file-types/image/comparison/dwg-vs-dxf.html (ver p. 26).

[34] Open Design Alliance. *Open Design Alliance: ODA DWG-DXF CONVERTER*. 2023. URL: https://www.opendesign.com/guestfiles/oda_file_converter (acedido em 05/02/2023) (ver p. 27).

[35] Cloudconvert. *Cloudconvert: DXF to DWG Converter*. 2023. URL: https://cloudconvert.com/dxf-to-dwg (acedido em 05/02/2023) (ver p. 27).

[36] Aspose. *Aspose: Aspose.CAD for .NET*. 2023. URL: https://docs.aspose.com/cad/net/ (acedido em 05/02/2023) (ver p. 27).

[37] Ibercad. *Ibercad: ZWCAD*. 2023. URL: https://www.ibercad.pt/zwcad.html (acedido em 05/02/2023) (ver p. 27).

[38] AutoDWG. *AutoDWG: DWG DXF converter 2022*. 2023. URL: https://www.autodwg.com/dwg_dxf_converter/ (acedido em 05/02/2023) (ver p. 27).

[39] Shaan Hurley. *AutoCAD Release History*. Acedido em: 2022-11-25, Url: https://www.btl-blog.com/between_the_lines/autocad-release-history.html (ver p. 28).

[40] *SQL vs. NoSQL Databases: What's the Difference?* Acedido em: 2023-01-22, Url: https://www.ibm.com/blog/sql-vs-nosql/. 2023 (ver p. 31).

[41] *O que é um banco de dados de chave-valor?* Acedido em: 2023-01-22, Url: https://aws.amazon.com/pt/nosql/key-value/. 2023 (ver p. 32).

[42] *O que é um banco de dados de documentos?* Acedido em: 2023-01-22, Url: https://aws.amazon.com/pt/nosql/document/. 2023 (ver p. 32).

[43] *O que é um banco de dados em colunas?* Acedido em: 2023-01-22, Url: https://aws.amazon.com/pt/nosql/columnar/. 2023 (ver p. 32).

[44] *O que é um banco de dados de grafos?* Acedido em: 2023-01-22, Url: https://aws.amazon.com/pt/nosql/graph/. 2023 (ver p. 33).

[45] *Blowfish algorithm*. Acedido em: 2023-07-14, Url: https://www.geeksforgeeks.org/blowfish-algorithm-with-examples/ (ver p. 33).

Parte II
Apêndices

Apêndice A
Documento de Requisitos

1. Introdução

Este documento especifica os requisitos para a aplicação **FTTx BoM**, sendo constituído pelas informações necessárias relativas às funcionalidades a serem disponibilizadas na primeira versão da mesma.

A aplicação pretende automatizar a geração da lista de materiais (**BoM**) para projetos de rede de fibra ótica, atualmente construída de forma manual, sendo um processo demoroso e suscetível a erros. Com a aplicação, espera-se que o tempo do processo de criação da lista seja reduzido, além do resultado se tornar mais preciso e fiável. O cliente pretende que a aplicação possibilite registar operadoras e associar a estas as suas respetivas regras de construção, além de ser capaz de gerar a lista de materiais com o fornecimento dos documentos relativos ao sinótico, ao mapa de rede e à tabela de alocação de juntas, já validados, resultantes do ***survey*** da área. A lista de materiais construída deverá considerar as regras da operadora cliente. Também deverá ser possível gerar a lista de materiais para uma operadora ao qual não pertença o sinótico, mapa de rede e tabela de alocação.

2. Requisitos Funcionais

Neste capítulo são apresentados os requisitos funcionais da aplicação, sendo classificados relativamente à sua **prioridade** e **risco** fazendo uso de uma escala de três valorações. No caso da prioridade, relativa à importância da sua implementação para o cliente, alta, média e baixa. No caso do risco, relativo à minha visibilidade sobre forma de implementação da solução para o problema, portanto, podendo ser baixo, médio ou alto. Também é estimado o **esforço** de implementação recorrendo à sequência de Fibonacci.

