INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Sistema de gestão integrada no âmbito da educação nas autarquias

PEDRO MILLER BRANDÃO PINHO
Outubro de 2021

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Sistema de gestão integrada no âmbito da educação nas autarquias

Pedro Miller Brandão Pinho

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Sistemas Gráficos e Multimédia

Orientador: Filipe Pacheco

Supervisor Externo: Paulo Cunha

Júri:

Presidente:

José Marinho, Professor, DEI/ISEP

Vogais:

Joaquim Santos, Professor, DEI/ISEP

Porto, outubro 2021

Aos meus pais, família, namorada e amigos.

Resumo

A componente digital do ensino tem vindo a crescer ao longo dos anos, tornando-se numa componente fundamental do sistema de ensino em vários países. Este crescimento é fruto das constantes evoluções tecnológicas, e do uso da *internet,* que torna possível interligar todos os intervenientes num ecossistema de educação.

A Medidata, empresa que trabalha com autarquias portuguesas e possui um sistema de gestão integrada de autarquias, encontrou uma oportunidade em conjunto com um dos seus clientes, que visa responder à crescente necessidade de informatizar todo o processo educativo, o que pode trazer muitas vantagens, não só para a autarquia, como para os encarregados de educação, assim como todos os outros intervenientes, que não têm de se deslocar e têm toda a informação desejada à distância de um clique. Esta solução será uma grande mais-valia para ambas as partes, quer os utilizadores quer o cliente.

A presente dissertação analisa com detalhe o problema e objetivos propostos, contextualiza e analisa os requisitos, faz um levantamento e análise de soluções já existentes, assim como propõe alternativas de design para uma solução robusta que venha responder a estes problemas – o Portal da Educação.

Ao longo do desenvolvimento do projeto, a aplicação foi sendo disponibilizada para a realização de testes por parte dos clientes, o que permitiu ser utilizada já em ambiente de produção com sucesso antes do final do desenvolvimento, enquanto era recolhido *feedback* valioso para o desenvolvimento.

Na fase final do projeto, a aplicação implementada foi instalada nas autarquias e é utilizada com grande sucesso pelos clientes, e continua a ser desenvolvida seguindo um processo iterativo. O *feedback* obtido é positivo, pelo que os objetivos foram alcançados, e a solução vai de encontro ao problema a resolver. Isto leva também ao aparecimento de trabalho futuro a desenvolver, com vista a melhorar ainda mais a solução desenvolvida.

Palavras-chave: Educação, Autarquias, Plataforma de Gestão

Abstract

The digital component of education has been growing over the years, becoming a fundamental component of the education system in several countries. This growth is the result of constant technological developments, and the use of the internet, which makes it possible to interconnect all the actors in an education ecosystem.

Medidata, a company that works with Portuguese municipalities and has an integrated management system for municipalities, has found an opportunity in conjunction with one of its clients, which aims to respond to the growing need to informatize the entire educational process, which can bring many advantages, not only for the municipality, but also for parents, as well as all other stakeholders, who do not have to move and have all the desired information just a click away. This solution will be a great asset for both parties, be it the users or the client.

This dissertation analyzes the proposed problem and objectives in detail, contextualizes and analyzes the requirements, makes a research and analysis of existing solutions, as well as proposes design alternatives for a robust solution that will answer these problems - the “Portal da Educação”.

Throughout the project's development, the application was made available for testing to the customers, which allowed it to be successfully used in a production environment before the end of the development, while valuable feedback for development was being collected.

In the final phase of the project, the implemented application was installed in the municipalities and is being used with great success by the customers and continues to be developed following an iterative process. The feedback obtained is positive, so the objectives were met, and the solution solves the proposed problem. This also leads to the emergence of future work to be done to further improve the developed solution.

Keywords: Education, Municipality, Management Platform

Agradecimentos

Nesta secção gostaria de demonstrar a minha gratidão a todos aqueles que me acompanharam ao longo do meu percurso académico e que tornaram tudo isto possível.

Agradeço ao Instituto Superior de Engenharia do Porto e a todos os docentes, colegas e amigos que acompanharam todo o meu percurso, e que me permitiram aprender e crescer enquanto pessoa.

Agradeço também em especial ao Eng.º Filipe Pacheco pela disponibilidade e orientação durante este projeto.

Agradecimento à Medidata e a todos os seus colaboradores, em especial ao Eng.º Paulo Cunha pela oportunidade de desenvolver este projeto e por todo o apoio e ferramentas disponibilizadas.

Um grande agradecimento a todos os meus amigos e colegas mais próximos que fazem parte da minha vida, pelo apoio e pela amizade, sem eles este caminho não teria sido o mesmo.

Agradeço aos meus pais e à minha família que sempre me apoiaram e proporcionaram todas as condições necessárias para concluir o meu percurso académico.

Por último, e em especial, um enorme agradecimento à minha namorada, por ter sempre acreditado em mim, pela sua companhia e particularmente por todo o seu apoio ao longo deste projeto.

A todos que foram aqui mencionados, o meu mais sincero obrigado.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos e Glossário

MEI	Mestrado em Engenharia Informática
ISEP	Instituto Superior de Engenharia do Porto
NCD	*New Concept Development Model*
FFE	*Fuzzy Front End (Front End of Innovation)*
AEC	Atividade(s) extra curriculare(s)
β	Largura de banda
UML	*Universal Modeling Language*
JSON	*JavaScript Object Notation*
ERP	*Enterprise Resource Planning* ou Sistema Integrado de Gestão Empresarial
WSL	*Windows Subsystem for Linux*
HTTP	*Hyper Text Transfer Protocol*
YAML	*Yet Another Markup Language*
CI/CD	*Continous Integration / Continuous Deployment*
SQL	*Structured Query Language*
UML	*Unified Modeling Language*
API	*Application Programming Interface*
UC	*Use Case* (Caso de Uso)
ORM	*Object-relational mapping*
CRUD	*Create, Read, Update, Delete*
GIT	Sistema de Controlo de Versões
REST	*Representational State Transfer*
URL	*Uniform Resource Locator*

Tabela 1 - Glossário de expressões e acrónimos utilizados

1 Introdução

1.1 Contexto

O presente documento surge no âmbito da unidade curricular de Tese / Dissertação / Estágio do MEI (Mestrado em Engenharia Informática), na área de especialização de Sistemas Gráficos e Multimédia do ISEP (Instituto Superior de Engenharia do Porto), e tem como objetivo expor e documentar todo o trabalho desenvolvido ao longo do ano letivo no projeto em questão, um Portal da Educação orientado à gestão da educação nas autarquias.

Este pequeno capítulo introdutório destina-se a contextualizar e a apresentar sucintamente o projeto a desenvolver, interpretar o problema e delinear objetivos e expetativas, assim como entrar em maior detalhe providenciando informação relativa a todos os conceitos de negócio, incluindo uma análise de valor e o estado da arte em soluções semelhantes, não só a nível de respostas ao problema, mas como a nível de tecnologias relevantes.

A organização responsável pelo desenvolvimento deste projeto é a Medidata, empresa portuguesa que opera principalmente na criação de soluções informáticas na área de Administração Pública, tendo como principais clientes diversas Autarquias em Portugal. Esta relação com diversas Autarquias leva então a que exista um meio propício a desenvolver novos projetos e serviços de acordo com requisitos e necessidades dos clientes.

1.2 Problema

A gestão digital do universo do ensino tem vindo a crescer ao longo dos anos, tornando-se uma componente fundamental do sistema de ensino em vários países. É neste sentido que surge a oportunidade de desenvolver uma plataforma que permita responder a todas as necessidades de uma determinada autarquia, no ecossistema de agrupamentos, escolas, alunos, AECs, etc.

Alia-se a esta necessidade crescente o facto de as autarquias clientes da Medidata terem elaborado uma série de requisitos e especificações para uma nova plataforma de gestão de

atividades extra curriculares, em que os encarregados de educação podem consultar todas as informações dos seus educandos, desde aulas, faltas e até refeições e escalões de subsídio, assim como ter perfis para professores e funcionários não-docentes, que permita consultar, editar e criar toda a informação relevante neste meio, e que ao mesmo tempo tenha uma integração profunda com a informação proveniente dos sistemas de administração já implementados e mantidos pela Medidata nas autarquias desde a década de 90.

1.3 Objetivos

O objetivo principal deste projeto é implementar uma plataforma que permita às autarquias gerir, criar, e disponibilizar toda a informação necessária à gestão de agrupamentos, escolas, professores, etc. Com vista a conseguir atingir este objetivo é necessária uma análise aos requisitos fundamentais, e efetuar uma subdivisão para que se consiga definir metas mais facilmente, e desta forma organizar todo o processo de desenho e desenvolvimento da plataforma.

O Portal da Educação permitirá acesso a um conjunto de utilizadores que se subdividem de acordo com diferentes perfis, logo um dos primeiros objetivos a definir é a clara subdivisão de comportamentos conforme os diferentes perfis de utilizador. Desta forma, podemos dividir todo o projeto em alguns grandes perfis:

- Autarquia;
- Agrupamento;
- Escola;
- Fornecedor;
- Professor;
- Funcionário Não Docente;
- Encarregado de Educação;

Cada um destes perfis terá uma série de funcionalidades, algumas partilhadas, outras únicas, sendo que é fundamental identificar soluções para os problemas apresentados, definir uma arquitetura que satisfaça todos os problemas, implementar esta arquitetura, e eventualmente testar toda a plataforma e as várias funcionalidades, de uma forma iterativa e contínua, de modo a ser possível o cliente receber sempre as últimas atualizações.

1.4 Abordagem preconizada

Numa primeira abordagem do projeto, após realizada uma interpretação do documento que contém a informação acerca da maior parte das funcionalidades fundamentais da plataforma, o passo lógico será definir uma arquitetura que responda ao que é pretendido, assim como tecnologias relevantes para a implementação em questão.

Em termos de limitações a ter em conta aquando do planeamento da abordagem a efetuar, existem algumas que devemos ter em conta, nomeadamente a dependência dos dados provenientes do *software* das Autarquias, tais como dados sensíveis referentes a escolas, turmas e até alunos, sendo que a plataforma terá de ser construída com um bom planeamento de segurança.

Em relação à forma como será abordada a implementação, é importante notar que assenta numa abordagem iterativa e incremental, através da realização de pequenos casos de uso, eventualmente levando à publicação de novas versões da plataforma, desta forma permitindo às Autarquias utilizar o Portal da Educação ainda sem todas as suas funcionalidades.

1.5 Análise de Valor

Num mundo empresarial cada vez mais competitivo e global, onde a qualidade desempenha um papel determinante, é fundamental analisar todas as variáveis

A análise de valor é um processo de avaliação e organização de um determinado produto ou solução, e tem como objetivo perceber qual o caminho a seguir com vista a minimizar o custo de um produto / serviço, enquanto se procura aumentar o seu valor sem diminuir a qualidade, assegurando, acima de tudo, a satisfação do cliente, mantendo a vantagem em relação a outros competidores. [1]

A análise de valor será aprofundada utilizando várias métricas que permitem estabelecer o valor do projeto no subcapítulo “Análise de Valor” do capítulo “Contexto e Estado da Arte”.

1.6 Estrutura da Dissertação

Este documento está organizado segundo capítulos e secções. Está dividido em seis grandes capítulos: Introdução, Contexto e Estado da arte, Design da solução, Implementação da solução, Avaliação e Experimentação e Conclusão. O primeiro capítulo, de caráter introdutório, descreve de forma sucinta o problema, o projeto proposto, objetivos a atingir e uma abordagem preliminar, contextualizando-o dentro do ambiente da empresa, etc.

No capítulo do Contexto e Estado da Arte, primeiramente é especificado em detalhe o contexto e o problema de uma forma mais aprofundada, os conceitos do negócio, processos e intervenientes, assim como possíveis restrições e limitações existentes. Quanto à parte do estado da arte é elaborada uma síntese de soluções e trabalhos que procuram apresentar soluções a problemas iguais ou semelhantes, tal como uma apresentação e eventual comparação de tecnologias utilizadas ou relevantes de alguma forma. Em relação à avaliação de soluções e abordagens, será discutido como devem ser usadas as soluções e trabalhos apresentados, assim como uma sucinta avaliação de várias métricas, de certa forma determinando o valor qualitativo dessas abordagens existentes.

No capítulo referente à análise e design da solução é apresentada uma abordagem mais técnica e detalhada sobre a solução pretendida, recorrendo a vários diagramas e subcapítulos, assim como são fundamentadas as decisões tomadas com base nos requisitos adquiridos do cliente.

De seguida, no capítulo da implementação da solução é documentada a implementação em si, a estrutura do código, como é que os componentes comunicam entre si, assim como os devidos testes. Neste caso documenta os vários passos de evolução do projeto, e o processo de desenvolvimento que leva à publicação de novas versões.

Em penúltimo surge o capítulo da avaliação e experimentação, que tem como objetivo explicitar de que forma a solução apresentada é avaliada, tanto em termos quantitativos como qualitativos, quais as métricas utilizadas para chegar a tal avaliação, assim como as metodologias de experimentação às quais a plataforma foi submetida.

Por fim surge o capítulo da conclusão, que contém um resumo e várias considerações importantes retiradas durante todo o desenvolvimento do projeto, assim como identificar trabalho futuro a ser desenvolvido, sem esquecer uma apreciação global e algumas notas pessoais.

2 Contexto e Estado da Arte

Neste capítulo será aprofundado o contexto do presente documento de uma forma mais detalhada como também o estado da arte, não só de soluções semelhantes já existentes, assim como das tecnologias relevantes para a solução do problema identificado. De forma a dar ao leitor uma melhor compreensão do projeto, serão também detalhados os conceitos de negócio, os intervenientes na plataforma e todas as restrições existentes.

2.1 Plataformas de Gestão no âmbito da Educação

Tal como foi apresentado no capítulo anterior, a crescente necessidade da componente de gestão digital relativamente ao ensino levou, naturalmente, ao aparecimento de várias plataformas semelhantes com o objetivo de responder a esta necessidade. Estas plataformas seguem um fluxo de utilização semelhante que procura facilitar a interação entre os agrupamentos / escolas e os encarregados de educação, assim como todos os outros intervenientes num sistema deste género, como por exemplo a consulta dos dados, aulas e faltas de um aluno por parte do seu encarregado de educação, entre outros. Este crescimento deve-se principalmente ao aumento do acesso a tecnologia no dia-a-dia, sendo que todo o conteúdo deve ser fácil de aceder, consultar e gerir. É também importante realçar que só recentemente é que começaram a existir este tipo de plataformas que não são exclusivamente para uso interno da escola, e que além de resolver vários problemas como por exemplo o da deslocação física dos pais dos alunos às escolas, fá-lo mantendo a utilização simples e muito amigável para utilizadores sem qualquer experiência prévia. Para isso é necessário desenvolver sistemas que capacitem todos os intervenientes de o fazer de uma forma fácil e intuitiva. O desenvolvimento deste tipo de sistemas depende da correta separação de atividades e necessidades dos intervenientes, que será estudado neste capítulo.

2.2 Abordagem ao problema

Na presente secção será analisado o problema identificado no capítulo anterior, de uma forma mais aprofundada, com vista a deixar claro para o leitor quais os conceitos de negócio a respeitar na implementação, quais os intervenientes nos vários processos e também quais os casos de uso que a solução final deverá apresentar.

2.2.1 Problema

O problema inicial foi colocado por um dos clientes da Medidata, que demonstrou com certa urgência a necessidade de desenvolver um sistema de gestão do ensino, para que todos os intervenientes no ecossistema do ensino da autarquia pudessem registar-se e ter acesso a diversas informações, e para que se pudesse efetuar uma gestão inteligente de todos os recursos da autarquia nesse sentido. É importante notar que, como já foi referido, um dos grandes requisitos é que a plataforma terá de ter integração próxima com os dados já existentes provenientes das escolas e agrupamentos de cada autarquia, tais como os dados dos alunos e dos munícipes, além dos próprios dados criados e mantidos no sistema, criando desta forma uma junção de toda a informação já disponível com novos dados a serem geridos e interligados.

Rapidamente é possível perceber a dimensão do problema, e de toda a estrutura de dados necessária para suportar uma plataforma assim, sendo que mais para a frente no capítulo do design serão abordados todos os requisitos propostos pelo cliente, assim como todo o desenho da solução de raiz, seguido depois da implementação da mesma.

2.2.2 Conceitos importantes

AECs

As atividades extracurriculares são importantes para o portal da educação, pois é sobre estas que incide a marcação de horários para aulas, faltas por parte dos encarregados de educação, etc (pelo menos para uma fase inicial). Estas consistem em diferentes tipos de aulas e disciplinas, principalmente orientadas a crianças mais novas, como Música, Inglês, e até para crianças com idade pré-escolar, no caso de ocupação de tempos livres, etc.

Perfis

Os perfis de utilizador, que serão aprofundados na secção seguinte relativa aos intervenientes, serão um conceito que surge ao longo de todo o projeto e desta dissertação, e é sobre eles que se contempla mais para a frente na análise de requisitos funcionais, a divisão de funcionalidades e apresentação de casos de uso divididos por hierarquia de cada perfil.

Presenças / Faltas

As presenças e faltas são uma parte importante do portal da educação, visto que grande parte das funcionalidades que rodeiam os professores e os funcionários não docentes estão diretamente relacionadas com presenças, que podem existir de vários tipos: podem ser presenças / faltas a uma determinada aula ou serviço, assim como podem ser diárias o que significa uma falta ao dia inteiro, e até faltas que englobem um período delimitado por uma data de início e uma data de fim.

Formulários

No portal da educação deverão existir formulários para todos os perfis. Estes são utilizados para diversas funcionalidades, sendo através deles que se efetuam requerimentos, pedidos especiais, como por exemplo no caso de um encarregado de educação que deseja inscrever o seu educando no serviço de transportes escolares, ou então no caso de um utilizador interno como um professor que deseja submeter um formulário para justificar uma falta dada.

Comunicações

A comunicação direta entre intervenientes também é uma possibilidade sendo que no portal será possível enviar e receber comunicações, que poderão ser enviadas entre perfis, assim como permitir respostas diretas a comunicações recebidas. Estas mensagens serão apresentadas de uma forma simples, quase como uma lista de emails com remetente e destinatário, o corpo da mensagem, anexo e data.

2.2.3 Intervenientes

Os perfis de utilizador são a grande subdivisão da plataforma, uma vez que todos os utilizadores que efetuam o *login* se encaixam num determinado perfil, consoante o seu papel em relação à plataforma. Seguindo uma ordem hierárquica podemos apresentar de uma forma mais aprofundada os vários perfis, que já foram identificados no capítulo anterior.

- P1 – Encarregado de Educação
- P2 - Escola
- P3 – Fornecedor
- P4 – Agrupamento
- P5 – Autarquia
- P6 – Não Docente
- P7 – Professor

Tendo estes aspetos em conta, é preciso que o perfil da autarquia seja o mais completo, com mais poderes de administração e de consulta, edição e até eliminação de dados, e que tem acesso a praticamente tudo desde informação dos utilizadores, horários e faltas dos professores e dos funcionários não-docentes, assim como a toda a configuração do portal, como módulos ativos, notícias da *homepage,* passando por ementas, transportes e todos os requisitos.

Seguindo a hierarquia, o segundo perfil será o perfil Agrupamento, pelo que já existem vários agrupamentos e sendo que o perfil autarquia é um e apenas um por cliente. Este perfil também tem de ter acesso à informação das escolas que lhe estão subordinadas, assim como a capacidade para aprovar faltas e rejeitar, visualizar formulários submetidos pelos diversos utilizadores, assim como muitas outras funcionalidades de alto nível hierárquico. A escola também é um perfil com bastante importância porque tem de gerir tudo o que é referente a si mesma, ou seja, o inventário, serviços de apoio, faltas comunicadas por encarregados de educação, serviços de transporte, ementas etc.

Os perfis de professor e funcionário não-docente são relativamente parecidos no sentido em que é necessário abordar de forma muito mais simples, e ter apenas os horários, mapas de presenças e faltas, capacidade de escrever sumários (no caso dos professores) e pouco mais, visto que já são utilizadores mais comuns do portal.

O perfil de fornecedor existe para servir as necessidades da entidade fornecedora de refeições

2.3 Análise de Valor

2.3.1 The New Concept Development Model

O *New Concept Development Model* (NCD) [3] é um modelo desenvolvido por *Peter Koen* em 2001, que divide a oportunidade em três partes fundamentais, o *Engine* (Motor), a roda, que são as 5 áreas de atividade, relacionadas com o *Front End of Innovation* (FFE) (área interna), e os Fatores Ambientais, que estão representados na figura abaixo como o aro exterior da roda, sob o nome de *Influencing Factors.* [3]

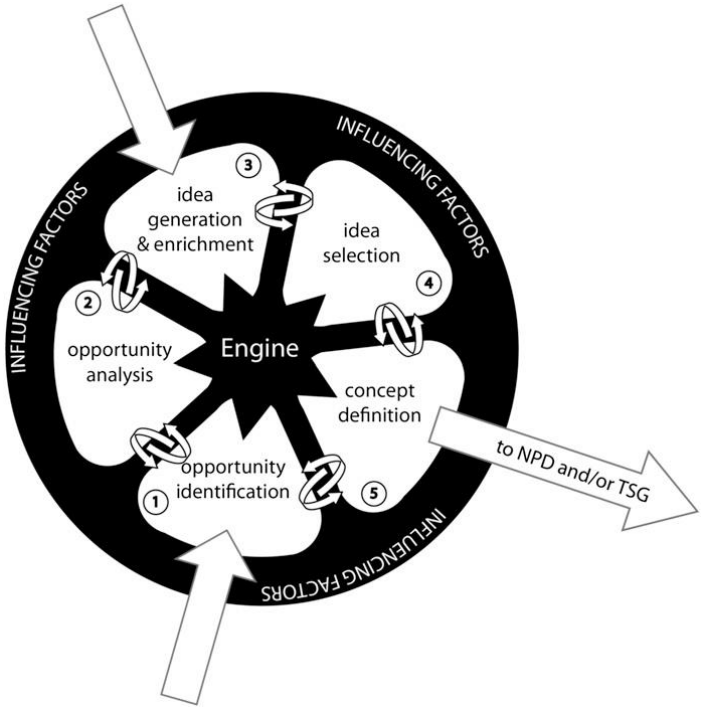


Figura 1 - O modelo *New Concept Development*

Após melhor análise da figura, é possível observar

O motor corresponde aos componentes essenciais que fazem movimentar o processo de *front-end*, tais como a visão, estratégia, recursos, cultura, equipas e colaboração.

A roda, parte interna do modelo, compreende 5 elementos de atividade:

- ***Opportunity identification*** **-** Identificação das oportunidades identificadas, desde a avaliação de fatores económicos, tendências culturais e fatores externos, até à avaliação de ameaças ao negócio atual.
- ***Opportunity analysis*** **–** Análise das oportunidades identificadas, na qual se estuda a probabilidade de sucesso no mercado, através do uso de várias ferramentas como o mapeamento de tecnlogias, planeamento de cenários, etc, contribuindo para uma análise competitiva das vantagens.
- ***Idea generation & enrichment*** **–** Consiste no nascimento e desenvolvimento de uma ideia concreta, em que se trata de um processo evolutivo até chegar a uma solução sólida. Uma nova ideia pode surgir de forma interna ou externa, através de requisitos de clientes, fornecedores, etc.
- ***Idea Selection*** **–** Depois de identificadas as várias ideias, é necessário fazer uma seleção para avançar com o desenvolvimento, de acordo com uma série de critérios de seleção, que têm como objetivo perceber vantagens e desvantagens para que possa haver uma tomada de decisão.
- ***Concept Definition*** **–** Definição final do conceito, apresentada formalmente, onde se evidenciam as vantagens do produto. Já são avaliadas todas as componentes de

mercado e vendas, requisitos técnicos e fatores económicos ligados ao projeto. Este ponto pode ser considerado um sinónimo de construir um plano de negócios.