Identificador	RF-01	**Título**	Gestão de operadoras		
Descrição	Como utilizador quero adicionar, remover, editar e procurar as operadoras que pretender.				
Prioridade	Alta	**Risco**	Baixo	**Esforço**	2
Condições de aceitação:					
1. Permitir que o utilizador adicione, remova, edite e procure uma operadora. 2. Quando uma operadora é removida, todos os seus dados são removidos, incluindo regras, templates e componentes. 3. Não podem ser adicionadas duas operadoras com um mesmo nome.					

Identificador	RF-02	**Título**	Gestão de regras
Descrição	Como utilizador quero adicionar, remover, editar e procurar regras nas operadoras que pretender. Cada regra deverá ser composta		

	pela aplicação de material a um componente de rede especifico, juntamente da quantidade a aplicar.				
Prioridade	Alta	**Risco**	Baixo	**Esforço**	3
Condições de aceitação:					
1. Permitir que o utilizador adicione, remova, edite e procure uma regra. 2. Terá de ser selecionado o componente da rede ao qual se aplica a regra. 3. Terá de ser selecionado o material a aplicar no componente de rede, juntamente com a respetiva quantidade.					

Identificador	RF-03	**Título**	Gestão de templates		
Descrição	Como utilizador quero adiconar e remover templates BoM às operadoras, devido a estes constantemente sofrerem modificações.				
Prioridade	Alta	**Risco**	Baixo	**Esforço**	3
Condições de aceitação:					
1. Permitir que o utilizador adicone e remova templates BoM, em formato XLSX. 2. No template BoM deverão constar todos os materiais possíveis de serem aplicados nas regras. Além de cada material ter de possui um identificador imutável. 3. O utilizador poderá selecionar o template pretendido quando criar as regras e quando gerar o BoM.					

Identificador	RF-04	**Título**	Gestão de componentes		
Descrição	Como utilizador quero adicionar, editar, atualizar e remover o nome dos atributos dos componentes que deverão ser reconhecidos do sinótico, mapa e tabela de alocação.				
Prioridade	Médio	**Risco**	Médio	**Esforço**	8
Condições de aceitação:					
1. Permitir que o utilizador adicione, edite, remova e atualize o nome dos atributos do sinótico, mapa de rede e tabela de alocação. 2. O reconhecimento só será bem efetuado se todos os atributos solicitados forem devidamente especificados tal como no: sinótico, mapa e tabela de alocação.					

Identificador	RF-05	**Título**	Importar sinótico		
Descrição	Como utilizador quero importar o sinótico, onde constam maioritariamente informações dos componentes de rede, para posteriormente gerar o BoM.				
Prioridade	Alta	**Risco**	Médio	**Esforço**	13
Condições de aceitação:					
1. Permitir que o utilizador importe a informação contida no sinótico em ficheiro de formato DXF.					

2. A gestão de componentes já deverá ter sido efetuada, antes da importação.
3. Não podem ser importados dois sinóticos em simultâneo. Uma vez importado um, para proceder à importação de outro, terá de ser descartado o anterior.

Identificador	RF-06	**Título**	Importar mapa		
Descrição	Como utilizador quero importar o mapa, onde constam informações maioritariamente relativas à infraestrutura de suporte da rede, para posteriormente gerar o BoM.				
Prioridade	Alta	**Risco**	Médio	**Esforço**	13
Condições de aceitação:					

1. Permitir que o utilizador importe a informação contida no mapa em ficheiro de formato DXF.
2. O mapa de rede tem de ser relativo ao sinótico previamente inserido.
3. A gestão de componentes já deverá ter sido efetuada, antes da importação.
4. Não podem ser importados dois mapas em simultâneo. Uma vez importado um, para proceder à importação de outro, terá de ser descartado o anterior.

Identificador	RF-07	**Título**	Importar tabela de alocação		
Descrição	Como utilizador quero importar a tabela de alocação, onde constam as informações das fusões a serem feitas nos componentes, para posteriormente gerar o BoM.				
Prioridade	Alta	**Risco**	Médio	**Esforço**	8
Condições de aceitação:					

1. Permitir que o utilizador importe a tabela de alocação em ficheiro de formato XLSX.
2. A tabela de alocação tem de ser relativa ao sinótico e mapa já inseridos.
3. Não podem ser importadas duas tabelas de alocação. Uma vez importada uma tabela, para proceder à importação de outra, terá de ser descartada a anterior.

Identificador	RF-08	**Título**	Selecionar a operadora		
Descrição	Como utilizador quero selecionar a operadora para a qual pretendo obter o BoM. Também pretendo que seja possível selecionar uma operadora à qual os documentos de projeto: sinótico, mapa e tabela de alocação não correspondam.				
Prioridade	Alta	**Risco**	Médio	**Esforço**	8
Condições de aceitação:					

1. O utilizador deverá selecionar qual a operadora para o qual pretende gerar o BoM.
2. A operadora já tem de estar inserida na aplicação.

Identificador	RF-9	**Título**	Selecionar o tipo de lista de materiais

Descrição	Como utilizador quero selecionar o tipo de lista de materiais a gerar, se total, partilhável ou não partilhável.				
Prioridade	Média	**Risco**	Médio	**Esforço**	3

Condições de aceitação:

1. O utilizador deverá selecionar se pretende o BoM total, com a totalidade de material necessário, esteja ele já implantado ou não, o BoM partilhável, contendo apenas o material do terreno já implantado, ou o BoM não partilhável, apenas com o material necessário de ser implantado.
2. Todos os componentes partilháveis devem constar inseridos, tanto no mapa como no sinótico sob um marcador especifico para identificação de materiais partilháveis.

Identificador	RF-10	**Título**	Selecionar o tipo de rede		
Descrição	Como utilizador quero selecionar o tipo de rede para a qual o BoM deve ser gerado: rede primária ou secundária.				
Prioridade	Média	**Risco**	Médio	**Esforço**	3

Condições de aceitação:

1. O utilizador deverá selecionar se prentede o BoM de rede primária ou secundária.
2. Já deverá ter sido efetuada a configuração, na gestão de componentes, do aspeto identificador da rede.

Identificador	RF-11	**Título**	Gerar o BoM		
Descrição	Como utilizador quero gerar e exportar o BoM.				
Prioridade	Alta	**Risco**	Baixo	**Esforço**	8

Condições de aceitação:

1. Permitir que o utilizador gere e exporte em ficheiro de formato XLSX o BoM.
2. Já deverão ter sido importados o sinótico, o mapa e a tabela de alocação.
3. A operadora para o qual se pretende o BoM já deverá ter sido selecionada.
4. Já deverá ter sido efetuada a gestão de componentes, a criação das regras pretendidas, e a adição de pelo menos um template BoM, à correspondente operadora.
5. Já deverá ter sido selecionado o template a utilizar, e o tipo de rede e BoM a obter.
6. O BoM deverá estar de acordo com o template selecionado.

Identificador	RF-12	**Título**	Histórico de operações		
Descrição	Como utilizador quero possuir o histórico de operações efetuadas sobre a aplicação e anomalias encontradas.				
Prioridade	Media	**Risco**	Baixo	**Esforço**	2

Condições de aceitação:

1. Permitir que o utilizador visualize o histórico de operações sobre a aplicação e anomalias ocorridas na aplicação.
2. Gerar anomalias sempre que ocorrer algum erro devido à má utilização da aplicação.
3. Registar finalização das operações, quando estas terminam.
4. O histório será temporário periodo de execução da aplicação, sempre que esta for fechada o histórico será eliminado.

Identificador	RF-13	**Título**	Persistência de dados		
Descrição	Como utilizador quero que as configurações efetuadas na aplicação sejam guardadas de forma a que possam ser utilizadas por vários utilizadores.				
Prioridade	Alta	**Risco**	Baixo	**Esforço**	5
Condições de aceitação:					

1. Permitir que o utilizador não necessite de efetuar a configuração da aplicação, após realizada uma vez.
2. Permitir que a configuração efetuada por um utilizador possa ser utilizada por outros utilizador noutras máquinas, em paralelo.