A parte exterior do modelo (fatores ambientais) são imprevisíveis e têm influência direta no Motor e nas cinco áreas de atividade. Fazem parte destes fatores:

- Gestão de conhecimento
- Tendências de mercado
- Conhecimento e capacidade das tecnologias
- Ameaças de competidores
- Mudança de regulamentos

Aplicando então este modelo ao projeto apresentado neste documento, podemos identificar e explicitar as cinco áreas de atividade:

- ***Opportunity identification*** **–** Não existe ainda uma solução semelhante que permita o nível de integração com o software já presente nas autarquias, pelo que surgiu a oportunidade de desenvolver uma plataforma para a gestão do ambiente escolar que esteja diretamente ligada a toda a informação já presente nas autarquias.
- ***Opportunity analysis*** **–** A principal vantagem desta oportunidade advém do facto de que existe integração direta com os dados e informação proveniente do software das autarquias, o que é extremamente atraente para as autarquias que já são clientes da Medidata. Combinando esta grande vantagem com a necessidade das autarquias de terem uma plataforma completa para gerir a educação, temos uma excelente oportunidade de negócio nas mãos.
- ***Idea generation & enrichment*** **–** Neste caso a ideia foi proposta inicialmente por um cliente da Medidata, fazendo uso da relação já estabelecida como cliente prévio do software de gestão integrada, pois tinha a necessidade de agilizar a forma como é gerida a educação no município, inovando nesse sentido. Consoante a análise dos requisitos a ideia evoluiu para uma plataforma completa, dividida logicamente em vários perfis de utilizadores, com determinado n número de funcionalidades para cada perfil, à partida.
- ***Idea Selection*** **–** Para um projeto desta dimensão é importante que seja feita uma boa seleção de métodos/abordagens e tecnologias, visto que se favorece a rapidez do serviço, a facilidade de utilização para o *end user*, e também a segurança dos dados sensíveis dos utilizadores, logo foi feito um pequeno estudo para perceber qual a abordagem mais apropriada ao projeto proposto.
- ***Concept Definition*** **–** O conceito final para este projeto consiste num Portal da Educação, com diversas funcionalidades das quais se destacam a gestão de horários, gestão de alunos por parte dos encarregados de educação, gestão de faltas, refeições, subsídios escolares, etc. Pretende-se uma plataforma rápida, intuitiva, e segura para todos os intervenientes.

2.3.2 Valor para o cliente e valor percecionado

De modo a entender a importância de um determinado serviço/produto para o cliente, é importante definir conceitos relacionados com o valor de um produto, neste caso o valor, valor para o cliente e valor percecionado.

Valor

O valor de um produto é uma medida qualitativa e que incide fortemente na relevância que este tem para o cliente, e o preço que o cliente está disposto a pagar por ele. Visto que grande parte das receitas de uma determinada organização é proveniente dos seus clientes, é de absoluta importância o foco no valor dos seus produtos/serviços.

O valor pode ser descrito como uma simples proposição que nos indica que maior será o valor de um produto, sistema ou serviço, quanto menor for o custo com o qual é desenvolvido e quanto maior o lucro obtido.

Valor para o cliente

O valor de um determinado produto/serviço para o cliente é diferente do valor geral da solução, sendo que podemos interpretar este conceito como sendo o valor que um produto tem para um cliente em comparação com alternativas possíveis. Ou seja, é importante pensar nesta medição de valor como sendo positiva se o comprador considerar o balanço entre os benefícios e os custos/sacrifícios:

Valor para o cliente = benefícios / custos

Neste sentido, é fundamental entender que um determinado produto ou serviço pode ter valor diferente para clientes diferentes, consoante a sua utilização e experiência com o mesmo, ou com soluções semelhantes. No caso do Portal da Educação, o grande objetivo de o tornar extremamente intuitivo é um enorme benefício que pode fazer aumentar bastante o valor para a maioria dos clientes.

Valor Percecionado

O valor percecionado é o que realmente representa o valor de um produto na prática, e consiste no resultado da diferença entre os benefícios e os sacrifícios para o cliente [4]. No contexto deste projeto, é possível representar os benefícios como o número de funcionalidades, a facilidade de uso e a integração automática com os dados da autarquia, sendo que a curva de aprendizagem para aprender a utilizar a plataforma e o processo de instalação e criação de dados necessários para o funcionamento do portal podem ser considerados como sacrifícios. Esta métrica de valor percecionado, embora parecida com o valor para o cliente, é algo que um determinado cliente pode valorizar mais por preferir a qualidade do produto, enquanto outro cliente pode preferir adquirir o produto/serviço mais económico

2.3.3 Proposta de valor

A proposta de valor tem como objetivo apresentar as razões pelas quais um cliente deve adquirir um produto ou serviço, e deve deixar claro qual o produto em questão, quais os clientes alvo, quais os benefícios que este traz para o cliente, e de que forma difere daqueles já existentes no mercado.

No contexto deste projeto, a proposta de valor é relativamente simples de elaborar: o produto é um portal da educação preparado com vários perfis de utilizadores, e diversas funcionalidades de gestão de horários, faltas, ementas, etc. Como principal cliente surgem as autarquias que já são clientes da Medidata e em segundo plano todas as outras autarquias que estejam à procura de uma solução semelhante, sendo que a integração do sistema com as autarquias já clientes é extremamente fácil, sendo este um ponto forte da nossa solução.

Do ponto de vista do cliente, o que é pretendido é sempre uma prioridade, visto que o portal é desenvolvido com base nos seus requisitos e necessidades, eliminando a grande preocupação se o produto desenvolvido vai de encontro ao que o cliente realmente precisa, de forma eficiente e com boa comunicação.

Embora já existam soluções semelhantes no mercado, e até algumas com um maior nível de complexidade, não há nenhuma que tenha uma integração direta com os dados da autarquia, o que é um dos pontos fortes que faz aumentar a satisfação do cliente, e consequentemente o valor percecionado.

2.3.4 Modelo *CANVAS*

O modelo de negócio *CANVAS* é uma ferramenta que permite que todo o negócio seja visualizado apenas numa única página, sob a forma de um diagrama que compreende nove blocos, de fácil elaboração, análise e compreensão, e indispensável em qualquer tipo de negócio [5]. A figura seguinte representa o modelo *CANVAS* da solução pretendida.

Figura 2 - *Business Model CANVAS*

2.4 Estado da arte

2.4.1 Soluções/abordagens existentes

Tratando-se de uma plataforma que tem como objetivo responder à necessidade crescente da informatização dos sistemas de ensino, naturalmente existem algumas soluções e abordagens bastante interessantes, e que devem ser estudadas para um melhor entendimento do mercado e do valor do produto a desenvolver, pelo que serão aplicados posteriormente uma série de critérios e métricas para perceber melhor qual o estado da arte neste campo.

Escola 360 (E-360)

"O Escola 360, E-360, é um sistema do Ministério da Educação que centraliza os processos de gestão do aluno, desde a educação pré-escolar ao ensino secundário. O objetivo é disponibilizar numa só plataforma toda a informação de caráter administrativo relativa aos alunos.

Com o E-360 pretende-se ainda a facilitar a interação de todos os intervenientes no processo educativo do aluno (encarregados de educação, professores, dirigentes escolares e pessoal administrativo e organismos da administração educativa) que resultará numa maior colaboração, troca de informação mais célere e eficaz, garantindo a segurança de informação." [6]

Através desta citação retiramos a informação de que este software está a ser desenvolvido pelo governo português, e tem como objetivo principal centralizar todos os processos de gestão escolar dos alunos, e todas as funcionalidades inerentes a este objetivo. Também é possível observar a existência de vários perfis de utilizador:

- Encarregados de Educação
- Docentes
- Escolas
- Organismos centrais do Ministério da Educação

A principal diferença desta solução para o portal da educação é que não é desenvolvido com o intuito de apelar a clientes, mas sim como uma ferramenta disponibilizada pelo estado, de forma pública, mas que partilha de qualquer forma muitas das funcionalidades com a solução pretendida, logo tornando-a relevante para o estado da arte neste contexto.

Outras pequenas diferenças passam pelo número inferior de perfis de utilizador como uma desvantagem, e por outro lado a possibilidade de efetuar matrículas na rede de ensino público e de estar ligado às escolas a nível nacional como vantagens a ter em conta.

Inovar Alunos

Outra solução bastante semelhante e que se adequa ao estado da arte, é o Inovar Alunos da empresa inovar+, que também apresenta ser uma solução completa para toda a gestão escolar, possuindo um vasto leque de funcionalidades: [7]

- Gestão de currículos
- Matrículas automáticas
- Informação em tempo real
- Sumários
- Avaliações
- Apoios
- Atas
- Comunicações
- Direção de turma
- Caracterização da turma
- Tratamento de dados
- Relatórios de final de período
- Produção de indicadores para a tomada de decisão
- Assiduidade e pontualidade docente
- Acessos configuráveis
- Agrupamentos
- Comunidade Escolar
- Integração

"o INOVAR serve para tudo, para a área de alunos, área administrativa, área financeira, o PAA… Neste momento o INOVAR já serve para uma panóplia enorme de gestão." [8] Desta citação conseguimos concluir que realmente esta solução é extremamente completa, e um produto com boa reputação e bastante tempo no mercado, logo um exemplo a seguir em termos de valor e de valor para o cliente.

Em relação ao modelo de negócio, e ao contrário do exemplo anterior do E-360, esta solução é desenvolvida por uma empresa particular com vista a ser adquirida por clientes, pelo que já pode existir uma comparação mais direta com a solução a desenvolver. Isto é sem dúvida um fator que pode ajudar a perceber e enquadrar a solução do portal da educação quer em termos de valor, quer em questões de design mais aprofundado.

A interface do inovar pode parecer antiquada, mas acima de tudo é completa, e funcional, embora pudesse ser um pouco mais amigável para o utilizador:

Figura 3 - Exemplo da interface gráfica do inovar alunos

Edubox SIGA

Das soluções apresentadas neste capítulo do estado da arte, surge aqui o principal concorrente ao portal da educação, uma plataforma denominada SIGA (Sistema Integrado de Gestão e Aprendizagem) da empresa Edubox, que segundo a descrição oficial do *website* é uma plataforma que permite ao município gerir, online e de forma rápida e eficaz os vários processos educativos. [26]

À semelhança do portal da educação, esta solução é desenvolvida com vista a ser adquirida pelas autarquias, visa ser modular e personalizável, fácil de utilizar e com boa performance, que são alguns dos principais objetivos.

Este sistema possui diversas funcionalidades nas seguintes áreas:

- Ação Social
- Refeições
- Curricular
- Transportes
- Espaços e Parque Escolar
- Pagamentos

- Atividades de Enriquecimento Curricular
- Notificações
- Candidaturas Online
- Material Escolar

É de notar que dentro destas áreas agrupam-se várias funcionalidades relacionadas, sendo estas áreas apenas os vários módulos do sistema, o que indica uma grande capacidade de adaptação do sistema às necessidades dos seus clientes.

Além do conteúdo apresentado nesta breve análise de estado da arte, ainda é possível consultar no *website* alguns testemunhos de clientes, assim como um vídeo demonstrativo, que evidenciam o valor deste tipo de solução e a importância em criar soluções que resolvam o problema apresentado nesta dissertação.

2.4.2 Tecnologia relevante

2.4.2.1 Docker

O *Docker* é uma ferramenta criada para facilitar a criação, implantação, e correr aplicações através do uso de contentores. Os contentores permitem a um programador juntar todas as peças das quais uma aplicação necessita, e distribuir essa aplicação apenas num "pacote". Ao fazer isto, garante que a aplicação corre em qualquer outra máquina de *Linux*, independentemente de quaisquer configurações personalizadas que essa máquina possa ter, que podem ser diferentes das da máquina utilizada para escrever e testar o código.

De uma certa forma, o *Docker* é como uma máquina virtual, mas em vez de virtualizar um sistema operativo inteiro permite às aplicações utilizar o mesmo *kernel* de *Linux* do sistema onde estão a correr, e o facto de já virem com todos os componentes necessários para o seu correto funcionamento é uma mais-valia em termos de performance.

Esta ferramenta beneficia diferentes tipos de utilizadores, sendo que permite aos programadores escreverem código sem se preocuparem com a máquina onde a aplicação será executada, e também lhes providencia com um avanço devido aos milhares de programas já desenhados para correr num contentor de *Docker* como parte da sua aplicação. Também oferece flexibilidade e reduz o número de sistemas que os administradores precisam de monitorizar, ou gerir. [9]

No contexto do portal da educação apresenta uma grande vantagem além das que já foram referidas, que é a possibilidade de permitir disponibilizar aos clientes pacotes de instalação sob a forma de contentores de *Docker*, tendo apenas de correr dois ou três comandos após ter o servidor preparado

2.4.2.2 Docker-Compose

O *docker-compose* é uma ferramenta para definir e correr aplicações "multi-contentor" de *Docker*. Utilizando um ficheiro YAML (*Yet Another Markup Language)* para configurar os serviços e variáveis da aplicação, é então possível, utilizando um único comando, correr, reiniciar e para todos os serviços ao mesmo tempo de uma forma muito acessível. [14]

Esta tecnologia funciona em vários ambientes, quer de produção como de testes e desenvolvimento, e suporta *workflows* de CI/CD (*Continuous Integration / Deployment*). Das restantes principais funcionalidades destacam-se a possibilidade de definir variáveis de ambiente na configuração, utilizar imagens já existentes (como por exemplo de uma base de dados SQL), e isolar ambientes num único anfitrião.

2.4.2.3 Git

Git é um sistema de controlo de versões grátis e *open source*, desenvolvido com o objetivo de lidar com todo o tipo de projetos, desde pequenos projetos até projetos de larga escala, de forma rápida e eficaz.

É uma das tecnologias mais utilizadas no desenvolvimento de software, visto que facilita trabalho colaborativo, regista o histórico de alterações, permite vários *branches* (múltiplos ramos de desenvolvimento simultâneos), a integração de alterações provenientes de diferentes programadores, sendo por estes motivos extremamente relevante no âmbito deste projeto. [27]

2.4.2.4 Meta

Meta é uma ferramenta desenvolvida para ser utilizada com repositórios Git, que permite a gestão de sistemas multi-projeto, eliminando o problema de decisão entre um único repositório para todo o sistema ou vários repositórios, conjugando as duas possibilidades com um "meta" repositório. [28]

De uma forma resumida, é possível ter um repositório "pai" e vários repositórios "filho" dentro deste, sem ter de gerir cada um individualmente. As vantagens desta tecnologia incluem a possibilidade de clonar todos estes repositórios com um só comando, dar a todos os engenheiros da equipa o mesmo *setup*, utilizar as familiares ferramentas do Git sem qualquer necessidade de aprendizagem, e ainda instalar / gerir dependências para todos os subprojetos, entre outras funcionalidades.

2.4.2.5 Node.js

O *Node.js* é um *event-driven runtime* de *JavaScript*, assíncrono, e desenhado para construir aplicações de rede escaláveis, de forma eficiente, e sem muitas das preocupações que modelos baseados em concorrência e bloqueios implicam [10]. Em *Node.js* não há bloqueios, logo é extremamente adequado para desenvolver sistemas muito escaláveis, como é o caso do portal da educação.

O servidor de *backend* necessita de ser uma solução robusta, rápida, segura e facilmente personalizável, pelo que no contexto deste projeto, *Node.js* é uma das tecnologias mais indicadas.

2.4.2.6 Express.js

Express.js é uma *framework* para a tecnologia descrita no ponto acima, *Node.js.* Esta visa ser uma *framework* minimalista, flexível e de alto desempenho, com vista ao desenvolvimento de aplciações *web*, como por exemplo uma *REST API.*

Além do que já foi enumerado, esta dispõe de um forte conjunto de ferramentas e métodos HTTP, assim como ferramentas de roteamento, sempre sem obscurecer as funcionalidades de *Node.*

2.4.2.7 React.js

O *React.js* é uma biblioteca de *JavaScript* para construção de interfaces gráficas, com várias características interessantes:

- É declarativo, o que permite criar interfaces interativas muito facilmente, desenhando vistas simples para cada estado da aplicação, e deixando o resto ao encargo do *React*, que atualiza os componentes certos aquando da mudança de dados.

- É baseado em componentes, pelo que é possível construir componentes encapsulados que gerem o seu próprio estado, e combiná-los para construir interfaces complexas, já para não falar do extenso leque de componentes e módulos desenvolvidos por milhares de programadores por todo o mundo. [11]

Estas características tornam o *React.js* a ferramenta ideal para uma interface baseada em *web*, sendo que o portal da educação pode ser acedido online a partir de qualquer browser, permitindo criar uma interface agradável, rica e completa com relativa facilidade.

2.4.2.8 PostgreSQL

O *PostgreSQL* é uma base de dados relacional *open-source*, muito completa, com mais de 30 anos de desenvolvimento ativo, o que lhe dá uma grande credibilidade, funcionalidades robustas e boa performance. [12]

Sendo uma base de dados relacional, e tão bem estabelecida, perfaz as necessidades do portal da educação, em que é necessário existir um modelo relacional, de acordo com a natureza intrínseca dos dados a serem trabalhados, logo é uma grande opção a ser considerada.

2.4.2.9 Sequelize

O *Sequelize* é um ORM (*Object-Relational Mapping)* para bases de dados relacionais, como a descrita acima, que utiliza a funcionalidade de *promises* presente no *Node.js*. ORM é uma técnica para converter dados entre sistemas incompatíveis, mapeando os “objetos” do

software para dados que possam ser armazenados numa base de dados, tudo utilizando a linguagem de programação original. Isto permite ao programador mapear todos os dados da base de dados virtualmente sem ter de escrever SQL manualmente, tirando raras exceções, visto que tudo isso é gerado a partir do mapeamento efetuado.

As suas principais vantagens são o extenso leque de possibilidades e funcionalidades, o sólido suporte para transações, relações entre objetos, os diferentes tipos de carregamentos de dados, sem esquecer de mencionar a documentação extremamente completa. [30]

3 Análise e Design

Para que seja possível apresentar uma solução sólida que responda da melhor forma possível ao problema, é necessário um planeamento composto por diversas partes. O objetivo deste planeamento é definir como é que a solução final será desenvolvida de acordo com os problemas identificados e o estado da arte.

Durante a fase de análise e desenho são desenvolvidos os diferentes artefactos que definem toda a plataforma do Portal da Educação, servindo como base para a implementação da solução. É importante notar que os diagramas apresentados serão obtidos através da especificação *Unified Modeling Language* (UML) [16].

3.1 Domínio do Problema

De acordo com a especificação apresentada pelo cliente, e após análise do problema e do objetivo, são então identificados um conjunto de conceitos de negócio, que englobam regras, funcionalidades, condições e restrições que a solução deve respeitar e refletir. Grande parte destes conceitos foram extraídos das especificações enviadas pelo cliente, pelo que é possível consultar uma versão antiga e simplificada de um desses documentos no final do documento, Anexo A.

3.1.1 Conceitos de Negócio

Aqui serão apresentados os conceitos mais importantes do negócio, e que serão cruciais para desenvolver as entidades pertencentes ao modelo de domínio:

Utilizador – Utilizador final da plataforma.

Perfil – Perfil(s) associado(s) ao utilizador, de entre vários existentes.

Autarquia – Perfil correspondente à autarquia, que é também administrador da plataforma.

Informação – Informações referentes ao portal numa dada autarquia, geridas pela mesma.

Manual Instrução – Manuais de instrução referentes ao portal disponibilizados e geridos pela autarquia.

Configuração – Configurações diversas do portal, que incluem a personalização, datas-limite para justificações e submissões, módulos bloqueados, configurações de formulários e etc.

Agrupamento – Perfil correspondente aos agrupamentos.

Escola – Perfil correspondente às escolas, que engloba os horários dos vários não docentes e tem ementas.

Ementa – Informação referente à ementa de uma determinada escola, que pode ser consultada.

Professor – Perfil correspondente ao professor, que pode lecionar em várias escolas e tem horários.

Falta – Representa as faltas dadas pelos professores e funcionários não docentes.

Não Docente – Perfil correspondente ao funcionário não docente, que pode estar vinculado a vários agrupamentos e escolas, e também tem horários.

Horário – Horários dos professores e dos não docentes, englobam um conjunto de horas e de atividades extracurriculares e contém informações sobre aulas, etc.

AEC – Atividades extracurriculares que são frequentadas pelos educandos e estão presentes nos horários.

Encarregado de Educação – Representa o perfil dos encarregados de educação, que podem fazer parte de um ou mais agrupamentos, dependendo dos seus educandos.

Educando – Aluno que pertence a um encarregado de educação, beneficia de serviços de transporte / ação social, e tem presenças.

Serviços – Diversos serviços de ação social e de transporte público dos quais o educando pode beneficiar, após requerimento via formulário.

Presença – Presenças referentes aos educandos, a não confundir com as faltas do pessoal docente e não docente.

3.1.2 Modelo de Domínio

O modelo de domínio é uma representação visual de todas as entidades necessárias para o funcionamento da plataforma assim como todas as relações entre elas, o que nos proporciona uma estrutura base para a fase da implementação da solução. É também um passo importante

na transição de requisitos formalizados em linguagem natural para especificações precisas, aplicando regras simples que nos permitem extrair informações importantes [15].

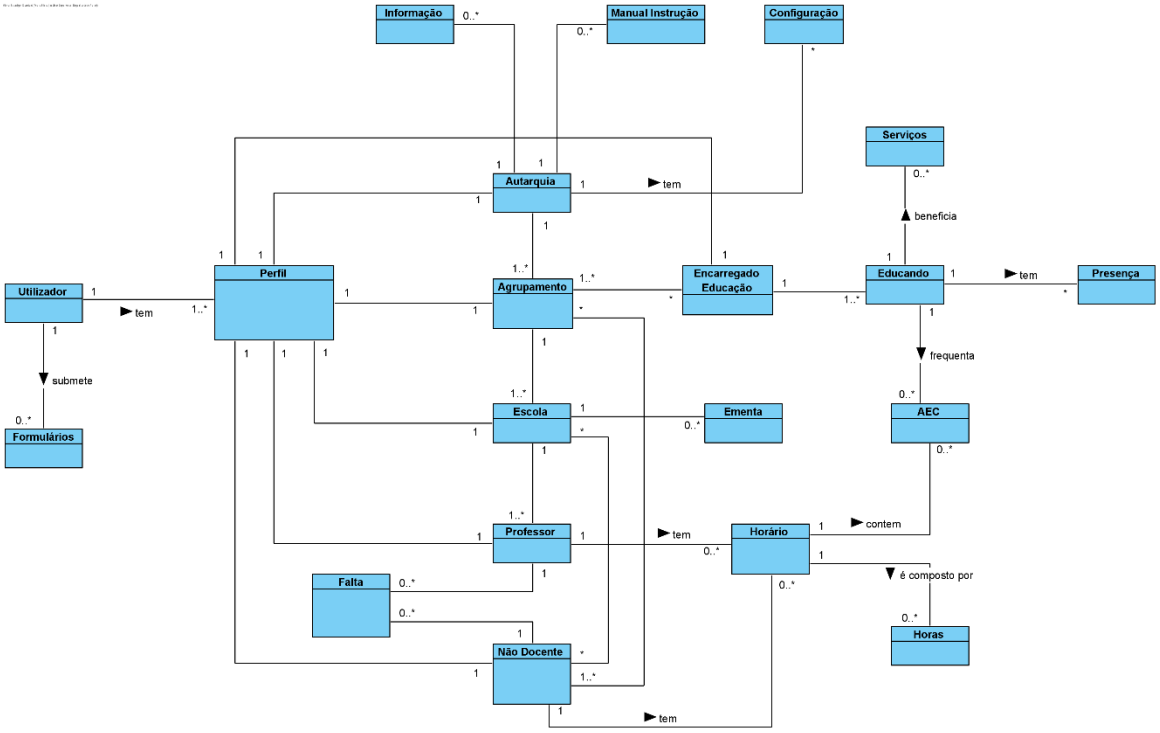


Figura 4 - Modelo de Domínio do portal da educação

3.2 Requisitos Funcionais e Não Funcionais

Estabelecendo continuidade aos conceitos de negócio apresentados no capítulo anterior, identificamos vários requisitos que implicam diversas funcionalidades. De modo a identificar funcionalidades e casos de uso relevantes tendo em conta os utilizadores da plataforma, será feita uma análise de requisitos funcionais e não funcionais, que são propriedades que o sistema deve possuir com o objetivo de satisfazer as necessidades descritas pelo cliente.