Identificador	RF-14	**Título**	Autenticação		
Descrição	Como utilizador quero a possibilidade de registar na aplicação utilizadores com privilégios de administrador ou regular. O administrador poderá aceder a todas as funcionalidades enquanto que o regular apenas poderá gerar o BoM.				
Prioridade	Alta	**Risco**	Médio	**Esforço**	8
Condições de aceitação:					

1. Permitir o registo de utilizadores com privilégio de administrador ou regular.
2. Impossiblitar a utilização de outras funcionalidades à exceção da geração do BoM, no caso do utilizador com privilégio regular.
3. Impossibilitar a utilização da aplicação sem autenticação.

3. Requisitos Não-Funcionais

Neste capítulo serão apresentados os requisitos não funcionais, correspondentes à forma como a aplicação deverá ser desenvolvida. A decisão dos requisitos não funcionais, devido à especificidade da aplicação, têm em principal consideração a finalidade e ambiente de utilização da mesma, portanto, a sua utilização será feita sempre por utilizadores envolvidos no domínio do problema, assim como o ambiente da sua utilização será interno à empresa.

3.1. Aparência

Identificador	RNF-01	**Título**	Aparência
Descrição	Não foram solicitados, nem apresentam importância para o sucesso da aplicação.		
Prioridade	Nula	**Risco**	Nulo

3.2. Usabilidade

Identificador	RNF-02	**Título**	Usabilidade
Descrição	Não foram solicitados por questões de tempo requerido para avaliação, apesar de apresentarem importância para o sucesso da aplicação.		
Prioridade	Nula	**Risco**	Nulo

3.3. Desempenho

Identificador	RNF-03	**Título**	Desempenho
Descrição	A geração do BoM deve ser feita num intervalo de tempo consistente, sem depêndencia de fatores como largura de banda ou sobrecarga. O intervalo de tempo não deve ultrapassar 5 minutos.		
Prioridade	Alta	**Risco**	Médio

3.4. Conectividade

Identificador	RNF-04	**Título**	Conectividade
Descrição	Para possibilitar a utilização dos dados da aplicação a múltiplos utilizadores é exigida conexão permanente à Internet.		
Prioridade	Alta	**Risco**	Médio

3.5. Manutenibilidade

Identificador	RNF-05	**Título**	Manutenibilidade
Descrição	Uma vez que estamos perante uma aplicação prótotipal, o surgimento de novos requisitos é imprevisível, logo é importante que todo o código desenvolvido esteja documentado e a sua arquitetura pensada sob a forma de componentes modulares, de modo a tornar cada um o mais independente possível.		
Prioridade	Alta	**Risco**	Baixo

3.6. Segurança

Identificador	RNF-06	**Título**	Segurança

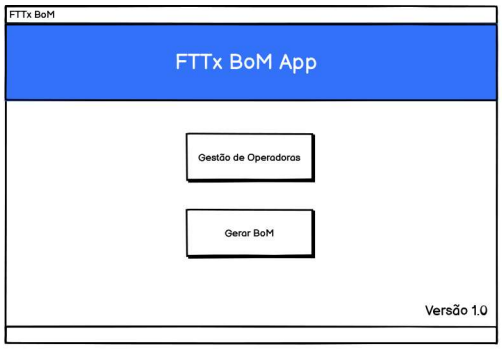
Descrição	As credênciais dos utilizadores devem ser encriptadas antes de serem guardadas e o acesso ao dados apenas poderá ser feito por utilizadores devidamente autenticados, com privilégio de acesso adequado (administrador ou regular).		
Prioridade	Média	**Risco**	Baixo

3.7. Legalidade

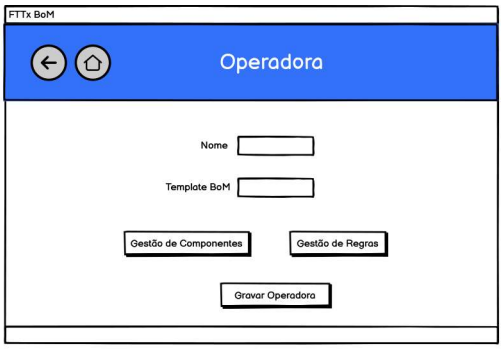
Identificador	RNF-07	**Título**	Legalidade
Descrição	De modo a minimizar os custos envolvidos no desenvolvimento e na manutenção da aplicação, a mesma, apenas pode recorrer a tecnologias *open-source*, portanto, que não necessitem de liceças pagas.		
Prioridade	Alta	**Risco**	Baixo

Apêndice B
Mockup da aplicação

A seguir é apresentado o primeiro **Mockup** concebido tendo em consideração a primeira versão do documento de requisitos. Como é possível verificar, existem diversos aspetos de diferenciação face à Interface e funcionalidades finais.