3.2.1 Requisitos Funcionais

Os requisitos funcionais captam o comportamento que se espera do sistema. Este comportamento pode ser definido através de variados serviços, tarefas ou funções. [17] De forma a apresentar os requisitos funcionais recorreu-se à elaboração de várias vistas de funcionalidades sob a forma de diagramas de casos de uso, de acordo com cada perfil da plataforma. Embora de uma forma abrangente, retira-se uma boa perceção sobre a forma da solução final, quer do ponto de vista de experiência para o utilizador assim como da estrutura do próprio portal em termos de implementação.

Para finalizar a apresentação dos requisitos funcionais, será estudado um caso de uso de uma forma mais aprofundada no final desta secção, que será também explicado e fundamentado na secção da implementação.

Antes de apresentar as funcionalidades de cada perfil, é importante referir que todos os tipos de perfis têm algumas funcionalidades em comum, sendo que algumas são comuns a todos os perfis, pelo que são apresentadas antes de qualquer perfil em específico, tendo como ator um "utilizador geral":

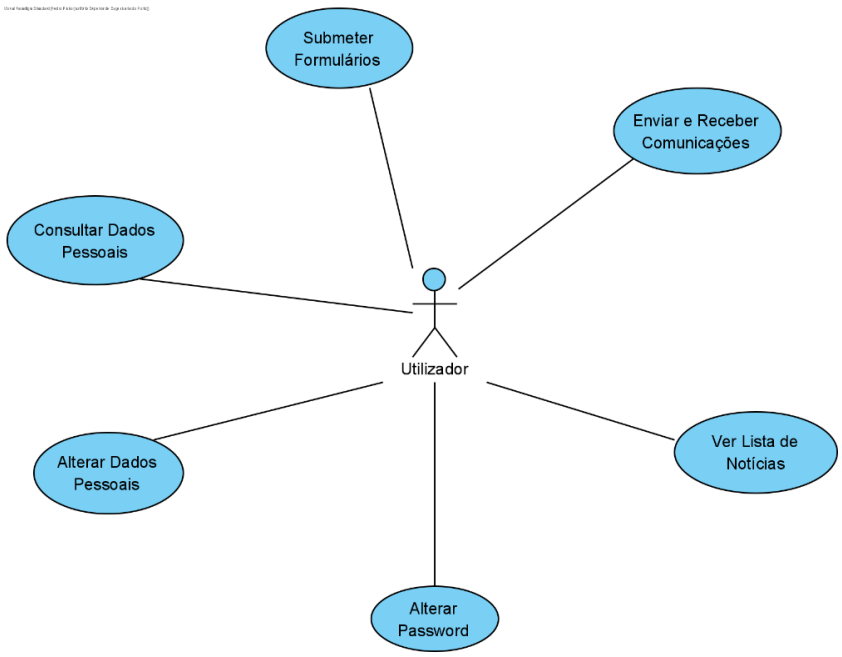


Figura 5 - Diagrama de Casos de Uso transversais a todos os perfis

Autarquia

A autarquia é o perfil mais abrangente de todos, pelo que é, naturalmente, o perfil que tem acesso a mais funcionalidades, incluindo as de administração e configuração da plataforma, assim como permissões para gerir (editar e eliminar) além de apenas consultar como é o caso da maioria dos perfis.

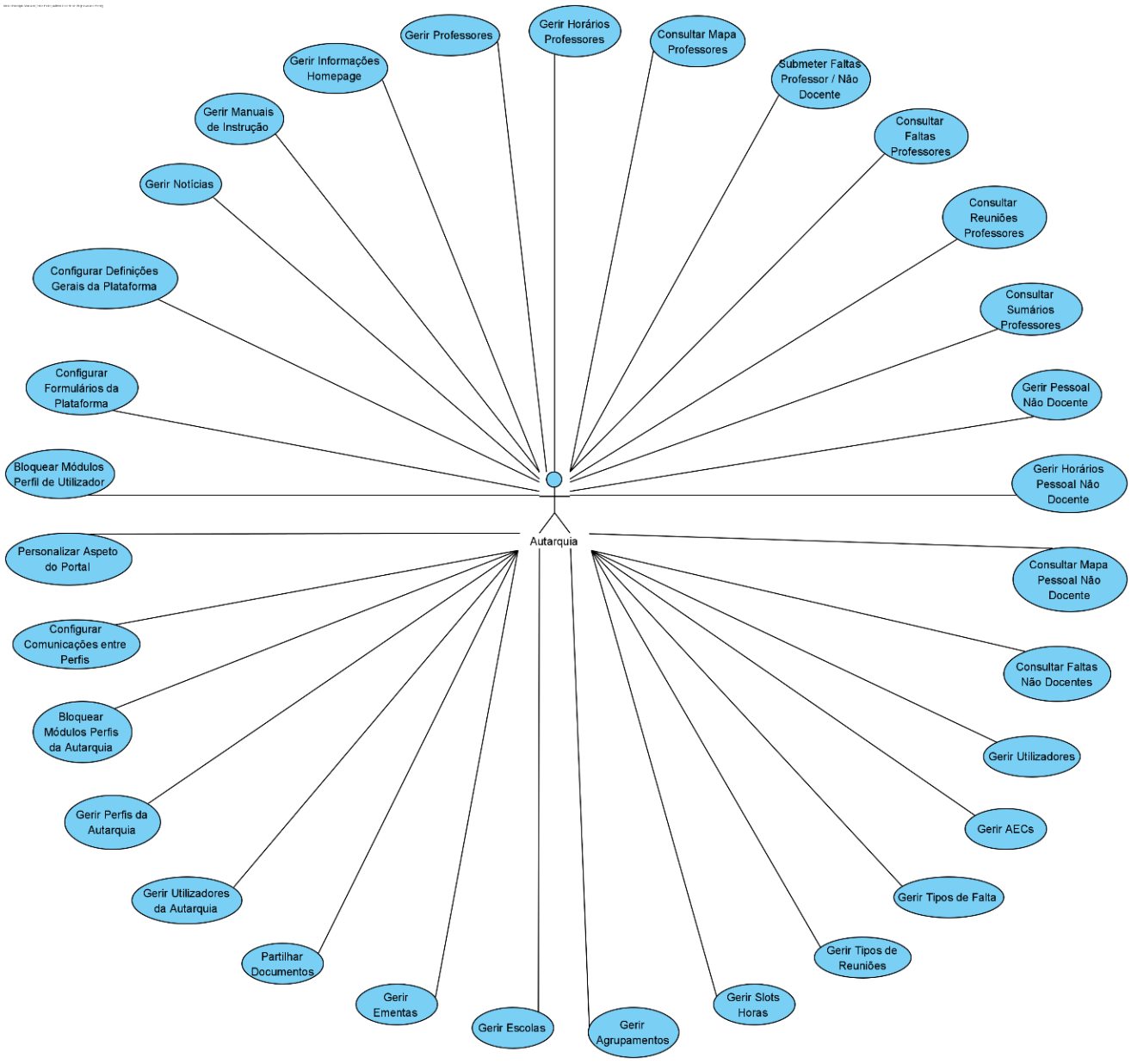


Figura 6 - Diagrama de Casos de Uso Autarquia

Agrupamento

O perfil Agrupamento é o próximo perfil na hierarquia, pelo que uma autarquia pode englobar um ou mais agrupamentos, tendo também bastantes funcionalidades de mais alto nível de privilégio, perdendo as funcionalidades de configuração e gestão de plataforma.

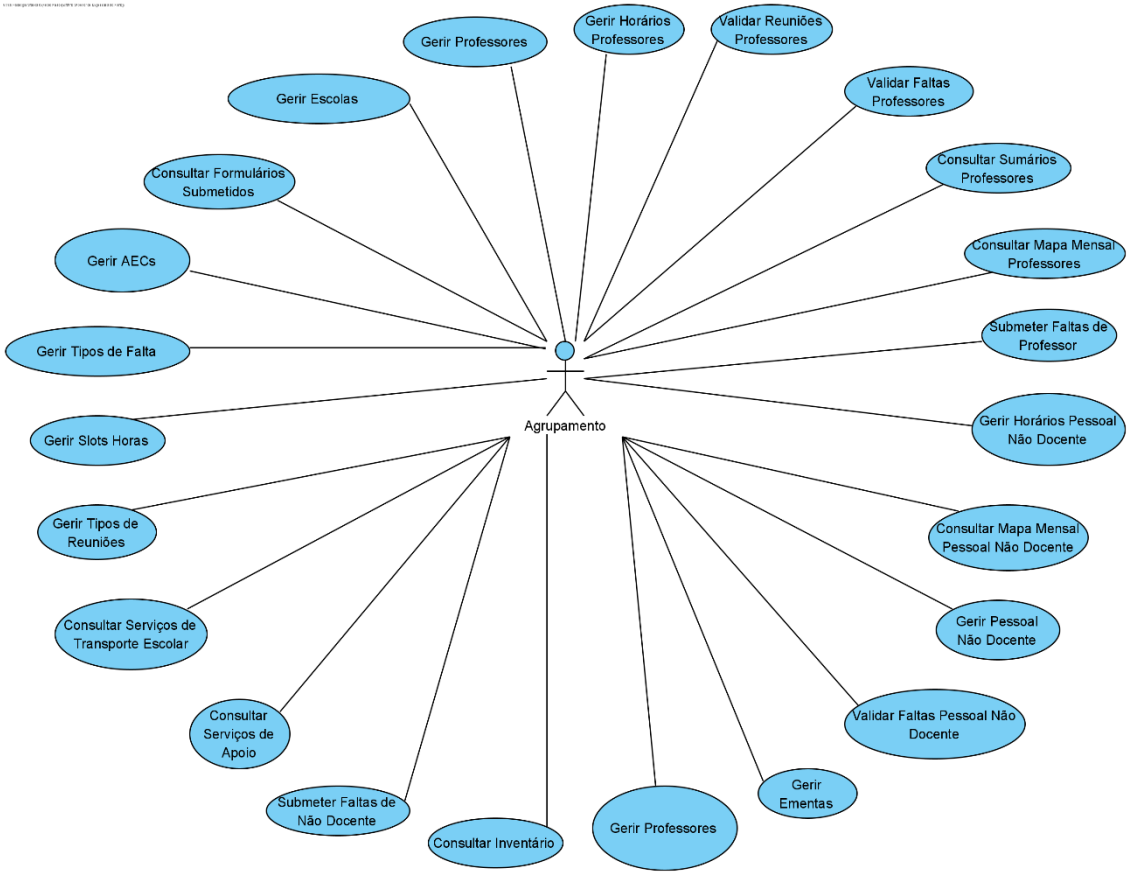


Figura 7 - Diagrama de Casos de Uso Agrupamento

Escola

De forma lógica, segue-se o perfil Escola, visto que um Agrupamento tem uma ou mais escolas, sendo que continuamos a reduzir o leque de funcionalidades, desta vez com um âmbito mais específico para os interesses da própria escola.

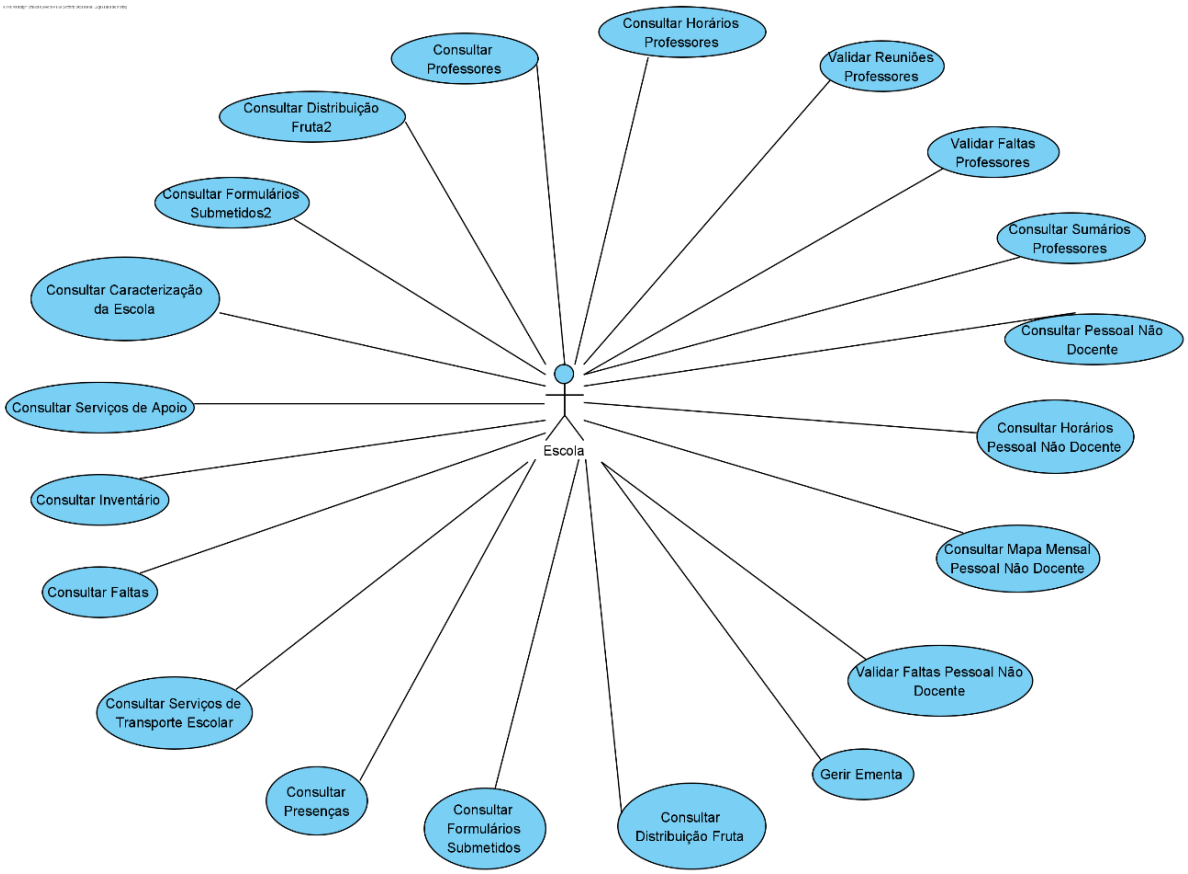


Figura 8 - Diagrama de Casos de Uso Escola

Professor

O perfil de Professor é o próximo a ser analisado visto que uma escola tem vários professores a cumprir nela horários. Encontramos aqui funcionalidades típicas de um docente, como a possibilidade de sumariar aulas, marcar faltas, e consultar os seus horários.

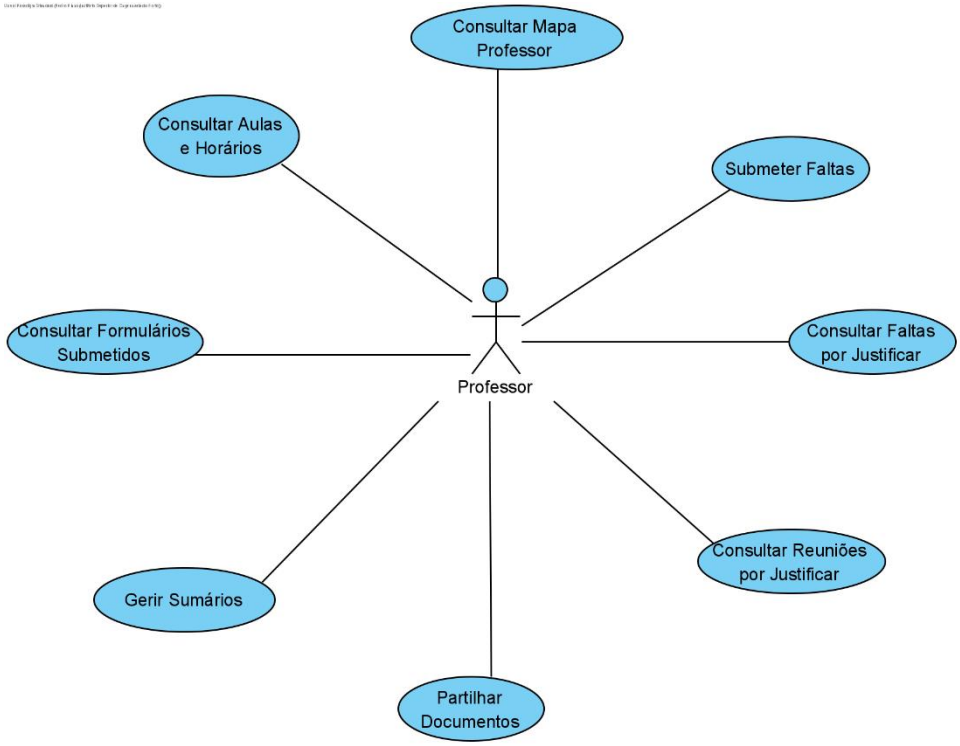


Figura 9 - Diagrama de Casos de Uso Professor

Pessoal Não Docente

Pessoal Não Docente (funcionários e etc.) também é englobado pelo âmbito da escola, e é o perfil com menor número de funcionalidades, devido às poucas necessidades do utilizador neste sentido.

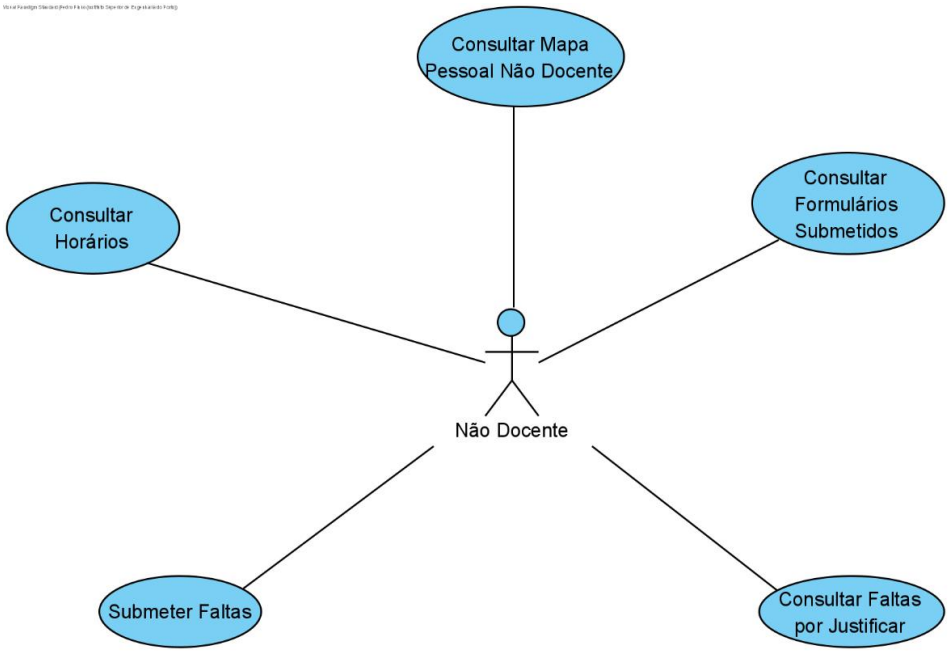


Figura 10 - Diagrama de Casos de Uso Pessoal Não Docente

Encarregado de Educação

Dos perfis mais importantes de todo o Portal da Educação, e o que terá maior número de instâncias devido à quantidade de encarregados de educação que existem, em proporção com os alunos das várias escolas dos vários agrupamentos da autarquia.

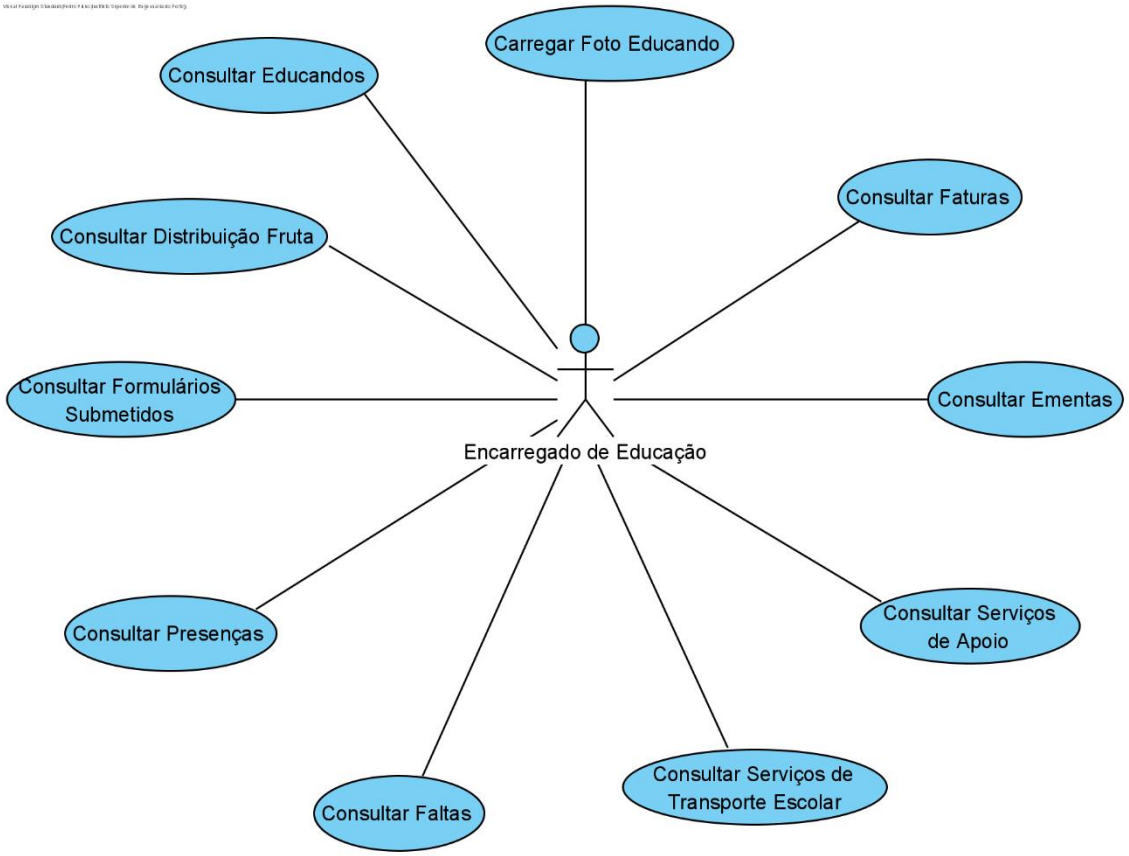


Figura 11 - Diagrama de Casos de Uso Encarregado de Educação

Depois de analisados e identificados os principais casos de uso com base na categorização por perfil, serão apresentados alguns casos de uso tidos como muito importantes para a solução, de uma forma aprofundada, o que servirá como exemplo prático para o resto das funcionalidades. Como referido acima, um destes mesmos casos de uso será depois utilizado novamente e aprofundado na parte da implementação, servindo como fio condutor para demonstrar a implementação da solução e visto que faz uso de todos os recursos da plataforma, torna-se uma excelente forma de seguir uma lógica que procura aprofundar mais e mais a cada capítulo.

3.2.1.1 Login

O primeiro UC a apresentar é o de iniciar sessão no portal. Este caso de uso foi escolhido para uma análise mais aprofundada devido ao facto de ser transversal a todos os componentes da plataforma, e poder desta forma dar ao leitor uma maior compreensão sobre o design do sistema.

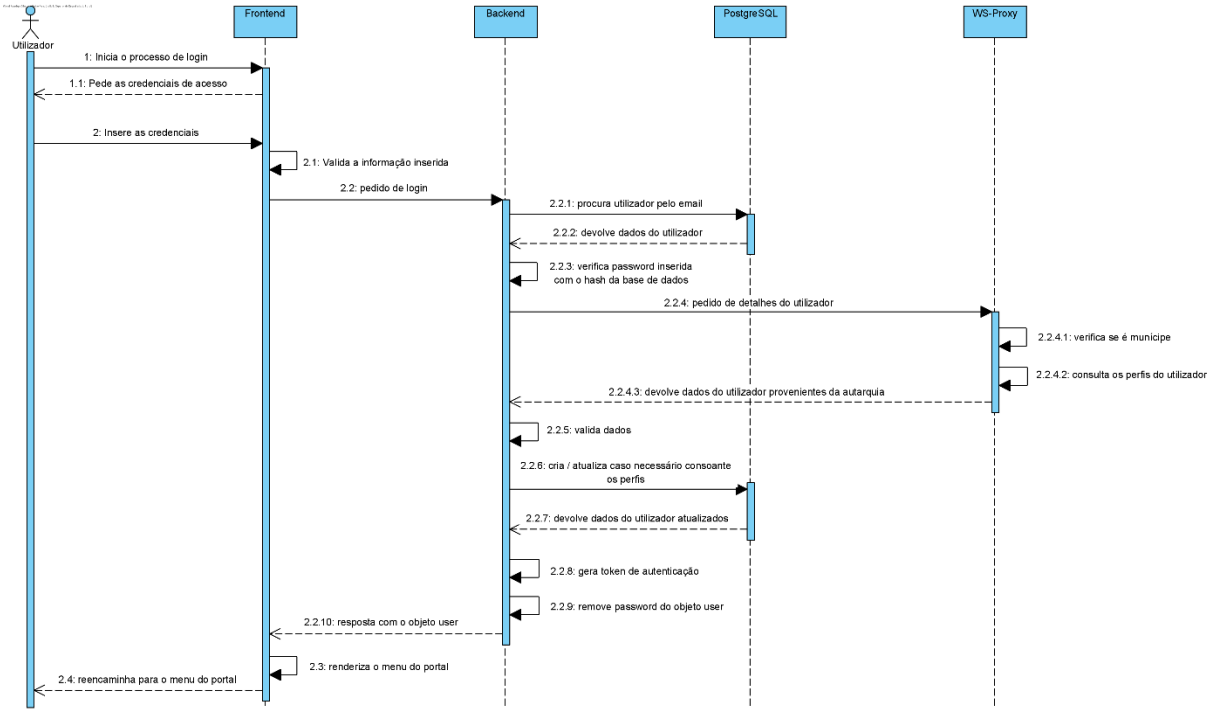


Figura 12 - Diagrama de sequência do login

3.2.1.2 Casos de uso extra

Aqui serão incluídos dois fluxogramas conceptuais, elaborados como uma possível alternativa de design de várias funcionalidades.

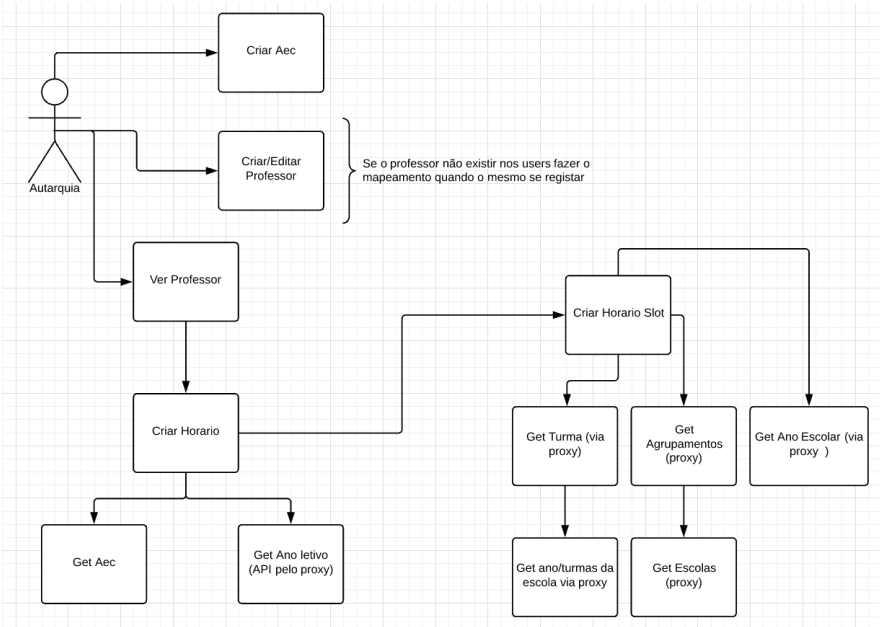


Figura 13 - Fluxograma parcial do perfil autarquia

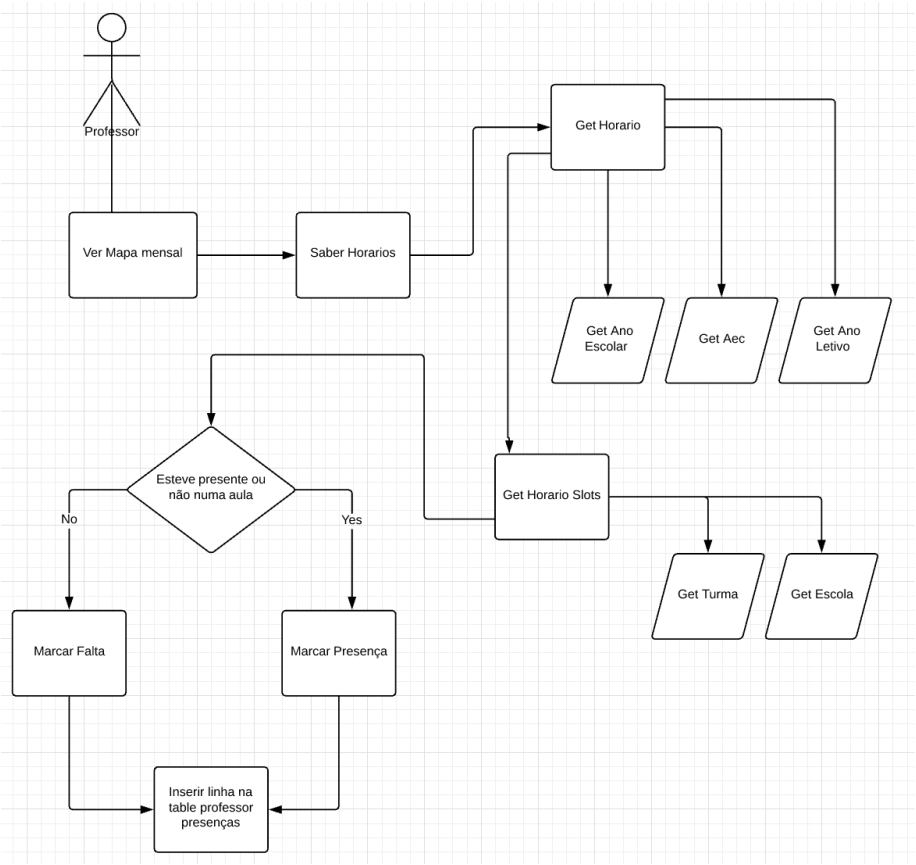


Figura 14 - Fluxograma parcial do perfil professor

3.2.2 Requisitos Não Funcionais

Após identificar todos os requisitos funcionais que ajudam a interpretar o que o sistema irá fazer, procedeu-se à identificação dos requisitos não funcionais. Os requisitos não funcionais são todos os requisitos relacionados com a utilização da aplicação, logo não estão diretamente relacionados com as funcionalidades de uma aplicação, mas que impõe várias restrições sobre o seu desenvolvimento. Este passo é considerado essencial para o desenvolvimento de software destas dimensões. [20]

De modo a apresentar os requisitos não funcionais, recorreu-se à utilização do modelo FURPS+, devido ao seu reconhecimento e aprovação por parte da comunidade. [25] Este modelo foi criado pela *Hewlett-Packard.* Este é um acrónimo para *functionality* (funcionalidade), *usability* (usabilidade), *reliability* (confiabilidade), *performance* (desempenho) e *supportability* (suportabilidade), em que o + identifica outros requisitos resultantes da evolução do modelo inicial, incluindo restrições de design, de implementação, de interface e ainda restrições físicas.

3.2.2.1 Usabilidade

A usabilidade é a capacidade que o sistema tem de garantir ao utilizador sucesso na execução de uma dada funcionalidade. Vários aspetos como a acessibilidade, estética, facilidade de uso e consistência da interface do utilizador, contribuem para melhorar a usabilidade geral do sistema. No contexto deste projeto identificou-se o seguinte:

- A interface gráfica do sistema deve ser intuitiva e de fácil navegação;
- O tempo de resposta das várias partes do sistema deve ser o mais rápido e eficiente possível;
- A interface deverá apresentar consistência nos vários menus e elementos utilizados.

3.2.2.2 Fiabilidade

A fiabilidade inclui aspetos como o *uptime* da plataforma, taxa de recuperação de erros, assim como o número e gravidade de erros admissível num sistema desta dimensão.

- O acesso ao servidor deve permanecer funcional e disponível após falhas;
- Idealmente o sistema deve estar sempre acessível, não só livre de erros, mas também ao estar em baixo o mínimo de tempo possível durante uma atualização;
- Deve existir uma verificação periódica de funcionalidades para assegurar o correto funcionamento da plataforma.

3.2.2.3 Desempenho

Devido à dimensão do projeto, à quantidade de interações e pedidos feitos pelos milhares de utilizadores do portal, o desempenho deverá ser dos principais focos de desenvolvimento. Este parâmetro reflete-se no consumo de recursos, e na rapidez e tempo de resposta de todas as interações. Embora no âmbito do portal não existam requisitos específicos em termos de tempos de resposta, etc. identificam-se de qualquer forma várias necessidades de desempenho:

- O tempo de carregamento das páginas não deve deixar o utilizador demasiado tempo à espera;
- Os pedidos à API devem apenas consumir informação relevante, de forma a aumentar o desempenho;
- O portal deverá suportar múltiplos acessos concorrentes sem degradar a experiência do utilizador.

3.2.2.4 Suportabilidade

Nesta categoria enquadram-se vários tipos de requisitos como a testabilidade, adaptabilidade, facilidade de manutenção, escalabilidade, entre outros:

- O sistema deve ser facilmente expansível e permitir a adição de novos módulos / funcionalidades;
- O acesso ao sistema deve ser versátil e permitir qualquer dispositivo móvel ou fixo;

- Deverá ser criado uma plataforma de testes para que seja possível identificar o máximo número de problemas antes de chegar ao utilizador final.

3.2.2.5 Restrições de Design

Uma restrição de design limita a forma como o sistema é planeado. Para o projeto em questão:

- A informação já existente relativamente aos munícipes, alunos, etc. é responsabilidade da autarquia e deverá ser daí consumida;
- Deverá ser implementada uma base de dados relacional (justificação deste requisito na secção que cobre a implantação e a base de dados);
- A solução desenvolvida deve ser instalada num servidor externo, providenciado pela autarquia.

3.2.2.6 Restrições de Implementação

Esta categoria impõe restrições à forma como será implementada a solução, em termos de codificação e construção do sistema, como padrões de desenvolvimento ou linguagens de programação. Depois de analisadas as tecnologias relevantes na secção do estado da arte, para o seguinte projeto identificam-se as seguintes restrições, que serão descritas textualmente por não se adequarem ao formato de lista.

Para codificação do *frontend* será utilizado *React.js*, não só pela familiaridade da linguagem de programação *JavaScript*, assim como a facilidade em desenvolver módulos reutilizáveis, assim como a possibilidade de utilizar módulos da comunidade, assim como ferramentas de estilo de interface, etc. Deve também ser altamente modular de forma a permitir acrescentar ou retirar funcionalidades, estando desta forma dividido em camadas de vistas e contentores.

O *backend* será codificado também em *JavaScript* fazendo uso da plataforma *Node.js* e da *framework* minimalista e de alto desempenho para web, *express.* Seguirá as regras adequadas a uma REST API, sempre com o objetivo de ser o mais eficiente possível, quer para os utilizadores, quer para facilidade de desenvolvimento.

O tipo de base de dados que se adequa melhor ao tipo de dados é relacional, pelo que será esta implementada no projeto a ser desenvolvida em *PostgreSQL*, visto ser gratuito e não haver diferenças significativas de desempenho ente este e outros sistemas. Será justificado também na parte da arquitetura a escolha de uma base de dados desta natureza.

3.2.2.7 Restrições de Interface

Este tipo de restrições refletem os requisitos no que diz respeito à interação com sistemas externos ao sistema, neste caso será fortemente dependente da autarquia:

- Os dados da autarquia serão provenientes de uma API REST disponibilizada, e consumidos pelo portal da educação sempre que necessário.

3.2.2.8 Restrições Físicas

As restrições físicas representam os limites físicos e de *hardware* onde o sistema estará hospedado, como tamanho peso e forma. Dado que o servidor é externo e da responsabilidade das autarquias, não se identificou nenhum requisito desta natureza.

3.3 Decisões de Design

Nesta secção serão abordadas várias decisões tomadas ao longo deste processo de design, de uma forma intermédia entre a análise dos requisitos e a definição da estrutura arquitetural.

3.3.1 A base de dados

A base de dados é responsável por suportar toda a informação necessária para o correto funcionamento do portal. Devido ao tamanho da solução, foi necessário decidir qual o tipo de base de dados mais apropriado para o sistema, e optou-se por uma base de dados relacional (SQL), que define relacionamentos sob a forma de tabelas. A conjugação de uma base de dados SQL com o ORM a ser utilizado facilita o uso de funções CRUD (*Create, Read, Delete, Update*). Para além disso pode-se dizer que é verticalmente escalável, de acordo com as necessidades de tráfego.

Claro que existem outras alternativas como uma base de dados *NoSQL,* ou base de dados não relacional, como por exemplo *MongoDB.* Este tipo de base de dados não depende de um esquema físico e é horizontalmente escalável, e evita ligações e associações. Acontece que numa aplicação com muito fluxo de transações e serviços mais pesados, as bases de dados relacionais são recomendadas porque seguem as propriedades ACID, que garantem Atomicidade, Consistência, Isolação e Durabilidade. Por outro lado, as bases de dados não relacionais usualmente seguem as propriedades BASE (*Basically Available, Soft State, Eventually Consistent*), [31] como é possível verificar na figura seguinte.

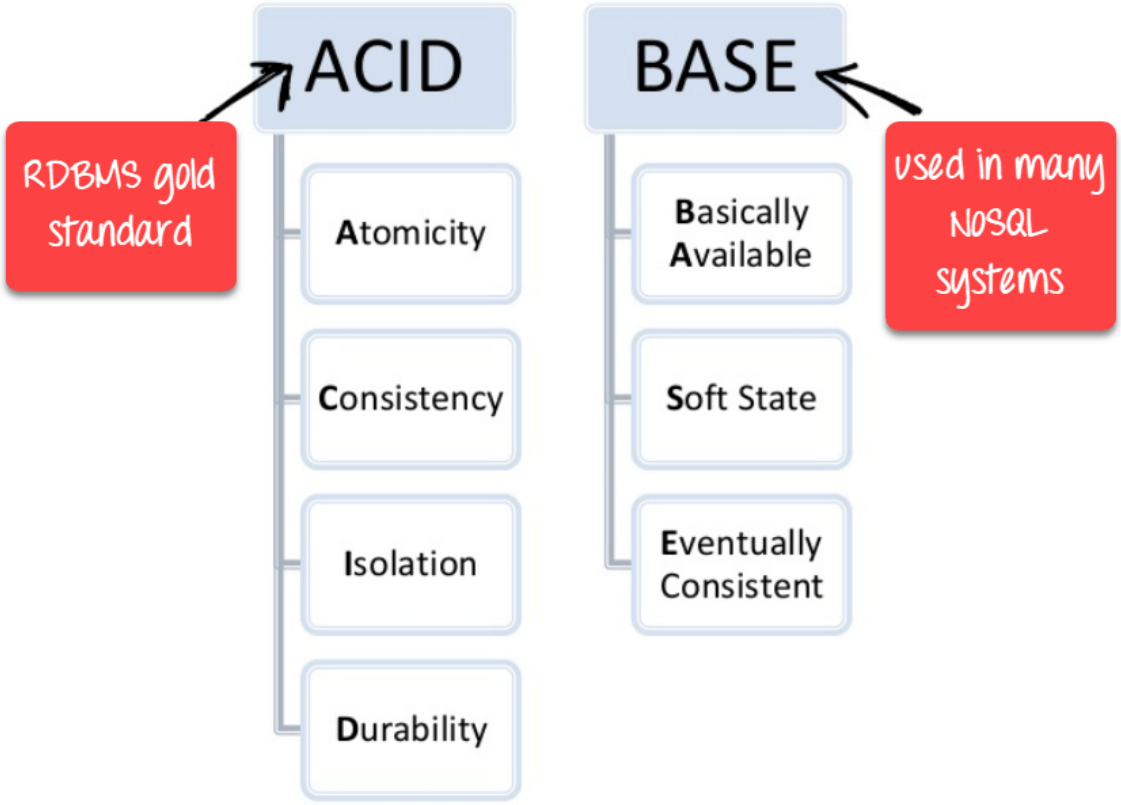


Figura 15 Propriedades ACID vs BASE (retirado de [32])

É apresentado, na figura 16, o modelo de dados da solução. Através de uma análise rápida do modelo é possível compreender a dimensão de dados com que estamos a trabalhar. Também se torna mais fácil compreender a decisão de optar por uma base de dados relacional. É possível ver as tabelas intermédias que são utilizadas quando existe uma relação “muitos-para-muitos”, como é por exemplo o caso das faltas diárias e dos horários dos professores.