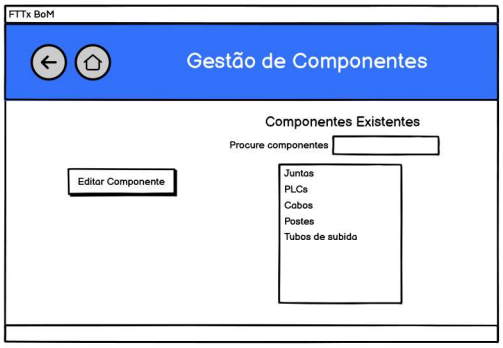

FTTx BoM
FTTx BoM App
Gestão de Operadoras
Gerar BoM
Versão 1.0

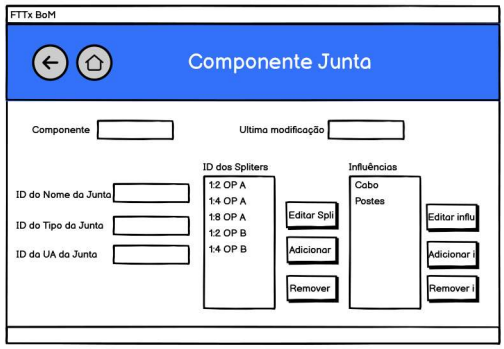

FTTx BoM
Gestão de Operadoras
Migração de dados
Importar
Exportar
Operadoras Registadas
Procure operadora
Adicionar Operadora
Editar Operadora
Remover Operadora

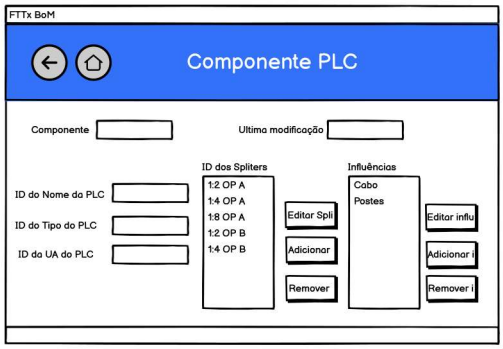

FTTx BoM
Operadora
Nome
Template BoM
Gestão de Componentes
Gestão de Regras
Gravar Operadora

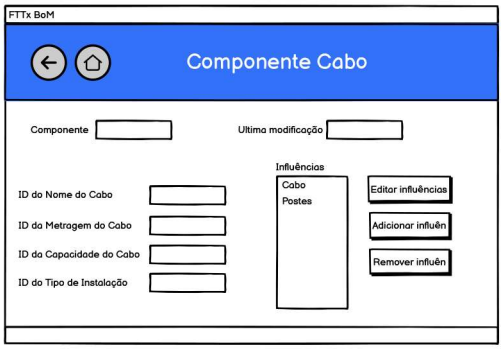

FTTx BoM
Operadora
Nome
Data do ultimo template BoM
Atualizar Template
Gestão de Componentes
Componentes
Regras
Juntas
PLCs
Cabos
Postes
Tubos de subida
Gestão de Regras
Existem 2 anomalias.
Verificar
Remover Operadora
JS -> Fivela, Fita, 2 preparacoes c
Cabo -> Mordente, parafuso.
PLC ->
Tubo de Subida ->
Poste ->


FTTx BoM
Gestão de Regras
Regras Registadas
Procure regras
Editar Regra
Adicionar Regra
Remover Regras
JS -> Fivela, Fita, 2 preparacoes cabo.
Cabo -> Mordente, parafuso.
PLC ->
Tubo de Subida ->
Poste ->