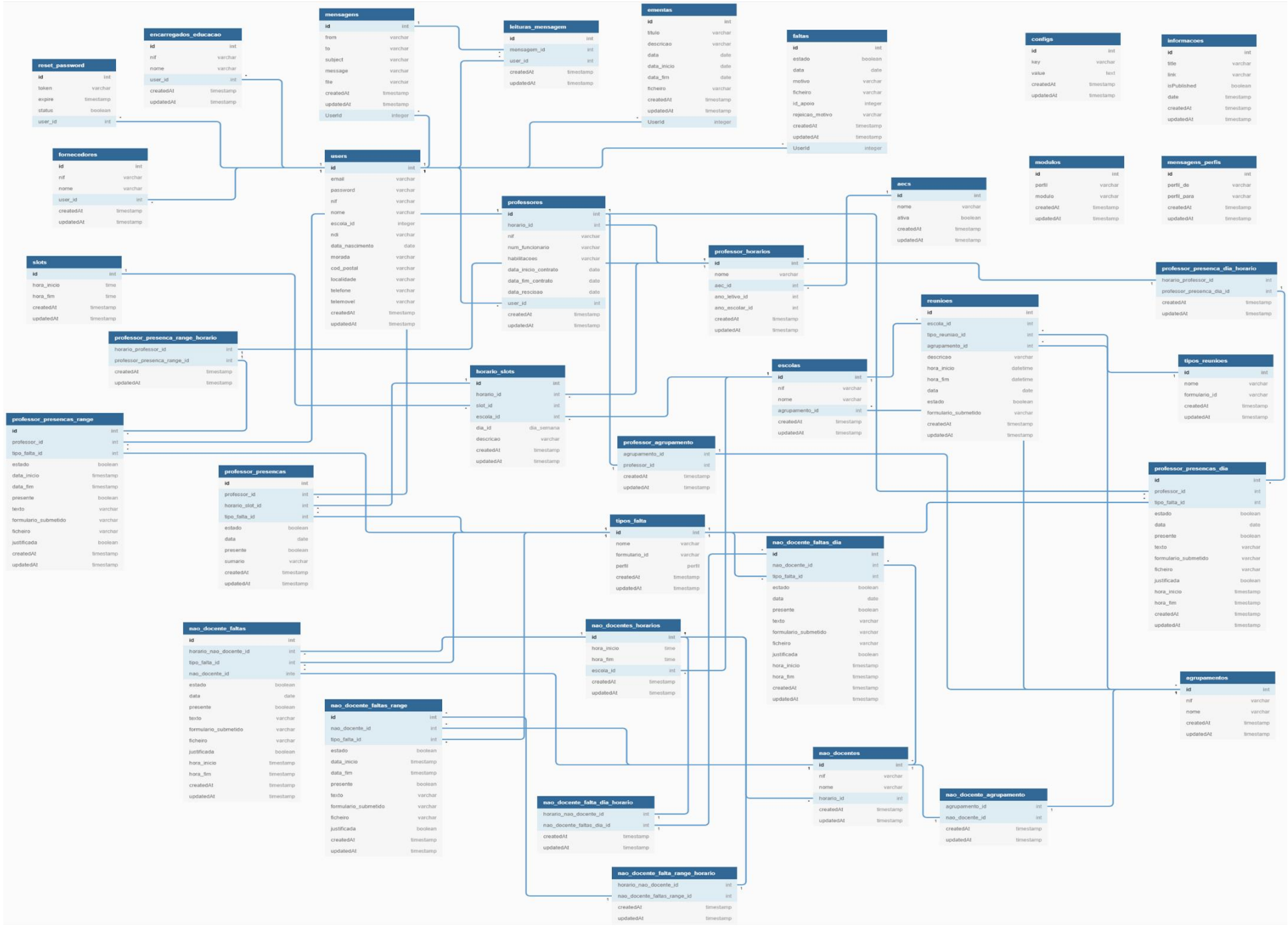

mensagens
id int
from varchar
to varchar
subject varchar
message varchar
file varchar
createdAt timestamp
updatedAt timestamp
UserId integer
leituras_mensagem
id int
mensagem_id int
user_id int
createdAt timestamp
updatedAt timestamp
ementas
id int
titulo varchar
descricao varchar
data date
data_inicio date
data_fim date
ficheiro varchar
createdAt timestamp
updatedAt timestamp
UserId integer
faltas
id int
estado boolean
data date
motivo varchar
ficheiro varchar
id_apoio integer
rejeicao_motivo varchar
createdAt timestamp
updatedAt timestamp
UserId integer
configs
id int
key varchar
value text
createdAt timestamp
updatedAt timestamp
informacoes
id int
title varchar
link varchar
isPublished boolean
date timestamp
createdAt timestamp
updatedAt timestamp
encarregados_educacao
id int
nif varchar
nome varchar
user_id int
createdAt timestamp
updatedAt timestamp
reset_password
id int
token varchar
expire timestamp
status boolean
user_id int
fornecedores
id int
nif varchar
nome varchar
user_id int
createdAt timestamp
updatedAt timestamp
users
id int
email varchar
password varchar
nif varchar
nome varchar
escola_id integer
ndi varchar
data_nascimento date
morada varchar
cod_postal varchar
localidade varchar
telefone varchar
telemovel varchar
createdAt timestamp
updatedAt timestamp
professores
id int
horario_id int
nif varchar
num_funcionario varchar
habilitacoes varchar
data_inicio_contrato date
data_fim_contrato date
data_rescisao date
user_id int
createdAt timestamp
updatedAt timestamp
aecs
id int
nome varchar
ativa boolean
createdAt timestamp
updatedAt timestamp
modulos
id int
perfil varchar
modulo varchar
createdAt timestamp
updatedAt timestamp
mensagens_perfis
id int
perfil_de varchar
perfil_para varchar
createdAt timestamp
updatedAt timestamp
slots
id int
hora_inicio time
hora_fim time
createdAt timestamp
updatedAt timestamp
professor_horarios
id int
nome varchar
aec_id int
ano_letivo_id int
ano_escolar_id int
createdAt timestamp
updatedAt timestamp
professor_presenca_dia_horario
horario_professor_id int
professor_presenca_dia_id int
createdAt timestamp
updatedAt timestamp
professor_presenca_range_horario
horario_professor_id int
professor_presenca_range_id int
createdAt timestamp
updatedAt timestamp
reunioes
id int
escola_id int
tipo_reuniao_id int
agrupamento_id int
descricao varchar
hora_inicio datetime
hora_fim datetime
data date
estado boolean
formulario_submetido varchar
createdAt timestamp
updatedAt timestamp
tipos_reunioes
id int
nome varchar
formulario_id varchar
createdAt timestamp
updatedAt timestamp
horario_slots
id int
horario_id int
slot_id int
escola_id int
dia_id dia_semana
descricao varchar
createdAt timestamp
updatedAt timestamp
escolas
id int
nif varchar
nome varchar
agrupamento_id int
createdAt timestamp
updatedAt timestamp
professor_presencas_range
id int
professor_id int
tipo_falta_id int
estado boolean
data_inicio timestamp
data_fim timestamp
presente boolean
texto varchar
formulario_submetido varchar
ficheiro varchar
justificada boolean
createdAt timestamp
updatedAt timestamp
professor_agrupamento
agrupamento_id int
professor_id int
createdAt timestamp
updatedAt timestamp
professor_presencas
id int
professor_id int
horario_slot_id int
tipo_falta_id int
estado boolean
data date
presente boolean
sumario varchar
createdAt timestamp
updatedAt timestamp
professor_presencas_dia
id int
professor_id int
tipo_falta_id int
estado boolean
data date
presente boolean
texto varchar
formulario_submetido varchar
ficheiro varchar
justificada boolean
hora_inicio timestamp
hora_fim timestamp
createdAt timestamp
updatedAt timestamp
tipos_falta
id int
nome varchar
formulario_id varchar
perfil perfil
createdAt timestamp
updatedAt timestamp
nao_docente_faltas_dia
id int
nao_docente_id int
tipo_falta_id int
estado boolean
data date
presente boolean
texto varchar
formulario_submetido varchar
ficheiro varchar
justificada boolean
hora_inicio timestamp
hora_fim timestamp
createdAt timestamp
updatedAt timestamp
nao_docente_faltas
id int
horario_nao_docente_id int
tipo_falta_id int
nao_docente_id inte
estado boolean
data date
presente boolean
texto varchar
formulario_submetido varchar
ficheiro varchar
justificada boolean
hora_inicio timestamp
hora_fim timestamp
createdAt timestamp
updatedAt timestamp
nao_docentes_horarios
id int
hora_inicio time
hora_fim time
escola_id int
createdAt timestamp
updatedAt timestamp
agrupamentos
id int
nif varchar
nome varchar
createdAt timestamp
updatedAt timestamp
nao_docente_faltas_range
id int
nao_docente_id int
tipo_falta_id int
estado boolean
data_inicio timestamp
data_fim timestamp
presente boolean
texto varchar
formulario_submetido varchar
ficheiro varchar
justificada boolean
createdAt timestamp
updatedAt timestamp
nao_docentes
id int
nif varchar
nome varchar
horario_id int
createdAt timestamp
updatedAt timestamp
nao_docente_agrupamento
agrupamento_id int
nao_docente_id int
createdAt timestamp
updatedAt timestamp
nao_docente_falta_dia_horario
horario_nao_docente_id int
nao_docente_faltas_dia_id int
createdAt timestamp
updatedAt timestamp
nao_docente_falta_range_horario
horario_nao_docente_id int
nao_docente_faltas_range_id int
createdAt timestamp
updatedAt timestamp

Figura 16 - Modelo relacional da base de dados

3.3.2 A Instalação nas Autarquias

Devido à necessidade de instalar o portal num servidor externo, disponibilizado pela autarquia para o efeito, e que não faz parte da responsabilidade da solução apresentada, foi necessário encontrar uma solução que tornasse simples o processo de instalação e manutenção, e de efetuar atualizações posteriores. Daí surgir a escolha da tecnologia *docker-compose*, para agrupar todos os componentes num projeto denominado de *deployer*. Tomou-se também a decisão de utilizar um registo de imagens, que guarda todas as versões e facilita todo o processo, visto que para instalar basta descarregar o projeto do *deployer* no servidor de destino e correr apenas alguns comandos para colocar a aplicação *online*.

Esta decisão acaba por fazer mais diferença num projeto destas dimensões, do que o impacto que teria num projeto mais pequeno. Devido ao tamanho das *builds* necessárias, à necessidade de ter um sistema sempre *online* como especificado nos requisitos não funcionais, e no geral para manter um certo nível de simplicidade, este projeto acaba por poupar muito trabalho e tempo, e facilita o processo de instalação em novos clientes.

3.4 Arquitetura

O planeamento da arquitetura de qualquer *software* é indispensável, e para tal é necessário conjugar toda a informação proveniente da análise feita anteriormente com os conceitos do modelo de negócio, e elaborar desta forma uma arquitetura lógica, não só como forma de referência para a fase da implementação, mas também como forma de facilitar a tomada de decisão antes do início do processo de desenvolvimento.

3.4.1 Diagrama de Componentes

Numa primeira fase de design arquitetural é apresentado o diagrama de componentes da plataforma, uma ferramenta útil que nos permite uma vista arquitetural de alto nível sobre o sistema, e nesse sentido ajudar a formalizar um *roadmap* para a implementação, assim como tomar decisões importantes durante toda a secção de *Design.* [19]

Podemos então identificar vários componentes, devidamente ilustrados na figura 17 sob a forma de um diagrama de componentes, que acaba por representar também o principal fluxo de informação da plataforma.

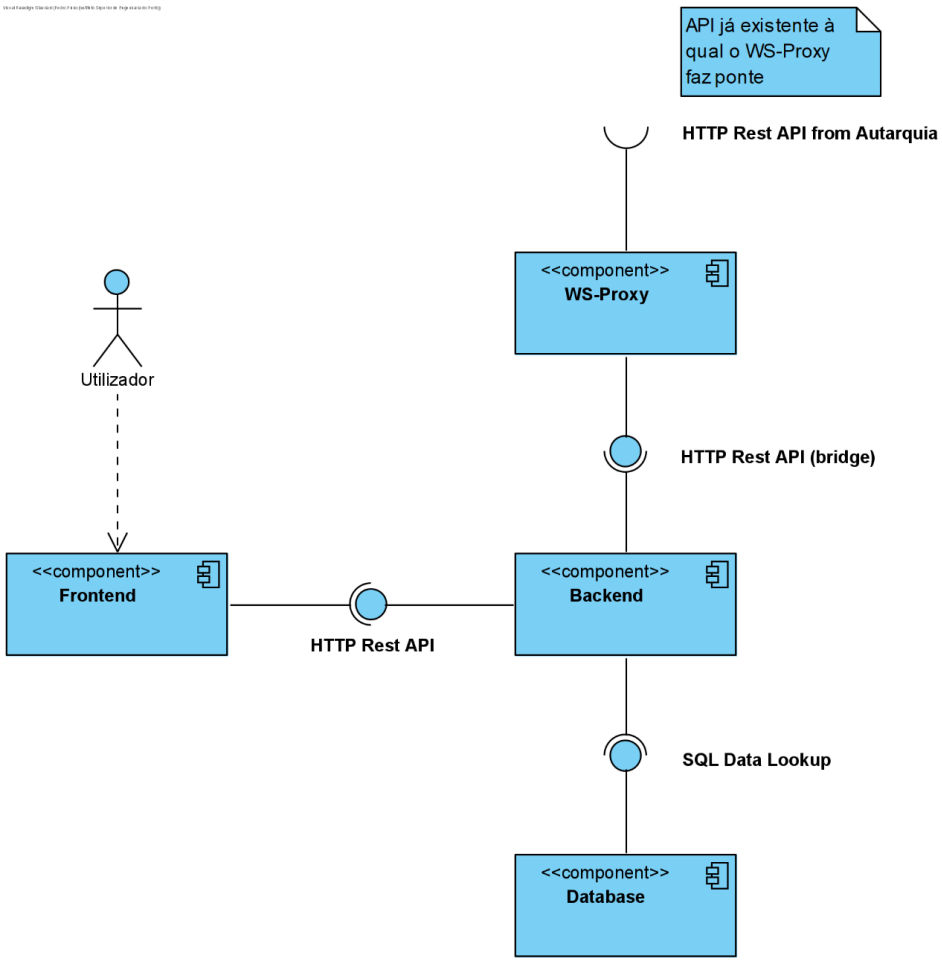


Figura 17 - Diagrama de Componentes do Portal da Educação

Através da análise do diagrama conseguimos perceber que existem 4 principais componentes do projeto.

- **Frontend:** Este componente corresponde a toda a interface gráfica da aplicação. O ponto de entrada para a plataforma está representado sob a forma do utilizador que interage apenas e unicamente com o componente de *frontend* da aplicação. O *frontend* é desenvolvido em *React.js*, sendo através dele que os utilizadores efetuam todo o tipo de atividades.
- **Backend:** Neste componente estão definidos todos os *endpoints* necessários para disponibilização ao *frontend*. A comunicação é feita através de pedidos HTTP. Toda a lógica da estrutura de dados e do modelo de negócio é aqui definida, assim como também é responsável pela persistência de dados. É desenvolvido em *Node.js.* É importante notar que este componente é o principal responsável por toda a estrutura do projeto, sendo por isso de alta importância.
- **Database:** Base de dados relacional (*PostgreSQL*), onde é guardada toda a informação do portal, que não seja diretamente proveniente da autarquia.

- **WS-Proxy:** Componente que desempenha a função de uma *Rest* API intermédia entre o *backend* do portal e a API da Medidata, com vista a consumir toda a informação necessária do software integrado de gestão de autarquias para o portal da educação.

3.4.2 Diagrama de Implantação

De forma a realizar uma melhor gestão de recursos / tecnologias e máquinas a serem utilizadas, bem como do alojamento dos diversos componentes da plataforma, desenhou-se um diagrama de implementação, que tem como objetivo representar a estrutura física do sistema a desenvolver [13].

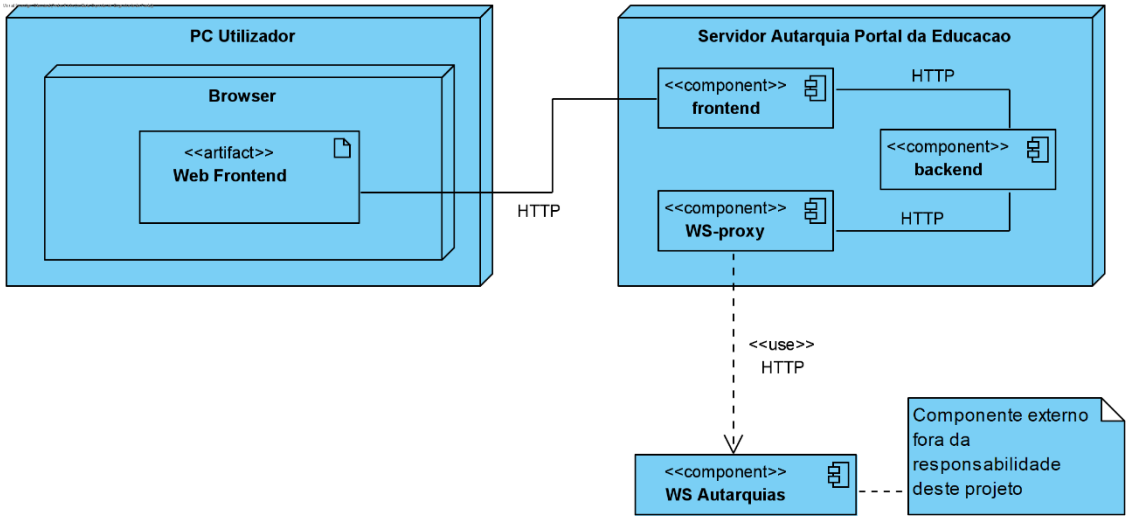


Figura 18 - Diagrama de Implantação

Na figura 18 podemos identificar 3 *nodes* distintos:

- **PC Utilizador:** Representa a máquina do utilizador onde será feito o acesso ao *website* do Portal da Educação através de um *browser*.
- **Servidor Autarquia Portal da Educação:** Servidor pertencente aos serviços informáticos de cada autarquia, onde estão alojados os componentes a desenvolver do sistema do Portal da Educação. Este servidor faz uso da tecnologia de "containerização" através de contentores de *Docker,* para uma fácil implantação e atualização dos componentes independentemente das características da máquina, sendo que cada componente é executado no seu próprio contentor, que por sua vez estão interligados através do *docker-compose* [14]. O componente WS-proxy comunica com o último *node* via HTTP.
- **Servidor Autarquia Sigma:** Esta máquina é onde está alojado o sistema de gestão de autarquias desenvolvido pela Medidata (*Sigma* ERP), de onde provêm vários tipos de dados necessários ao funcionamento do Portal da Educação. Esses dados existem num componente de persistência e são disponibilizados ao portal por uma *REST* API, através de consumo por parte do *WS-proxy.*

3.5 Alternativas ao Design

Nesta secção serão cobertas alternativas ao design, de forma a explicitar opções, decisões e ideias que possam funcionar com esta solução.

A opção mais óbvia que também foi pensada, era a de um sistema mais simples em termos de componentes, em que o *WS-Proxy* não existiria. Seria um sistema mais tradicional de interface gráfica, uma REST API e uma base de dados para a persistência. Põe-se claro, o problema dos dados provenientes da autarquia: poderia estar toda a lógica que faz e processa esses pedidos definida no *backend,* mas depois de análise mais cuidada identificam-se várias desvantagens ao utilizar este design alternativo:

- Desempenho do sistema diminui;
- Aumento da dificuldade de navegar no código do *backend* para o programador;
- Desenvolvimento de um componente de escala larga demais.

Não vale então a pena comprometer-nos com estas desvantagens todas, mesmo que no geral a estrutura do portal fique mais simples. Na figura seguinte podemos observar um diagrama de componentes desta alternativa ao design.

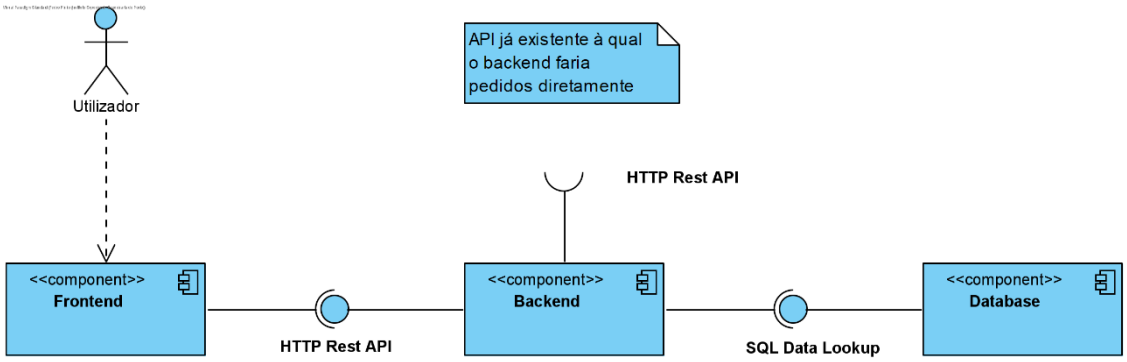


Figura 19 - Diagrama de componentes de um design alternativo

4 Implementação

Neste capítulo será descrito todo o processo de implementação do portal da educação, fortemente baseado em todo o planeamento e análise desenvolvidos anteriormente. Optou-se por um método de apresentação que segue um fio condutor, utilizando como exemplo algumas das principais funcionalidades do portal, e tudo o que elas englobam no seu fluxo, de início a fim, desde a interação inicial do utilizador com o *website*, passando pela interação entre componentes, até à persistência de dados e consumo destes.

Segue-se então uma breve exposição de toda a *codebase* e estrutura dos componentes necessários para a plataforma.

4.1 Estrutura dos Componentes

Como explicado anteriormente, a solução desenhada apresenta 3 componentes principais, o *Frontend*, o *Backend*, e o *WS-Proxy*, que serão devidamente identificados ao longo do capítulo sempre que a eles se fizer referência, quer por intermédio de figuras ou textualmente. O código está encapsulado dentro de um repositório *GIT* geral, utilizando a ferramenta *meta* (descrita no capítulo do estado da arte), contendo por sua vez os repositórios "filho" dos 3 componentes, excluindo ainda o repositório do instalador da plataforma (*deployer*), que será coberto no final do capítulo da implementação, na secção referente à instalação.

Nas figuras seguintes é possível visualizar este repositório *meta* e as estruturas dos repositórios subsequentes:

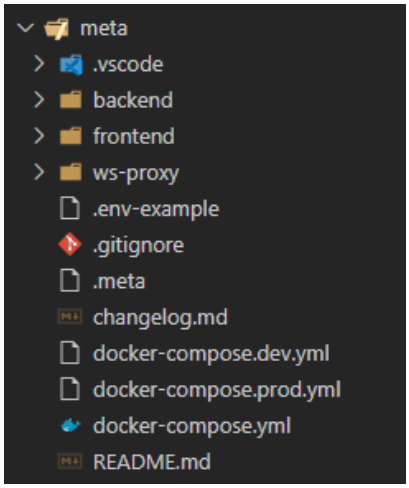


Figura 20 - Estrutura do repositório *meta*

Dentro deste repositório encontramos então as pastas correspondentes aos projetos do *frontend, backend e WS-Proxy,* os ficheiros de *docker-compose* que permitem correr facilmente a aplicação em qualquer PC, por exemplo para fins de desenvolvimento, também um *README.md* que dá algumas informações sobre o projeto, assim como o ficheiro *changelog.md*, que é o histórico de versões e respetivas alterações a cada atualização efetuada. De seguida estão apresentadas as estruturas de cada projeto "filho".

4.1.1 Frontend

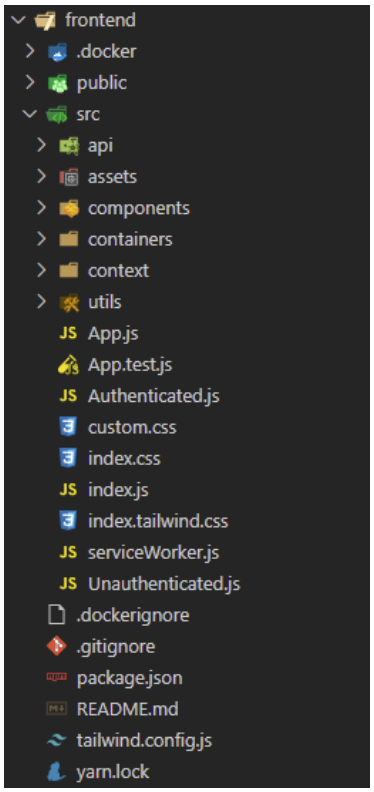


Figura 21 – Estrutura do projeto do frontend

No *frontend,* dentro da *package src* que contém o código encontramos várias packages, assim como as classes *Authenticated.js* e *Unauthenticated.js*. O ponto de entrada é no *index.js* que invoca o *App.js*, que é onde se encontra encapsulada a lógica da personalização do tema do portal, assim como é onde se efetua a verificação da autenticação do utilizador, utilizando a *package context,* que contém o código que efetua os pedidos de *login* ao *backend*. Consoante o utilizador esteja ou não autenticado, é mostrado o conteúdo do portal ou a *landing page*, respetivamente, que são representados pelas duas classes mencionadas acima.

É importante notar que este componente faz uso da funcionalidade do *React* denominada de *Code Splitting,* que permite separar a aplicação em vários componentes que podem ser carregados de forma dinâmica apenas quando são necessários, o que pode aumentar drasticamente o desempenho da aplicação e desta forma corresponder com vários requisitos explicitados na secção de requisitos não funcionais. [34]

```
//professor
const POConsultar = lazy(() => import('./containers/po/consultar/index'))
const POConsultarFormularios = lazy(() => import('./containers/po/consultar/formularios'))
const POConsultarSumarios = lazy(() => import('./containers/po/consultar/sumarios/index'))
const POCriarSumario = lazy(() => import('./containers/po/consultar/sumarios/create'))
const POConsultarReunioes = lazy(() => import('./containers/po/consultar/reunioes/index'))
const POCriarReuniao = lazy(() => import('./containers/po/consultar/reunioes/create'))
const POConsultarPresencas = lazy(() => import('./containers/po/consultar/presencas/index'))
const POEditarPresenca = lazy(() => import('./containers/po/consultar/presencas/edit'))
const POCriarPresenca = lazy(() => import('./containers/po/consultar/presencas/create'))
const POConsultarAvaliacoes = lazy(() => import('./containers/po/consultar/avaliacoes'))
const POConsultarHorarios = lazy(() => import('./containers/po/consultar/horarios'))
const POConsultarHorarioSlots = lazy(() => import('./containers/po/consultar/horarios/slots'))
const POConsultarFaltasAceites = lazy(() => import('./containers/po/consultar/faltas-aceites'))
const POConsultarReunioesAceites = lazy(() => import('./containers/po/consultar/reunioes-aceites'))
const POConsultarSumariosProfessores = lazy(() => import('./containers/po/consultar/sumarios-professores/index'))
const POConsultarPartilhaDocumentos = lazy(() => import('./containers/po/consultar/partilha-documentos/index'))
const POCriarPartilhaDocumento = lazy(() => import('./containers/po/consultar/partilha-documentos/create'))
const POEditarPartilhaDocumento = lazy(() => import('./containers/po/consultar/partilha-documentos/edit'))
const POPartilhaDocumento = lazy(() => import('./containers/po/consultar/partilha-documentos/show'))
```

Figura 22 – Declaração dos componentes para o perfil professor na classe *Authenticated.js*

Na figura 22 está representada a declaração dos caminhos para os vários componentes, e é possível constatar que é invocado o método *lazy(),* que é o carregamento dinâmico mencionado acima. Estes componentes são depois associados a um URL para que sejam carregados quando o utilizador aceder a este URL.

```
//Professores
const PRoutes = [
    {
        path: "/",
        exact: true,
        component: Home
    },
    {
        path: "/noticias/:id",
        component: Noticia
    },
    {
        path: "/noticias",
        component: Noticias
    },
    {
        path: "/perfil",
        component: Perfil
    },
    {
        path: "/password/editar",
        component: EditPassword
    },
    {
        path: "/consultar/horarios/:id/slots",
        component: POConsultarHorarioSlots
    },
```

Figura 23 - Associação de URLs aos diferentes componentes

As várias vistas do *frontend* estão depois divididas pela *package components*, que contém todos os componentes reutilizáveis como cartões, modais, calendários, etc. e pela package *containers*, que se encontra dividida por perfis (daí todo o foco na divisão de funcionalidades por perfis ao longo desta dissertação) de uma forma lógica, como é possível constatar nas figuras abaixo.

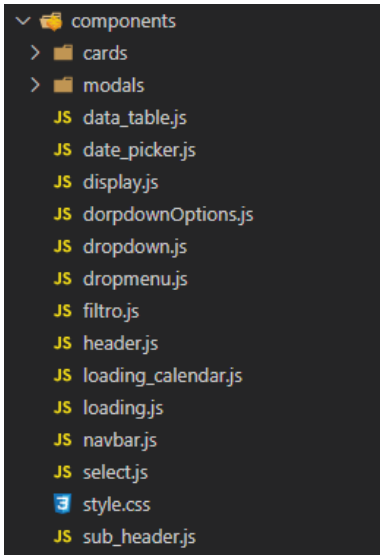


Figura 24 - A *package dos* componentes

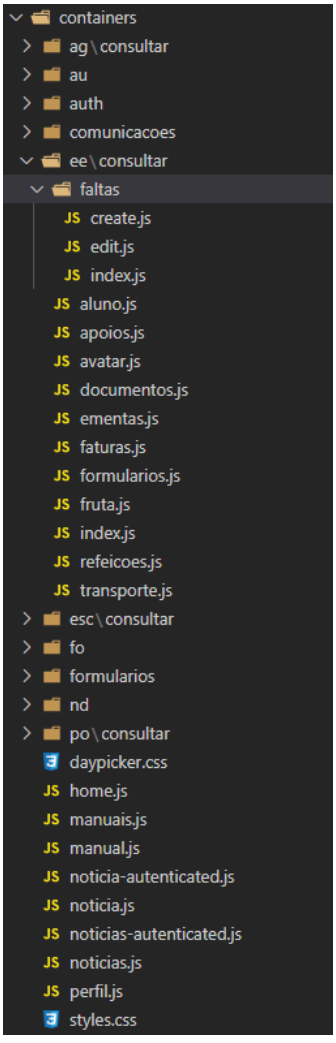


Figura 25 - A *package* dos *containers*

De acordo com a figura 25, cada divisão por perfil é ainda subdividida por módulo do portal. No caso da figura a sigla “ee” corresponde ao perfil do encarregado de educação, que contém várias classes relativas a cada módulo que está disponível para utilizadores com este perfil associado. Para as funcionalidades em que está presente, além da consulta de informação, as ações de criar dados ou de os editar, criou-se também mais uma subdivisão que contém cada vista específica para o efeito, como é o caso da pasta denominada faltas.

4.1.2 Backend

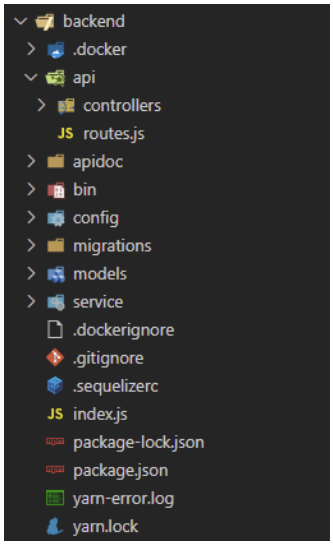


Figura 26 - Estrutura do projeto do *backend*

O próximo componente a ser descrito neste capítulo da implementação é o *backend*, uma REST API cuja função é disponibilizar todos os *endpoints* necessários para o funcionamento do *frontend*, segundo a estrutura REST. [18]

Na figura 26 é possível observar a árvore de ficheiros do *backend* de forma compactada, em que se percebe rapidamente como este componente está estruturado. Tendo em conta o que foi abordado na secção do design e do estado da arte em tecnologias, identificamos rapidamente a pasta *models* que contém a definição dos diferentes tipos de dados do domínio, e a partir dos quais o ORM mapeia essas informações para a base de dados, através das migrações geradas, que estão contidas dentro da pasta *migrations*. Nas figuras abaixo está exemplificada uma definição de um modelo utilizando a sintaxe específica do *Sequelize*, assim como uma das migrações geradas:

```
'use strict';
module.exports = (sequelize, DataTypes) => {
  const Agrupamento = sequelize.define('Agrupamento', {
    nif: {
      type: DataTypes.STRING,
      allowNull: false,
      unique: {
        msg: 'Já existe um Agrupamento com o mesmo NIF'
      },
      validate: {
        notNull: {
          msg: 'O campo NIF é obrigatório'
        }
      }
    },
    nome: DataTypes.STRING
  }, {});
  Agrupamento.associate = function(models) {
    models.Agrupamento.hasMany(models.Escola)
    models.Agrupamento.hasMany(models.Reuniao)
    models.Agrupamento.belongsTo(models.User)
  };
  return Agrupamento;
};
```

Figura 27 – Definição de modelo (agrupamento) no *backend*

```
'use strict';
module.exports = {
  up: (queryInterface, Sequelize) => {
    return queryInterface.createTable('Agrupamentos', {
      id: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      nif: {
        type: Sequelize.STRING
      },
      nome: {
        type: Sequelize.STRING
      },
      createdAt: {
        allowNull: false,
        type: Sequelize.DATE
      },
      updatedAt: {
        allowNull: false,
        type: Sequelize.DATE
      },
      UserId: {
        type: Sequelize.INTEGER,
        references: {
          model: 'Users',
          key: 'id'
        }
      },
    });
  },
  down: (queryInterface, Sequelize) => {
    return queryInterface.dropTable('Agrupamentos');
  }
};
```

Figura 28 - Migração gerada sobre o modelo apresentado na figura anterior

Dentro da pasta *api* o ficheiro *routes.js* define todos os *endpoints* disponíveis, assim como os perfis que podem aceder a cada *endpoint*. Todos os *endpoints* são reencaminhados para um ficheiro *controller* com o nome do modelo correspondente, que estão contidos na pasta *controllers*. Neste primeiro ficheiro o método que verifica se um pedido corresponde a um utilizador autorizado compara o perfil que efetuou o pedido com o perfil (ou perfis) especificados como autorizados em cada *endpoint*:

```
const gatekeeper = (perfis) => {
    return function (req, res, next) {
        if (req.authUser.perfis.some(obj => perfis.includes(obj.PERFIL))) return next()
        return res.status(401).send({ success: false, message: "Não autorizado" })
    }
}
```

Figura 29 - Método *gatekeeper* que verifica acesso aos *endpoints*

```
//agrupamentos
router.get('/agrupamentos', gatekeeper(["P5"]), agrupamento.get)
router.get('/agrupamentos/:id', gatekeeper(["P5"]), agrupamento.find)
router.post('/agrupamentos', gatekeeper(["P5"]), agrupamento.create)
router.patch('/agrupamentos/:id', gatekeeper(["P5"]), agrupamento.update)
router.delete('/agrupamentos/:id', gatekeeper(["P5"]), agrupamento.delete)
router.get('/agrupamentos/:id/escolas', gatekeeper(["P5"]), agrupamento.getEscolas)
router.get('/agrupamento/escolas', gatekeeper(["P4"]), agrupamento.getSelfEscolas)
router.get('/agrupamento/professores', gatekeeper(["P4"]), agrupamento.getSelfProfessores)
router.post('/agrupamento/professores/sumarios', gatekeeper(["P4"]), agrupamento.getSelfProfessoresSumarios)
router.get('/agrupamento/professores/horarios', gatekeeper(["P4"]), agrupamento.getSelfProfessoresHorarios)
router.get('/agrupamento/professores/horarios/slots', gatekeeper(["P4"]), agrupamento.getSelfProfessoresHorariosSlots)
router.get('/agrupamentos/:id/professores', gatekeeper(["P4", "P5"]), agrupamento.getProfessores)
```

Figura 30 - Exemplos de *endpoints* referentes aos agrupamentos

Na figura 29 verifica-se que são disponibilizados *endpoints* especificamente para as 4 funções básicas do CRUD [33], além de todos os outros que visam responder a todas as necessidades de consumo / criação de dados provenientes do *frontend*.

Por fim, cada *controller* redireciona o pedido (ou vários se for o caso) para as classes presentes na pasta *service*, onde está efetivamente presente toda a lógica do *backend*, sendo que em várias funcionalidades do portal são feitas a partir daqui chamadas ao *WS-Proxy,* com o objetivo de ir buscar dados provenientes da autarquia, o que será coberto mais para a frente.

```
getProfessores: async function (req, res, next) {

    let escola = await Escola.findByPk(req.params.id)
    if (!escola)
        return res.status(401).send({ status: false, error: 'Escola não existe' })

    Professor
        .findAll({
            include: {
                model: HorarioProfessor,
                required: true,
                include: {
                    model: HorarioSlot,
                    as: 'Slots',
                    required: true,
                    where: {
                        EscolaId: escola.id
                    }
                }
            }, order: [
                ['nome', 'ASC'],
            ],
        })
        .then((professores) => {
            res.status(200).send({ success: true, data: professores })
        })
        .catch((error) => {
            res.status(401).send({ status: false, error: error.message })
        })
},
```

Figura 31 - *Endpoint* para obter os professores de uma escola

4.1.3 WS-Proxy

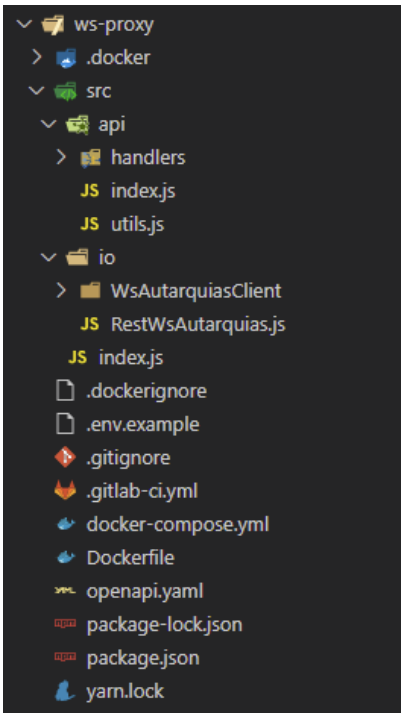


Figura 32 - Estrutura do projeto do *WS-Proxy*

O *WS-Proxy*, como mencionado na secção de design, é também uma API intermédia entre o *backend* e a API da autarquia, e utiliza a especificação *OpenAPI* pelo que tem uma estrutura fundamentalmente diferente do *backend*. [35]

Na *package* de io encontra-se toda a lógica que faz a ligação à API externa da autarquia, de onde são provenientes todos os dados que o *proxy* vai devolver ao backend. Inclui a lógica de autenticação e o *client* necessários para estabelecer uma ligação com sucesso.

Os *endpoints* desta API são então definidos pelo ficheiro de configuração *openapi.yaml*, que define o tipo de pedidos, os parâmetros, códigos de resposta, etc. como é possível observar na figura 33. Logicamente o *backend* efetua pedidos para os URLs aqui definidos.

```
"/agrupamento/{nif}/escolas":
  get:
    tags:
      - escola
    operationId: escola_getEscolasAgrupamento
    responses:
      200:
        $ref: "#/components/responses/ResponseObjectDataArraySchema"
    parameters:
      - $ref: "#/components/parameters/Nif"
```

Figura 33 - *Endpoint* para obter as escolas de um determinado agrupamento

O *operationId* presente nestes *endpoints* corresponde ao nome da lógica que vai efetuar o pedido, que se encontra na *package api -> handlers*. Aqui é definido qual o *endpoint* a ser consumido na API da autarquia, quais os parâmetros do pedido, e como deve ser devolvida a resposta. Para o mesmo pedido da figura 33, está apresentado na figura 34 o respetivo *handler*.

```
module.exports = async ({ wsAutarquias, restWsAutarquias }, ctx, req, res) => {
    const { nif } = ctx.request.params

    const result = await restWsAutarquias.apis.Escola.Escola_GetEscolasAgrupamento({
        contrib: nif
    })

    return {
        success: true,
        statusCode: 200,
        data: result.body['ESCOLASAGRUPAMENTO']
    }
}
```

Figura 34 - *Handler* do pedido que obtém as escolas de um determinado agrupamento

4.2 O Portal da Educação

Nesta secção será estudada a implementação do portal como um todo composto por "peças" que interagem entre si, do ponto de vista de um utilizador a realizar uma ação, desde a sua interação com a interface gráfica até ao *feedback* final, incluindo todo o processo interno.

4.2.1 Iniciar Sessão

Será estudado o caso de uso de início de sessão, por ser mais complexo do que o que inicialmente aparenta, e fazer uso de todos os componentes do projeto, como mencionado na secção da análise de requisitos funcionais, em que é possível consultar o respetivo diagrama de sequência.

Acedendo ao URL onde o portal estiver hospedado, o utilizador não autenticado depara-se com o ecrã inicial do portal, que contém as notícias, opções de registo e login, e ainda os manuais de instrução. Depois ao clicar no botão de login é redirecionado para um menu onde deverá inserir as suas credenciais. Neste passo é onde o componente do *frontend* fará pedidos ao *backend*, o que será aprofundado em termos de código.

Figura 35 - Página inicial do portal da educação

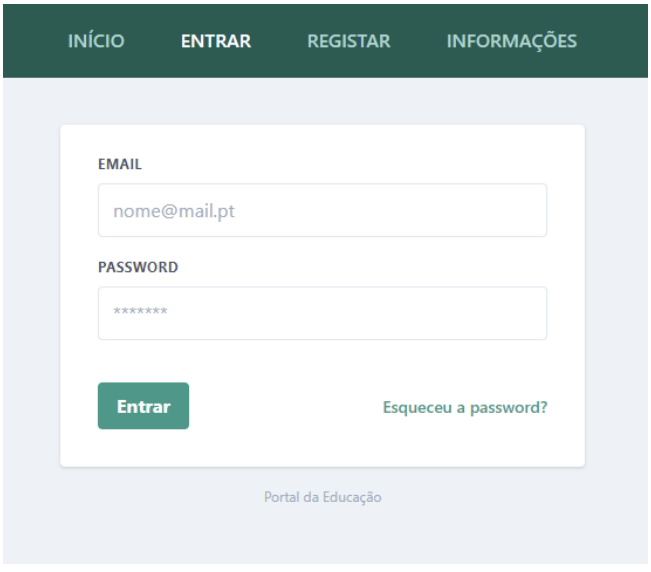


Figura 36 - Página de *login* do portal da educação

Após inserir as suas credenciais e clicar no botão de login, o *frontend* faz uma validação dos dados inseridos antes de enviar os dados num pedido para o *backend*, implementada de acordo com a figura 37.

```
<Formik
    initialValues={{ email: '', password: '' }}
    validate={values => {
        const errors = {}
        if (!values.email) {
            errors.email = 'Obrigatório';
        } else if (
            !/^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}$/i.test(values.email)
        ) {
            errors.email = 'Email inválido';
        }
        if (!values.password) {
            errors.password = 'Obrigatório';
        }
        return errors;
    }}
    onSubmit={(values, { setSubmitting }) => {
        login(values)
            .then((res) => {
                setSubmitting(false)
                setResponse(res)
            })
            .catch((err) => {
                setSubmitting(false)
                console.log(err)
            })
    }}
>
```

Figura 37 - Validação dos dados no *frontend*

```
function performLogin(params) {
  return fetcher('login', 'POST', params)
    .then(response => {
      if (!response) return Promise.reject(null)
      return response
    })
    .then(handleUserResponse)
    .catch(() => {
      return Promise.reject(null)
    })
}
```

Figura 38 - Pedido de login ao *backend* com os respetivos parâmetros

Após este pedido ser efetuado pelo *frontend,* é recebido pelo *backend* no respetivo *endpoint*, a partir do qual é feita uma chamada à base de dados que procura o utilizador pelo email inserido. Caso não exista é devolvido um erro para o *frontend* que avisa o utilizador de que não existe nenhum login com aqueles dados. No caso de existir é feita uma comparação da password por parte do *backend*, que verifica se a password inserida corresponde ao *hash* da base de dados, explicitado na figura 39.

```
User.prototype.comparePassword = function (passw, cb) {
    bcrypt.compare(passw, this.password, function (err, isMatch) {
        if (err) {
            return cb(err);
        }
        cb(null, isMatch);
    });
};
```

Figura 39 - Comparação da password

No caso de a comparação ser válida, é então feito um pedido de dados do utilizador do *backend* para o *WS-Proxy*, nomeadamente se é munícipe, assim como os perfis correspondentes a esse utilizador, dados estes já existentes provenientes da autarquia. Consoante o perfil do utilizador, o *backend* efetua uma validação para verificar se esse utilizador já existe na base de dados como um objeto do domínio (professor, escola, agrupamento, etc), ou se sofreu alguma alteração ao seu perfil. Caso não exista ou tenha sofrido alguma alteração, o *backend* cria / atualiza na base dados a informação correspondente.

```
getPerfis: function (req, res, next) {
    request(url + '/perfil/' + req.authUser.nif,
        function (error, response, body) {
            if (!error && response.statusCode == 200) {
                response = JSON.parse(body)
                req.perfis = response.data
                next()
            } else {
                //É municipe mas não tem perfis associados ao perfil
                next('err')
            }
        })
},
```

Figura 40 - Chamada do *backend* ao *WS-Proxy*

```
module.exports = async ({ wsAutarquias, restWsAutarquias }, ctx, req, res) => {
    const { nif } = ctx.request.params

    const result = await restWsAutarquias.apis.Perfil.Perfil_GetPerfis({
        contrib: nif
    })

    return {
        success: true,
        statusCode: 200,
        data: result.body['PERFIS']
    }
}
```

Figura 41 - *Handler* do *WS-Proxy* que pede dados à autarquia

Caso os passos descritos acima sejam bem-sucedidos, é gerado um *token* de autenticação pelo *backend,* que é devolvido para o *frontend* juntamente com um objeto do utilizador, após ser removida a palavra-passe. Isto significa que o processo de login teve sucesso e permite que o *frontend* "renderize" a página do menu principal e reencaminhe para lá o utilizador.

```
if (isMatch && !err) {
    proxyUserDetails(req, res, user)
        .then(user => {
            var token = jwt.sign(JSON.parse(JSON.stringify(user)), key, { expiresIn: 86400 * 30 });
            jwt.verify(token, key, function (err, data) {
                console.log(err, data);
            })

            //remover a password do objecto user para retornar
            delete user.password

            res.status(200).json({ success: true, user: user, token: 'JWT ' + token });
        }).catch(error => res.status(400).send({ success: false, message: 'Não foi possível autenticar. Entre em contacto com a autarquia' }))
} else {
    res.status(401).send({ success: false, message: 'Dados de acesso incorrectos. Por favor tente novamente' });
}
```

Figura 42 – Lógica do processo de login e geração do *token*

Figura 43 - Menu principal mostrado ao utilizador (Encarregado de Educação)

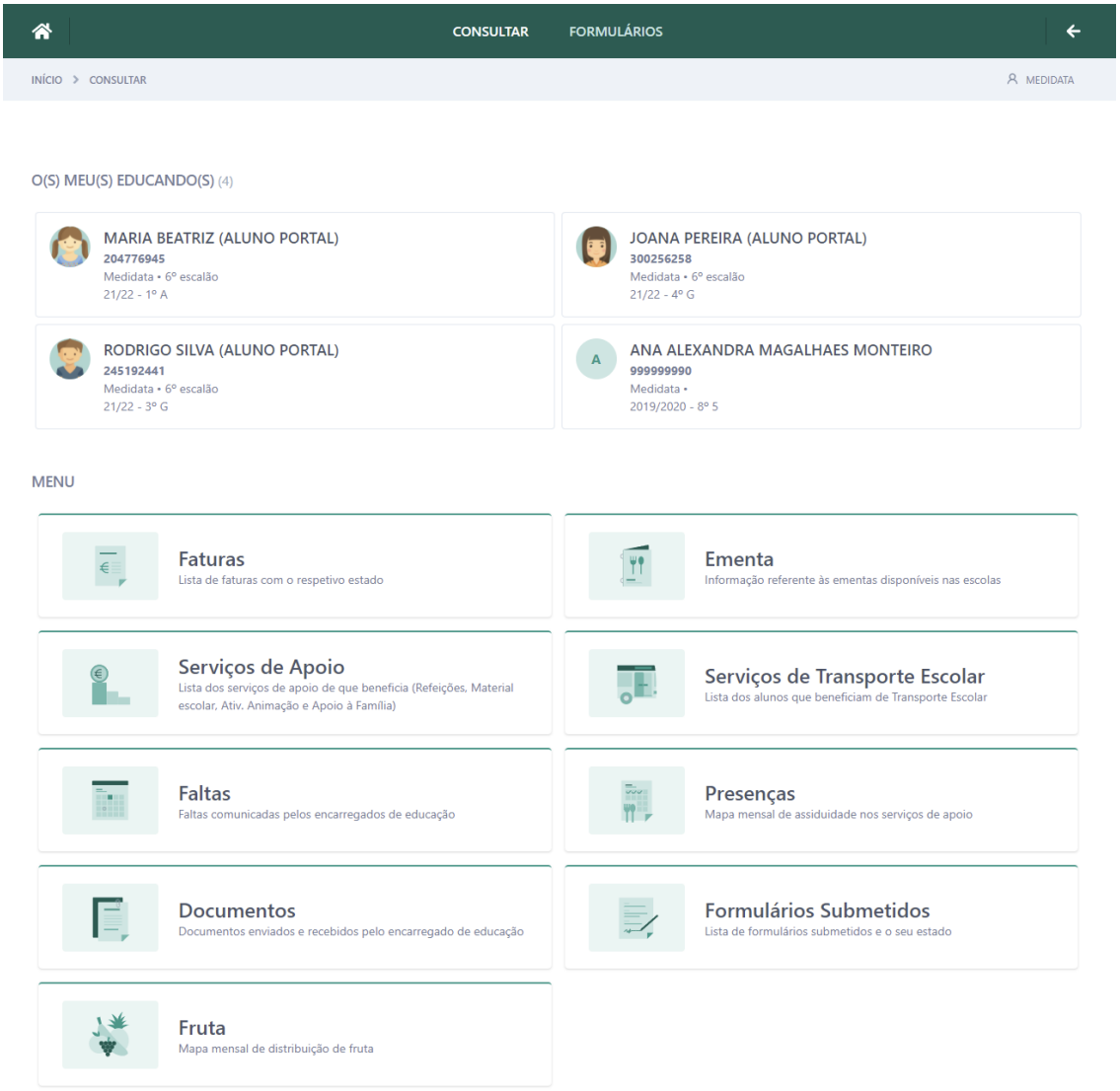


Figura 44 - Menu de consulta do Encarregado de Educação

4.2.2 Restante implementação

Em termos de implementação, excluindo os componentes do *frontend* e a lógica específica do *backend*, o processo de comunicação entre os diversos componentes segue a mesma lógica do que neste caso de uso descrito acima, ou seja, sempre que há uma funcionalidade que não necessite de dados da autarquia ou de confirmar utilizadores e etc, há apenas interação entre o *frontend* e o *backend*, sendo que o *backend* efetua todas as operações e consultas necessárias na base de dados. Sempre que um caso de uso envolve alunos, turmas, validações do utilizador e etc, então há mais uma camada sob a forma de chamadas do *backend* ao *WS-Proxy* para obter esses dados.

Para melhor elucidar o leitor acerca da restante implementação da solução, serão incluídos alguns *screenshots* de funcionalidades do portal.