FTTx BoM
Gestão de Componentes
Componentes Existentes
Procure componentes
Editar Componente
Juntas
PLCs
Cabos
Postes
Tubos de subida

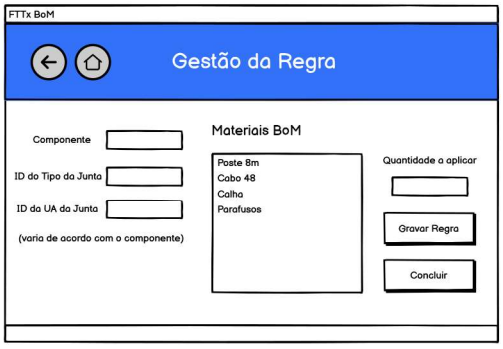

FTTx BoM
Componente Junta
Componente
Ultima modificação
ID do Nome da Junta
ID do Tipo da Junta
ID da UA da Junta
ID dos Spliters
1:2 OP A
1:4 OP A
1:8 OP A
1:2 OP B
1:4 OP B
Editar Spli
Adicionar
Remover
Influências
Cabo
Postes
Editar influ
Adicionar i
Remover i

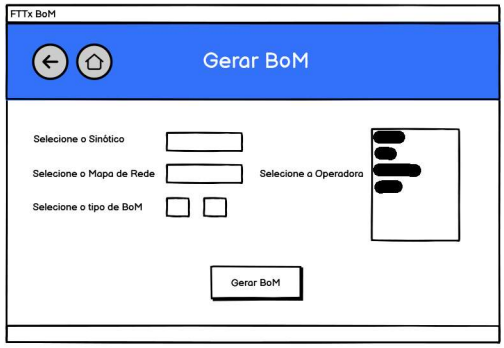

FTTx BoM
Componente PLC
Componente
Ultima modificação
ID do Nome da PLC
ID do Tipo do PLC
ID da UA do PLC
ID dos Spliters
1:2 OP A
1:4 OP A
1:8 OP A
1:2 OP B
1:4 OP B
Editar Spli
Adicionar
Remover
Influências
Cabo
Postes
Editar influ
Adicionar i
Remover i

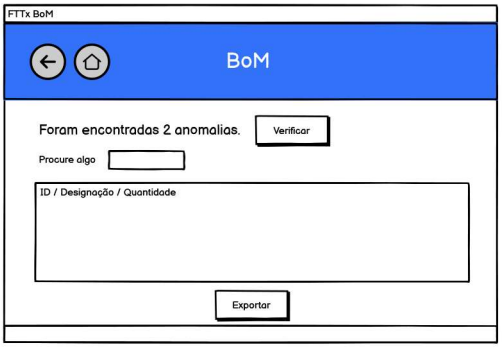

FTTx BoM
Componente Cabo
Componente
Ultima modificação
ID do Nome do Cabo
ID da Metragem do Cabo
ID da Capacidade do Cabo
ID do Tipo de Instalação
Influências
Cabo
Postes
Editar influências
Adicionar influên
Remover influên


FTTx BoM
Influências
Selecione os componentes que influência
Cabo
Postes
Influência por posição
Qual o raio de influência


FTTx BoM
Regra
Selecione o componente
Junta
PLCs
Cabo
Postes
Tubo de Subida
Seguinte


FTTx BoM
Gestão da Regra
Componente
ID do Tipo da Junta
ID da UA da Junta
(varia de acordo com o componente)
Materiais BoM
Poste 8m
Cabo 48
Calha
Parafusos
Quantidade a aplicar
Gravar Regra
Concluir


FTTx BoM
Gerar BoM
Selecione o Sinótico
Selecione o Mapa de Rede
Selecione o tipo de BoM
Selecione a Operadora
Gerar BoM


FTTx BoM
BoM
Foram encontradas 2 anomalias.
Verificar
Procure algo
ID / Designação / Quantidade
Exportar


FTTx BoM
Erros
Erros encontrados
Material x nao foi associado.
Componente y n teve materiais.
Exportar