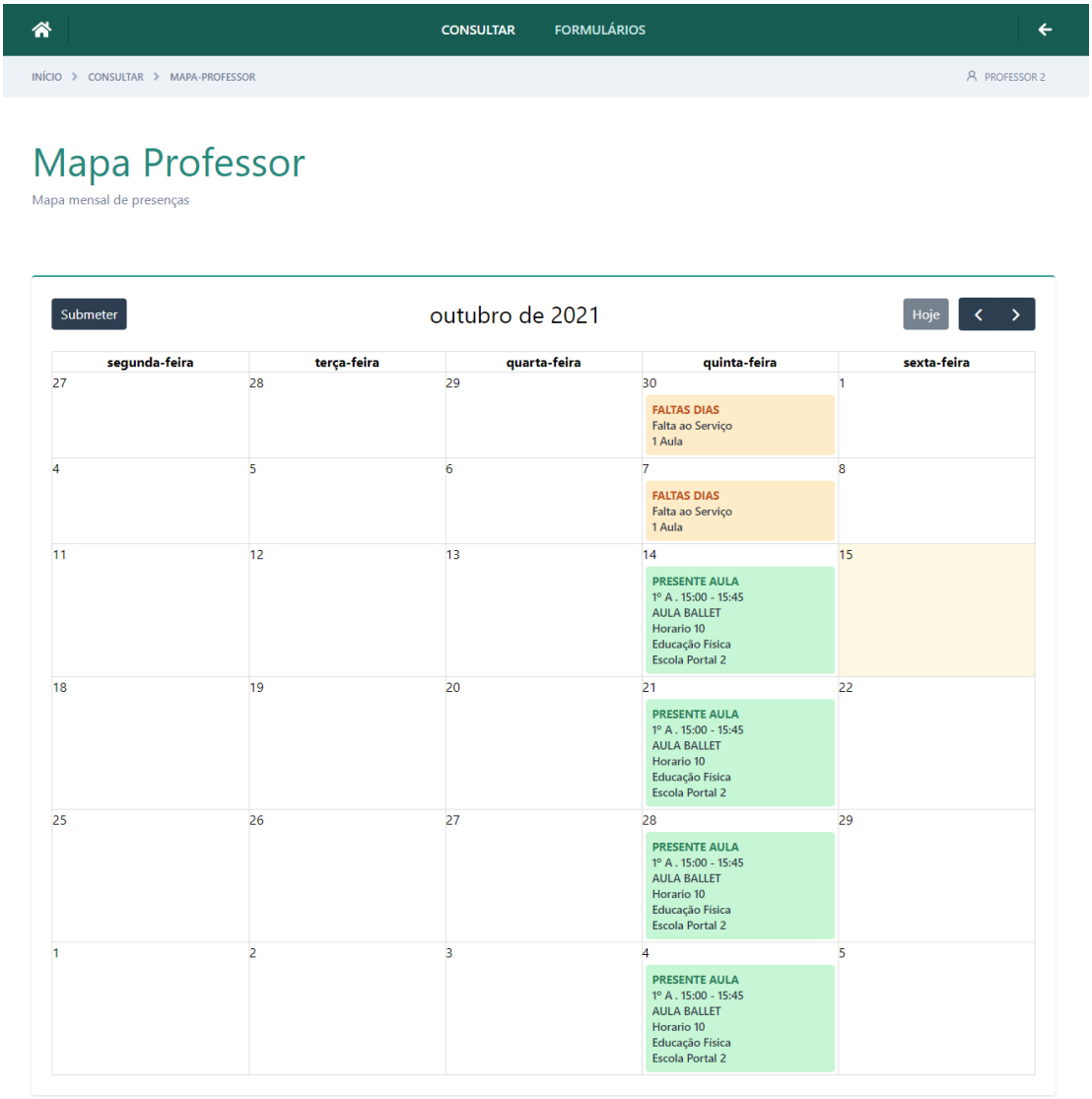


Figura 45 - Mapa mensal de presenças do professor

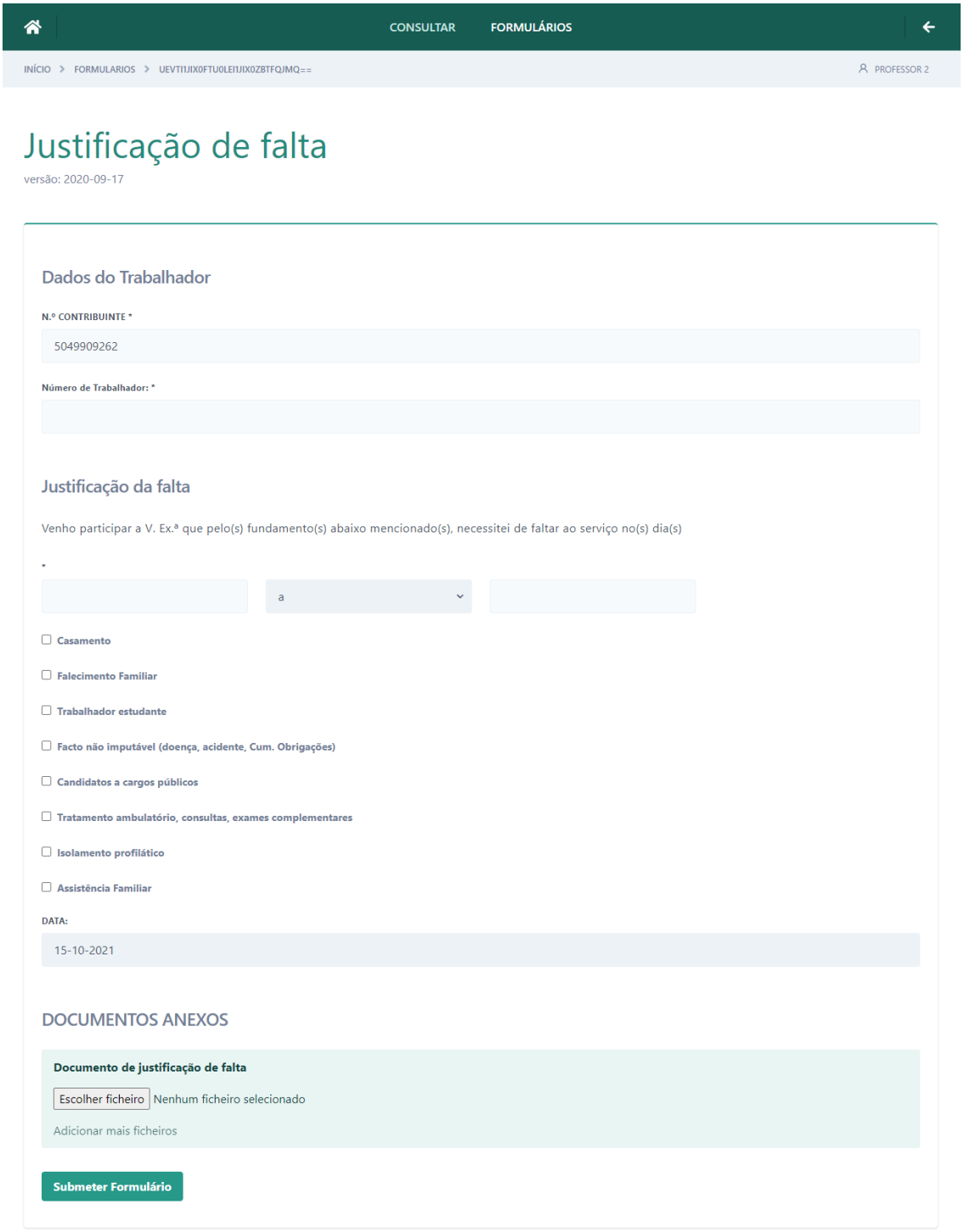


Figura 46 - Formulário de justificação de falta

Figura 47 - Horário de aulas de um professor

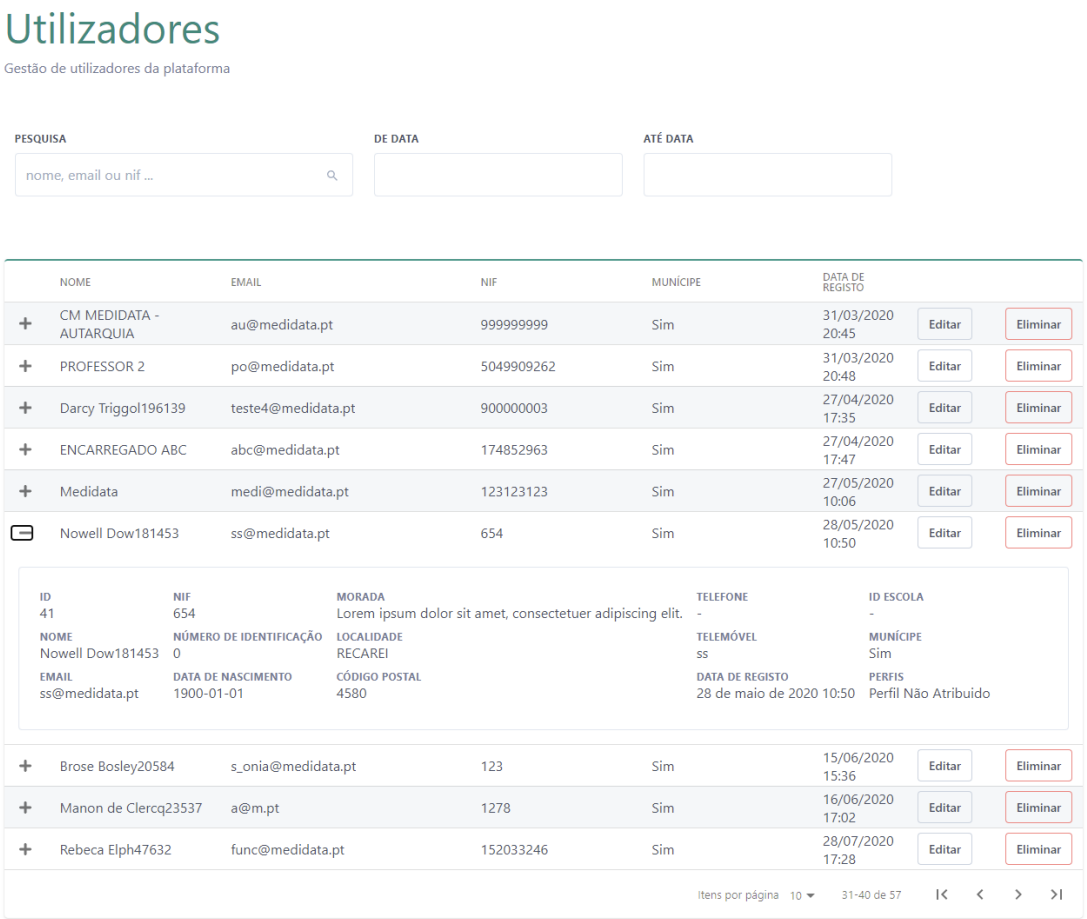


Figura 48 - Gestão de utilizadores da plataforma

4.3 Instalação

4.3.1 Nova Instalação

Como mencionado anteriormente, de acordo com os requisitos iniciais, cada autarquia tem a sua própria infraestrutura, logo o portal da educação terá de ser instalado num servidor externo. Aprofundando sobre a informação referente à instalação presente na parte da arquitetura, são apresentados de seguida os detalhes da implementação do *deployer*.

O *deployer* faz uso de *Docker* e do *docker-compose*, pelo que é o único *software* que já tem de estar instalado no servidor destino, seja *Windows* ou *Linux*. Preferencialmente recomenda-se aos clientes a utilização de uma distribuição *Linux* de servidor, devido a ser *open-source* e sem custos adicionais de licenciamento. Caso a autarquia decida utilizar uma edição normal do *Windows,* será necessário também ligar a funcionalidade de virtualização de hardware, denominada de *Hyper-V*. [22] Esta funcionalidade permite criar e correr máquinas virtuais, e é necessária para utilizar docker, segundo a documentação. [21]

Para novas versões do *Windows* é também possível recorrer ao *Windows Subsystem for Linux* (WSL), que, como o nome indica, é um subsistema de ambiente *Linux* a correr dentro do sistema operativo *Windows*, que dá ao programador várias ferramentas sem a sobrecarga encontrada nas tradicionais máquinas virtuais. [23]

Depois de instalados os requisitos básicos, podemos proceder à instalação do portal, começando por clonar o projeto do *deployer* e os respetivos ficheiros de configuração. A configuração específica para cada uma das diferentes autarquias é feita através de um ficheiro ".env", que contém todas as variáveis de ambiente necessárias para o funcionamento do portal. Assim ao colocar a plataforma em funcionamento com o *docker-compose,* serão lidas todas estas variáveis, que definem uma série de dados relevantes, como portas de funcionamento, URLs e até palavras-passe da base de dados, etc. Na figura seguinte podemos ver o ficheiro ".env-example", que é disponibilizado às autarquias aquando da instalação para que baste apenas preencher com a informação que lhes diz respeito:

```
COMPOSE_PROJECT_NAME=medidata-portal-educacao
COMPOSE_FILE=docker-compose.yml
DOMAIN=domain.com
EMAIL=email@domain.com
AUTARQUIA_CODIGO=
FRONTEND_PORT=3000
BACKEND_PORT=7000
WS_PROXY_PORT=9000
DATABASE_PORT=5432
REST_WS_AUTARQUIAS_SPEC_URL=
REST_WS_AUTARQUIAS_ISSUER=
REST_WS_AUTARQUIAS_AUDIENCE=
REST_WS_AUTARQUIAS_SECRET=
WS_AUTARQUIAS_BASE_URL=
WS_AUTARQUIAS_ID_AUTARQUIA=
WS_AUTARQUIAS_USERNAME=
WS_AUTARQUIAS_PASSWORD=
POSTGRES_USER=
POSTGRES_PASSWORD=
POSTGRES_DB=
SMTP_HOST=
SMTP_PORT=
SMTP_USERNAME=
SMTP_EMAIL=
SMTP_PASSWORD=
SMTP_BCC=
ROLES=P1,P2,P3,P4,P5,P6,P7,P8
NIF_AUTARQUIA=
```

Figura 49 - Ficheiro template ".env-example"

Após preencher corretamente este ficheiro de configuração podemos iniciar a aplicação escolhendo utilizar HTTPS ou apenas HTTP, recorrendo aos respetivos ficheiros, "*docker-compose.yml"* ou "*docker-compose-http-only.yml".* Estes ficheiros definem e configuram todas as dependências e serviços da aplicação, quais as portas a utilizar (carregadas do ficheiro .env), comandos a correr no *startup* e etc. Para inicializar o portal utilizamos os seguintes comandos:

HTTPS: docker-compose up -d

HTTP: docker-compose -f docker-compose-http-only.yml up -d

```yaml
backend:
    image: registry.redadviser.xyz/portaleducacao/backend:1.0.22
    restart: always
    volumes:
        - uploads-data:/app/uploads
    depends_on:
        - database
        - ws-proxy
    environment:
        - PORT=${BACKEND_PORT}
        - AUTARQUIA_CODIGO=${AUTARQUIA_CODIGO}
        (etc)
    ports:
        - "${BACKEND_PORT}:${BACKEND_PORT}"
    labels:
        - traefik.enable=true
        - "traefik.http.routers.backend.rule=PathPrefix(`/service/api`)"
```

Figura 50 - Exemplo de definição do *backend* no *docker-compose.yml*

Na figura 50 é possível observar um exemplo simplificado da definição do *backend*. Aqui encontramos as configurações de cada serviço, assim como podemos constatar que as variáveis de ambiente do ficheiro ".env" são de facto carregadas e utilizadas nos respetivos campos. É muito importante reparar na linha que especifica a imagem a ser utilizada encontra-se um URL, que aponta para um *Docker Registry*, que é uma aplicação alojada na infraestrutura da empresa que permite guardar e distribuir imagens de *docker* [24]. É utilizado neste contexto para um fácil controlo de versões, assim como para a instalação não necessitar de mais pré-requisitos. Este mecanismo de registo será coberto na secção seguinte que contempla a atualização da plataforma.

Dado ser uma nova instalação, é necessário encontrar o *id* do contentor do *backend*, entrar nesse mesmo contentor e correr as migrações do *schema* para que sejam criadas as tabelas da base de dados. Para correr as migrações utiliza-se o comando:

npx sequelize-cli db:migrate

4.3.2 Atualização

Com vista a atualizar o portal numa dada autarquia, primeiro é necessário dar *build* das novas versões do *frontend, backend,* etc. Depois deste passo faz-se o *upload* destas novas imagens para o nosso registo, atualizando também o número da versão nos ficheiros de *docker-compose* dentro do *deployer*. O registo está sempre online e é um meio intermédio que confere uma versatilidade enorme a todo este processo, daí ser referido na parte da nova instalação, porque supõe-se que as últimas versões das imagens já lá estão carregadas. Este método torna extremamente fácil a instalação e atualização de versões do software em qualquer autarquia,

até na eventualidade de ser preciso fazer o *downgrade* para uma versão anterior, visto que estão todas armazenadas no registo.

Do lado do servidor da autarquia basta entrar na pasta do *deployer*, carregar as alterações (usualmente apenas números de versões), e aplicar as alterações utilizando um dos dois comandos apresentados na parte da nova instalação. Caso exista também alguma alteração na base de dados é necessário correr novamente as migrações, caso contrário a atualização

A documentação completa do *deployer* está anexada para consulta no Anexo B.

5 Avaliação e Experimentação

Este capítulo apresenta o processo de avaliação e experimentação do projeto desenvolvido, os seus resultados, *feedback* apresentado pelo cliente, e situações de uso reais.

Antes de passar à avaliação da solução desenvolvida, identificam-se as metodologias e estratégias utilizadas para o fazer

5.1 Métricas de Avaliação

Para o estudo a realizar são tidas em conta várias métricas, quer do ponto de vista direto de utilização por parte de um utilizador, assim como aquelas que visam verificar se os requisitos não funcionais foram cumpridos. Neste sentido, a avaliação será desenvolvida com vista a responder a duas grandes questões:

- Qual o *feedback* dos clientes em relação ao portal da educação?
- Foram encontrados erros comprometedores na solução desenvolvida?

Claro que o sistema não será apenas testado sobre estas duas questões, sendo que todos os pontos positivos e negativos da solução são considerados parte integrante das métricas de avaliação.

5.2 Metodologia

A metodologia utilizada para a avaliação da solução recai muito sobre a experimentação, não só do cliente, mas de vários intervenientes. Durante todo o processo de desenvolvimento do portal, foi utilizado um servidor de testes, para o qual eram publicadas todas as atualizações, com o objetivo de simular o produto final num ambiente semelhante, e ao mesmo tempo permitir testagem e feedback que por sua vez impulsionou o desenvolvimento a cada iteração do *software*. Tinham acesso a este servidor de testes o pessoal da Medidata, assim como os

responsáveis pelos clientes já existentes, assim como eventuais novos clientes que demonstrassem interesse, para motivos de demonstração.

Após cada publicação neste servidor de testes era elaborado um conjunto de ideias e *feedback* geralmente positivo, assim como potenciais erros e diversas pequenas coisas a melhorar e iterar. É importante notar também que o portal se encontra ativo e em produção em várias autarquias, pelo que este servidor também foi crucial na identificação de erros antes de lançar novas versões oficialmente.

5.3 Resultados

5.3.1 Satisfação do cliente

Para a satisfação do cliente irá ser considerado o *feedback* do cliente piloto (câmara de Paredes), assim como o *feedback* de clientes subsequentes, assim como experiência de utilização dentro da Medidata e de todos os utilizadores em cada uma das autarquias. Para provar que o cliente ficou de facto satisfeito, é preciso ter em conta que os clientes, além de testarem no servidor de testes mencionado acima, também estavam ativamente a utilizar o portal num ambiente real e com dados reais, e tempo e dinheiro investidos na solução.

O *feedback* depois de realizados os testes era maioritariamente positivo, e por vezes incluindo um relatório de funcionalidades / erros a corrigir (consultar Anexo C). Depois de corrigidos e implementados todos os problemas e sugestões, uma nova versão era testada e depois publicada para produção. Ao longo das várias versões registou-se uma evolução positiva do *software*, que para além dos relatórios pós-testagem, também se manifestou sob a forma de satisfação das autarquias:

“O Município de Paredes dispõe agora de um Portal da Educação. Trata-se de uma ferramenta que vai estar ao dispor e aproximar a comunidade educativa. A partir deste Portal podemos encontrar notícias da área da educação, informações úteis aos encarregados de educação, para além de permitir a inscrição nos diversos serviços disponibilizados pelo Município aos alunos/crianças. É também um canal de comunicação e partilha de informação com as nossas escolas.” [36]

“Implementado com sucesso em Paredes, o portal da educação da Medidata- Sigma WsEduca, tem-se mostrado uma ferramenta imprescindível no contexto da educação e seus intervenientes. Numa altura em que as exigências a nível do contexto epidemiológico continuam a ser grandes, minimizar os contactos presenciais e privilegiar os meios e ferramentas digitais é cada vez mais a alternativa adequada e segura.” [37]

“A adesão à nova plataforma conta já com 2860 encarregados de educação que efetuaram o registo no Portal da Educação e realizaram online as inscrições para o novo ano escolar. Por conseguinte, no novo ano letivo, têm disponível todas as informações relativas aos seus

educandos podendo consultar notícias sobre a educação, faturas ou aceder a ementas escolares, e acesso a serviços online como a marcação de refeições, prolongamentos de horário ou pedidos de transporte escolar.” [38]

Temos então aqui dados que comprovam efetivamente a satisfação do cliente em relação à solução desenvolvida, e à forma como foi de encontro ao problema apresentado nesta dissertação.

5.3.2 Erros comprometedores da solução

Não tendo existido interesse por parte do cliente em testes unitários, nem testes de integração, apenas testes de usabilidade através do servidor de testes, não é possível apresentar valores ou dados estatísticos que comprovem a ausência de erros comprometedores da solução, embora seja possível argumentar que se o cliente se encontra satisfeito e a plataforma implementada a ser utilizada em ambiente de produção, é seguro assumir que não foram encontrados erros que comprometam a solução como um todo, excluindo claro pequenos erros que vão sendo corrigidos ao longo do tempo com este processo de desenvolvimento iterativo.

Em relação à gravidade destes erros menores, poderia ter sido pensada uma melhor forma de proceder à sua identificação, como por exemplo a implementação de diferentes tipos de testes automatizados que tornassem este processo mais eficiente, preciso, e acima de tudo mais fácil. Não tendo sido implementada tal solução, pode ser considerada uma limitação e a ser apontada como trabalho futuro no próximo capítulo de conclusão da dissertação.

5.4 Conclusão da Avaliação

No geral o sistema vai de encontro aos objetivos que foram delineados inicialmente, e cumpre com a principal métrica de satisfação dos clientes. Embora a métrica de erros comprometedores pudesse ter uma melhor avaliação em termos teóricos, é considerado aceitável que não existam erros que impeçam o correto funcionamento do portal da educação num ambiente real.

Desta forma é possível afirmar que os requisitos estipulados pelo cliente, assim como os objetivos que procuravam dar resposta ao problema, foram alcançados com sucesso

6 Conclusão

Neste capítulo final são apresentadas todas as conclusões relativas ao trabalho desenvolvido na presente dissertação, abordando os objetivos alcançados, todas as dificuldades / limitações que surgiram ao longo do projeto, assim como possíveis melhorias a implementar em trabalho futuro. No final também serão apresentadas algumas considerações pessoais para terminar este capítulo.

6.1 Objetivos Alcançados

A solução implementada ao longo deste projeto permitiu responder ao objetivo principal de criar uma plataforma de gestão integrada no âmbito da educação, com bom desempenho, facilidade de uso, e rica em funcionalidades.

A presente dissertação iniciou-se pela contextualização do problema, identificação de conceitos e de tecnologias relevantes, assim como um levantamento de soluções semelhantes que poderiam resolver partes do problema proposto. A análise de valor foi também importante para perceber como uma solução deste tipo se enquadraria em termos de valor para o cliente, e quais os pontos com mais potencial para se poder seguir o melhor caminho nesse sentido.

Depois de cuidadosamente analisado, juntamente com a identificação de todos os requisitos necessários, procedeu-se ao desenho e planeamento da solução, antes de finalmente descrever a implementação e demonstrar as capacidades do portal da educação.

Conclui-se então que a esmagadora maioria dos requisitos foi cumprido, e que o sistema desenvolvido responde de forma sólida ao problema proposto, podendo então declarar o projeto como um sucesso.

6.2 Limitações e Trabalho Futuro

As principais dificuldades sentidas no projeto foram o ritmo de desenvolvimento elevado, devido à necessidade de novas funcionalidades por parte dos clientes, assim como a larga escala do portal, que por vezes tornava inevitável alguma confusão durante a fase de implementação.

Embora os principais objetivos para o projeto tenham sido cumpridos, existe ainda um longo caminho a percorrer para melhorar a solução, mesmo estando esta em ambiente de produção e a ser utilizada pelos clientes! É constante a necessidade de novas funcionalidades, não só para os clientes existentes, mas como forma de apelar a novos clientes, assim como há sempre correções e melhorias a serem feitas, para não falar da otimização de desempenho e melhoria da experiência para o utilizador

6.3 Considerações Finais

Do ponto de vista do projeto e da solução em si, o trabalho desenvolvido foi extremamente positivo, de uma forma geral. Em relação à presente dissertação, fica a sensação de que o trabalho desenvolvido sobre o documento foi aquém das expetativas, pelo que não fez justiça à dimensão da solução desenvolvida. De qualquer forma, surge então uma excelente oportunidade de aprendizagem, aliada a toda a experiência adquirida ao longo deste projeto, que pode ser considerada única, e um bom ponto de partida para enfrentar todo o tipo de desafios no futuro.

Referências

[1] António Manuel Pires. *Uma reflexão sobre a análise do valor e o seu posicionamento no actual panorama da gestão da qualidade*. https://repositorio-aberto.up.pt/bitstream/10216/12568/2/Texto%20integral.pdf. Acedido em março 2021.

[2] Medidata. *Medidata*. https://www.medidata.pt/pages/228

[3] P. A. Koen, H. M. J. Bertels, and E. J. Kleinschmidt. *Managing the Front End of Innovation—Part II,* doi: 10.5437/08956308X5703199. Acedido em março 2021.

[4] P. Kotler e K. L. Keller. *Creating Customer Value, Satisfaction, and Loyalty*. Pearson Education, 2009.

[5] Pereira, Daniel. *O que é o Business Model Canvas*. https://analistamodelosdenegocios.com.br/o-que-e-o-business-model-canvas/. Acedido em março 2021

[6] Escola 360. *O Escola 360*. https://e360.edu.gov.pt/. Acedido em março 2021.

[7] Inovar+. *Inovar Alunos.* https://inovar-mais.com/inovar-alunos/#1475534964637-e542d62f-c9b7. Acedido em março 2021.

[8] Ana Paula Catalão. *O papel das plataformas informáticas na regulação da organização e gestão da escola. https://repositorio.ipl.pt/bitstream/10400.21/10956/5/O%20papel%20das%20PI%20na%20regula%C3%A7%C3%A3o%20da%20Organiza%C3%A7%C3%A3o%20e%20Gest%C3%A3o%20da%20Escola%20%28VF%29.pdf.* Acedido em março 2021.

[9] Open Source. *What is Docker?* https://opensource.com/resources/what-docker. Acedido em março 2021.

[10] Node.js. *About Node.js.* https://nodejs.org/en/about/. Acedido em março 2021.

[11] React.js. *React.js*. https://reactjs.org/. Acedido em março 2021.

[12] PostgreSQL. *PostgreSQL.* https://www.postgresql.org/. Acedido em março 2021.

[13] UML-Diagrams. *Deployment Diagrams Overview*. https://www.uml-diagrams.org/deployment-diagrams-overview.html. Acedido em setembro 2021.

[14] Docker. *Overview of Docker Compose*. https://docs.docker.com/compose/. Acedido em setembro 2021.

[15] Chetan Arora, Mehrdad Sabetzadeh, Lionel Briand. *Extracting Domain Models from Natural-Language Requirements: Approach and Industrial Evaluation.* https://orbilu.uni.lu/bitstream/10993/28067/1/paper.pdf. Acedido em setembro 2021.

[16] Object Management Group. *OMG® Unified Modeling Language® (OMG UML®).* https://www.omg.org/spec/UML/2.5.1/PDF. Acedido em setembro 2021.

[17] Ruth Malan, Hewlett-Packard Company, and Dana Bredemeyer, Bredemeyer Consulting. *Functional Requirements and Use Cases.* http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.436.4773&rep=rep1&type=pdf. Acedido em setembro 2021.

[18] Red Hat. *What is a REST API?*. https://www.redhat.com/en/topics/api/what-is-a-rest-api. Acedido em outubro 2021.

[19] Donald Bell. *The component diagram.* https://developer.ibm.com/articles/the-component-diagram/. Acedido em outubro 2021.

[20] Lawrence Chung, Julio Cesar Sampaio do Prado Leite. *On Non-Functional Requirements in Software Engineering.* http://ce.sharif.edu/courses/96-97/1/ce475-1/resources/root/NonFunctionalRequirements.pdf. Acedido em outubro 2021.

[21] Docker. *Install Docker Desktop on Windows.* https://docs.docker.com/desktop/windows/install/. Acedido em outubro 2021.
[22] Windows Documentation. *Hyper-V Technology Overview.* https://docs.microsoft.com/en-us/windows-server/virtualization/hyper-v/hyper-v-technology-overview. Acedido em outubro 2021.
[23] Windows Documentation. *What is the Windows Subsystem for Linux?* https://docs.microsoft.com/en-us/windows/wsl/about. Acedido em outubro 2021.
[24] Docker. *Docker Registry.* https://docs.docker.com/registry/. Acedido em outubro 2021.
[25] COEPD. *What is FURPS+?* https://businessanalysttraininghyderabad.wordpress.com/2014/08/05/what-is-furps/. Acedido em outubro 2021.
[26] Edubox. *Siga*. https://edubox.pt/siga/. Acedido em outubro 2021.
[27] Git. *Git*. https://git-scm.com/. Acedido em outubro 2021.
[28] Mateo del Norte. *Meta*. https://github.com/mateodelnorte/meta. Acedido em outubro 2021.
[29] OpenJS Foundation. *Express*. https://expressjs.com/. Acedido em outubro 2021.
[30] Sequelize. Sequelize Documentation. https://sequelize.org/master/. Acedido em outubro 2021.
[31] Aastha (Medium). *ACID vs BASE in Databases*. https://medium.com/geekculture/acid-vs-base-in-databases-1bcad774da26. Acedido em outubro 2021.
[32] Amit Kumar. #15. *ACID vs BASE: Database transactions*. https://systemdesignbasic.wordpress.com/2020/06/20/15-acid-vs-base-database-transactions/. Acedido em outubro 2021.
[33] Mozilla. *CRUD*. https://developer.mozilla.org/en-US/docs/Glossary/CRUD?retiredLocale=pt-PT. Acedido em outubro 2021.
[34] React.js. *Code-Splitting.* https://reactjs.org/docs/code-splitting.html. Acedido em outubro 2021.
[35] SmartBear Software. *OpenAPI Specification.* https://swagger.io/specification/. Acedido em outubro 2021.
[36] Câmara Municipal de Paredes. *Portal da Educação*. https://www.cm-paredes.pt/pages/390. Acedido em outubro 2021.
[37] Medidata. *O Sigma WsEduca já está implementado em Paredes e é um sucesso!* https://www.medidata.pt/pt/noticias/sigmawseducaparedes. Acedido em outubro 2021.
[38] Medidata. *2860 pais registados no Portal da Educação de Lousada.* https://www.medidata.pt/pt/noticias/sigmawseducalousada2. Acedido em outubro 2021.

Anexos

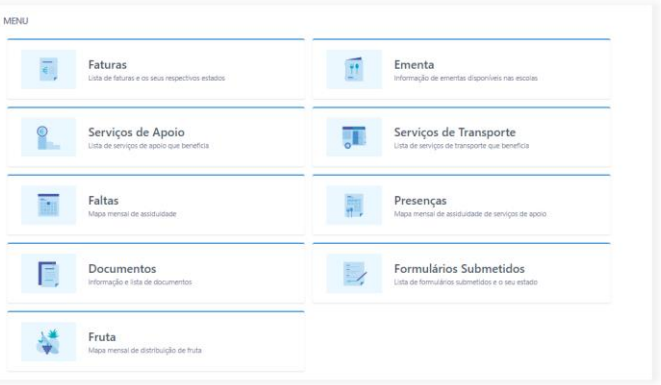
Anexo A

No presente anexo está apresentada uma versão inicial e simplificada do documento de especificação da plataforma.

medidata

Sigma WsEduca

O Portal de Educação disponibiliza o canal de comunicação entre os vários perfis: Autarquia, Encarregado de Educação, Escola, Entidade Fornecedora de Refeições, Agrupamento, Pessoal não docente e Professores. Este produto integra o observatório de educação da autarquia.



FUNCIONALIDADES DA APLICAÇÃO

Encarregado de Educação

- Consultar as faturas dos seus educandos
- Consultar toda a informação dos serviços de apoio dos seus educandos
- Consulta dos transportes escolares dos seus educandos
- Consulta da fruta fornecida na escola de cada educando
- Comunicar faltas dos seus educandos, com possibilidade de anexar documentos
- Consultar documentos: requerimentos/formulários e notificações
- Consultar presenças nas refeições e noutros serviços
- Consultar ementa
- Submeter formulários tais como: Inscrição de serviço, Cancelamento de serviço, Pedido de alteração do encarregado de educação, Pedido de alteração de escalão, Reclamação/Sugestão, ...

Escola

- Consultar presenças nas refeições e noutros serviços
- Visualizar as faltas comunicadas pelos encarregados de educação
- Marcar faltas de presença nas refeições e noutros serviços
- Consultar listas de alunos por escalão dos serviços de apoio
- Consultar lista dos alunos que usufruem do serviço de transporte
- Caraterização da Escola
- Inventariação do equipamento / material por espaço
- Consulta da fruta a distribuir aos alunos
- Submeter formulários tais como: Pedido de intervenção de obras na escola, Pedido de reparação de equipamento escolar, Exposição/Sugestão, ...

Figura 51 - Página 1 do documento de especificação

FUNCIONALIDADES DA APLICAÇÃO

Entidade Fornecedora de Refeições

- Consulta de número de refeições por escola

Agrupamento

- Mapa mensal de todos os professores do agrupamento
- Mapa mensal de todo o pessoal não docente do agrupamento

Pessoal não docente

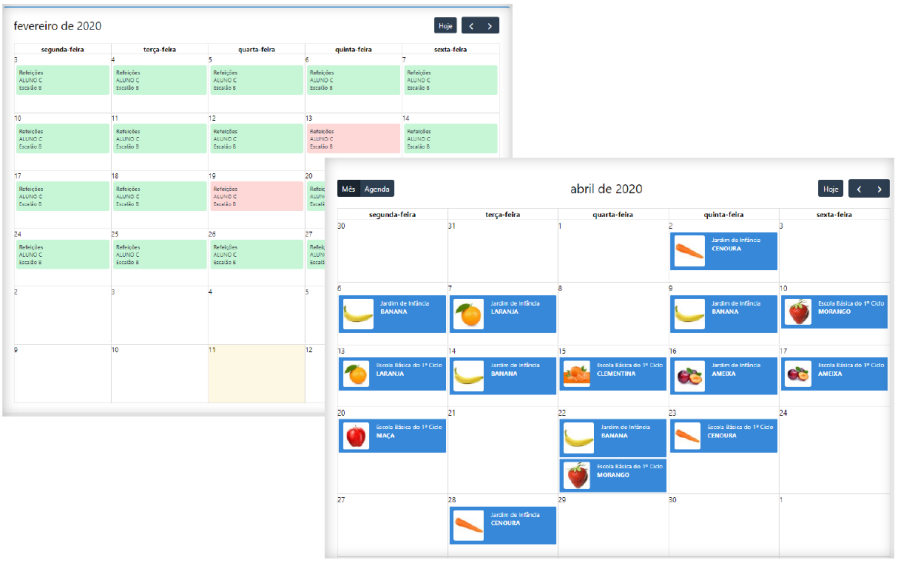
- Mapa mensal de pessoal não docente
- Consulta de formulários submetidos

Professores

- Consulta do mapa mensal do professor (sumários, reuniões, falta)
- Partilha de documentos entre professores da mesma AEC
- Partilha de documentos entre professores e autarquia
- Avaliação dos alunos nas Aec's

Autarquia

- Visualizar faltas comunicadas pelos encarregados de educação
- Mapa mensal de todos os professores
- Mapa mensal de pessoal não docente



INTEGRAÇÃO

- Autenticação integrada com o Sigma WsCidadão, cartão do cidadão, chave móvel.
- Integração com a aplicação de Ensino.
- Integração com a aplicação de Transportes Escolares.
- Integração com a aplicação de Sistema Integrado de Documentos e Atendimento Municipal.
- Integração com a aplicação de Imobilizado.

Figura 52 - Página 2 do documento de especificação

Anexo B

Portal da Educação Deployer

Instalador Portal Educação

Current Version: *1.0.12*

Requisitos:

- Docker (https://www.docker.com/)
- Docker Compose (https://docs.docker.com/compose/)

Instalação

Entrar no servidor via SSH

```
ssh <user>@<ip> -p <porta>
```

Ver versão do linux instalada (necessário para instalar os requisitos)

```
cat /etc/os-release
```

Instalar requisitos em Linux

Instalar Docker

https://docs.docker.com/install/linux/docker-ce/ubuntu/

Install Docker Engine - Community Seguir os passos 1,2,3,4 e 1,2 do guia

Para verificar se a instalação foi efetuada corretamente

```
docker -v
```

Instalar Docker Compose

https://docs.docker.com/compose/install/

Selecionar a versão (Linux) Seguir os passos 1 e 2 do guia

Para verificar se a instalação foi efetuada corretamente

```
docker-compose --version
```

Instalar dependencias em Windows

Instalar Docker Toolbox (Docker + Docker Compose) https://docs.docker.com/toolbox/toolbox_install_windows/

A máquina virtual deverá ter ativado o Hyper-V caso contrário não é possível instalar o docker https://kb.parallels.com/en/116239

Para verificar se a instalação foi efetuada corretamente

`docker -v` e `docker-compose --version`

Figura 53 - Página 1 da documentação do *deployer*

Instalar o projeto

Clonar o projeto e os ficheiros de configuração

(Necessário conta no gitlab e permissões no projeto)

```
git clone https://gitlab.com/portal-educacao/deployer.git
```

Copiar o `.env-example` para o `.env` a preencher

```
cp .env-example .env
```

Prencher corretamente o ficheiro `.env` com as configurações da instalação antes de iniciar a aplicação

```
nano .env
```

Depois de preenchido o env corretamente iniciar a aplicação

Para iniciar a aplicação com HTTPS

(porta 80 e 443 com ssl automático)

```
docker-compose up -d
```

Para iniciar a aplicação apenas com HTTP

(apenas porta 80 sem ssl)

```
docker-compose -f docker-compose-http-only.yml up -d
```

No primeiro deploy deverá entrar no container `backend` e migrar o schema da base de dados. Para tal deverão ser efetuados os seguintes passos:

- Encontrar o id do container backend `docker ps`
- Entrar no container `docker exec -it <backend_container_id> sh`
- Migrar o schema `npx sequelize-cli db:migrate`
- Sair `exit`

Atualizar o projeto

- Entrar na pasta "deployer" onde foi instalado o projeto
- Carregar as alterações no projeto (necessário credenciais) `git pull`
- Aplicar as alterações no servidor (2 opções)
  - porta 80 e 443 com ssl automático `docker-compose up -d`
  - apenas porta 80 sem ssl `docker-compose -f docker-compose-http-only.yml up -d`
- Verificar se é necessário migrar alguma alteração na base de dados (se sim realizar os passos a cima para migrar o schema)
- Verificar que as imagens estão online `docker ps`

Figura 54 - Página 2 da documentação do *deployer*

Commandos Úteis

Listar todos os ficheiros incluindo os ocultos

```
ls -la
```

Para ver o conteudo de um ficheiro de texto

```
cat <ficheiro>
```

Para atualizar os ficheiros de configuração

```
git pull
```

Para parar a aplicação

```
docker-compose down
```

Para visualizar os logs de um container

```
docker logs <container_id>
```

Desinstalar Projeto

Apagar dados e projecto do portal da educação

- Aceder com permissões `sudo`
- Entrar na pasta deployer `cd deployer`
- Ver os containers a correr `docker ps`
- Parar os containers `docker-compose down`
- Listar os repositórios `docker images`
- Apagar os repositórios (confirmar) `docker images prune -a`
- Listar os volumes `docker volume ls`
- Apagar os volumes executar (confirmar) `docker volume prune`
- Sair da pasta deployer `cd ..`
- Apagar a pasta deployer `rm -rf deployer`

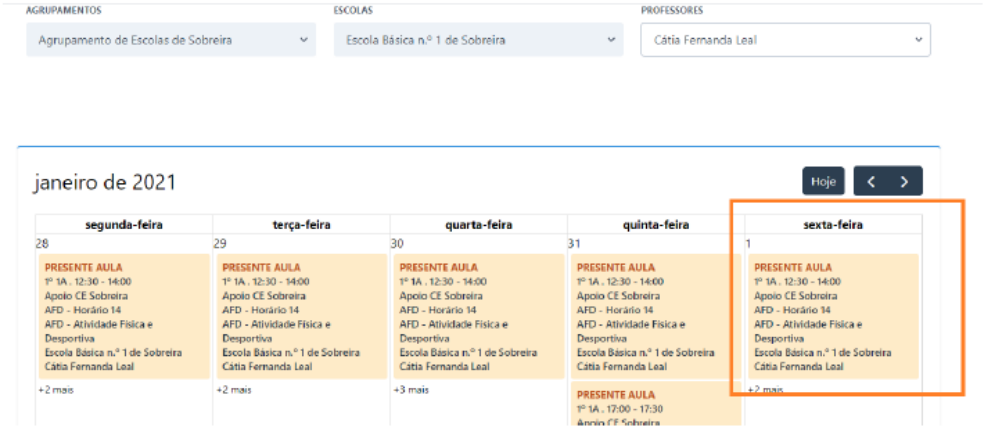
Figura 55 - Página 3 da documentação do *deployer*

Anexo C

No presente anexo encontra-se um exemplo de lista de *feedback* após testagem do sistema.

Lista de prioridades:

1. Exportar Dados alterar em todos os cartões, como desenvolvimento efetuado no cartão Faltas Professor
2. Os mapas não estão a contabilizar os feriados e deveria. Exemplo: Dia 1 de Janeiro é feriado



3. Perfil Agrupamento Retirar os cartões Faltas injustificadas - Pessoal Não Docente e Faltas injustificadas – Professor
4. Eliminar – no portal, em diversos locais, existe o botão eliminar. Devia de aparecer um alerta antes de eliminar para segurança do utilizador.
5. Corrigir o bug de Editar horários.
6. Perfil Agrupamento – Criar ementa para uma escola do agrupamento; para várias ou para todas as escolas do agrupamento
7. Perfil Autarquia
    - Criar Ementa para todas as escolas; Criar Ementa para as escolas dum agrupamento;Criar Ementa para várias escolas
    - Editar Ementa
8. As comunicações de perfil tem que fazer Filtros e não está a fazer.

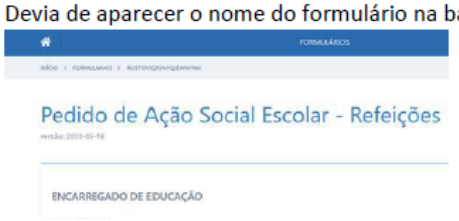
    Exemplo: EE -> Escola, só pode enviar para as escolas associadas aos seus Educandos, se existir mais que uma escola deve perguntar qual ou se é para ambas
    EE -> Agrupamento, só pode enviar para os agrupamentos associados aos Educandos
    Escola ->Agrupamento só pode enviar para o seu agrupamento
    Agrupamento->Escolas, só pode enviar para as suas escolas
    ...
    Outra pretençao perfil Agrupamento comunicar com o perfil Escola -> e ter a possibilidade de escolher a Escola
    Tem que existir filtros em todas as comunicações.
9. Nas comunicações existir a possibilidade de também enviar mail aos destinatários
10. Nas informações da Homepage ter a possibilidade de ordenar os conteúdo
11. Perfil autarquia

    Menu Plataforma

Figura 56 - Página 1 do documento de *feedback*

a. Agrupamentos – inserir/atualizar os agrupamentos automaticamente, sem ser necessário existir registo
b. Escolas – inserir/atualizar as escolas automaticamente, sem ser necessário existir registo
c. Professores– inserir/atualizar os professores automaticamente, sem ser necessário existir registo – Medidata a disponibilizar endpoint
d. Pessoal Não Docente– inserir/atualizar o pessoal não docente automaticamente, sem ser necessário existir registo – Medidata a disponibilizar endpoint

12. Os agrupamentos deviam ter os cartões: serviços de apoio, os serviços de transporte escolar e inventário das suas escolas
13. Eliminar Utilizadores, mesmo que estes sejam agrupamentos, escolas, professores ....
14. Devia de aparecer o nome do formulário na barra de cima e não o código



15. A câmara poder criar Notícias para um ou mais perfis, idêntico às notícias da Homepage. Exemplo: Notícia para os Encarregados de Educação, apenas os utilizadores logados com este perfil visualizam a notícia.
16. (Diogo)Criar utilizadores que possam aceder ao perfil autarquia e bloquear módulos por utilizador criados com este perfil
17. (Diogo)No portal, em utilizadores, devia ser melhorado tal como existe nos serviços online:

    - devia dar para pesquisar por data, ou seja, conseguir saber quantos registos foram feitos naquela data;

    - devia permitir ver se o utilizador já é municipe no portal;

    - devia sincronizar com a aplicação. Exemplo: se alterarmos na contrib o nome da pessoa ou outro dado devia alterar também no Portal;

    - caso o utilizador registado não tenha pedido de adesão, o perfil autarquia ter um botão para pedir o pedido de adesão

18. As cores do Portal devem ir de encontro às cores do site de cada câmara, temos que arranjar uma forma de resolver esta questão
19. (Implementado para os alunos, falta para as escolas, serviços de apoi e turmas)Nos formulários integrar com dados do Portal, exemplo: escolher o educando e preencher a informação associada a ele, ex: nif, nome, data de nascimento, sexo, ano escolar, escola, turma, ...
20. Cartões das Aec's dos Agrupamentos, passar os cartões para o perfil escola
21. Perfil Professor

    Faltam os cartões definidos no projeto:

    - Partilha de documentos (entre professores da área)

      (Título, Texto, Documento, Vídeo, Data e Hora criado)

Figura 57 - Página 2 do documento de *feedback*

- Partilha de documentos entre Câmara e os Professores (Título, Texto, Documento, Vídeo, Data e Hora criado)
- Avaliação dos Alunos das Aec's

22. Perfil Encarregado de Educação
    Faltam os cartões definidos no projeto:
    i. Consulta da avaliação dos Educandos nas Aec's inscritas
23. (Diogo para falarmos sobre este ponto)Aceder com qq conta de email e com uma password standard entre nós. Para fazer validações

Figura 58 - Página 3 do documento de *feedback*