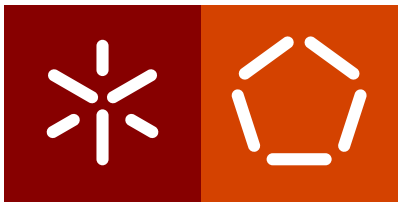
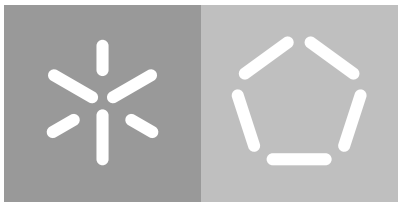


Universidade do Minho
Escola de Engenharia
Departamento de Informática

Cesário Miguel Pereira Perneta

Aplicação Móvel Inovadora de Apoio ao Planeamento e Administração de Medicação em Lares

Dezembro 2020



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Cesário Miguel Pereira Perneta

Aplicação Móvel Inovadora de Apoio ao Planeamento e Administração de Medicação em Lares

Dissertação de Mestrado
Mestrado Integrado em Engenharia Informática

Trabalho efetuado sob a orientação de
Professor Doutor José Manuel Ferreira Machado

Dezembro 2020

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

DECLARAÇÃO DE INTEGRIDADE

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

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

AGRADECIMENTOS

Mais uma etapa da maratona da vida concluída. Esta provavelmente a mais difícil até ao momento, mas ao mesmo tempo também a mais desafiadora e gratificante. Uma etapa cheia de altos e baixos, momentos felizes e outros não tanto, memórias e experiências, risos e devaneios, mas essencialmente de aprendizagem, tanto a nível académico como pessoal. Todos estes momentos tenho a agradecer às pessoas que me acompanham desde o início deste percurso e também às que conheci no desenrolar do mesmo.

Aos amigos do 'Proxys' e 'Discípulos e Irmãos' que fiz na universidade e que levo para a vida, que me proporcionaram momentos de diversão, descontração e felicidade, que em muito ajudou neste árduo percurso, mas também pelas conversas e palavras de confiança e motivação, que em muito ajudaram em alturas mais complicadas.

Aos amigos da "terrinha"do 'Easy Boy Easy Life' e 'Panchitos' que apesar da distância e de nem sempre ser possível estar com eles, passasse o tempo que passasse estavam sempre lá quando mais era preciso.

Ao Tiago Sá por todos os momentos que passamos, relembrando sempre os trabalhos de grupo que duravam até altas horas da madrugada.

Ao João e à Sandrina por toda a amizade, companheirismo e acompanhamento durante toda esta etapa.

Ao Carlos e à Andreia por todas as conversas e apoio que me proporcionaram.

À Ana, pela companhia, apoio incondicional, carinho e por me ter conseguido aturar nos momentos mais exaustivos.

Ao Professor Doutor José Manuel Ferreira Machado e à Doutora Marisa Araújo Esteves por me terem orientado e principalmente pela dedicação e confiança depositadas em mim.

À minha madrinha por todas as palavras, conselhos e incentivos.

Aos meus avós que sempre foram uma enorme fonte de inspiração e exemplo.

E por fim, aos meus pais. Os meus dois maiores pilares, os meus maiores exemplos de força e superação, sem eles nada disto seria possível. Ensinaram-me que quando se quer muito algo, todos os esforços, dedicação, lágrimas e suor valem a pena.

A todos, obrigado.

RESUMO

Nos dias de hoje, devido à acentuada evolução que a tecnologia tem vindo a sofrer, as aplicações informáticas tornaram-se indispensáveis no nosso dia-a-dia. Inúmeras áreas beneficiam, das mais variadas formas, das aplicações que têm ao seu dispor e a área da saúde não foge à regra.

No que toca aos lares de idosos, tendo em conta que estes estão cada vez mais lotados, os auxiliares de saúde não têm mãos a medir em relação à saúde dos utentes. No entanto, ainda são usados métodos arcaicos, que passam por manter toda a informação relativa aos tratamentos farmacológicos em formato físico, o que conduz a uma probabilidade de erro humano bastante elevada.

Para que seja possível tornar mais eficiente e reduzir a probabilidade de erro nos tratamentos farmacológicos dos utentes nos lares, é essencial que o processo de administração e planeamento destes tratamentos seja agilizado. Desta forma, pretende-se desenvolver uma aplicação que vise minimizar os problemas anteriormente mencionados, assegurando assim informação atualizada a todo o instante, históricos fidedignos, controlo sobre *stocks* de medicação, entre outros.

Palavras-chave: React Native, Node.js, MHealth

ABSTRACT

Nowadays, due to the accentuated evolution that technology has been undergoing, informatics applications have become a must in our daily lives. Countless areas benefit, in the most varied ways, from the applications they have at their service and the health area is no exception.

When it comes to nursing homes, taking into account that they are at their maximum capacity, the healthcare assistants have to much work on their hands. However, archaic methods are still used, which involve keeping all information related to pharmacological treatments in physical format, which leads to a very high probability of human error.

In order to make it more efficient and reduce the error of pharmacological treatments of patients in nursing homes, it is essential to streamline the procedure of planning and administration of these treatments. Because of this, we intend to develop a mobile application to minimize the problems previously addressed, assuring that the information is constantly updated, that the historic of the patient is reliable, that exists control over medication stock, among others.

Keywords: React Native, Node.js, MHealth

CONTEÚDO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE CÓDIGOS

1

INTRODUÇÃO

Neste capítulo será apresentada a contextualização e enquadramento (secção 1.1), a motivação para a presente dissertação (secção 1.2), os objetivos a alcançar (secção 1.3) e, por último, a estrutura do presente documento (secção 1.4).

1.1 CONTEXTUALIZAÇÃO E ENQUADRAMENTO

Em Portugal, a esperança de vida à nascença tem vindo a aumentar ao longo dos anos [2], bem como o índice de envelhecimento [3]. Estes dois aspetos, traduzem-se num aumento significativo do número de idosos em lares de terceira idade [4], o que resulta num aumento substancial da quantidade de medicação com que os profissionais desta mesma área têm de lidar. Aliado a este fator, a falta de profissionais a trabalhar em lares de idosos [5], faz com que se torne imprescindível que os métodos de planeamento e administração da medicação implementados sejam aperfeiçoados, deixando assim para trás os métodos arcaicos utilizados no planeamento e administração de medicação que vigoram atualmente.

Antes da administração de medicação, é necessário elaborar o planeamento dos tratamentos. Esta é uma tarefa que se relaciona com a assistência individualizada de cada utente, de maneira a satisfazer as exigências de saúde dos mesmos. Para que seja possível diminuir os equívocos relacionados com horário de administração, via de administração, dosagem e administração errada de medicação a utentes, é essencial que seja feita antecipada e periodicamente uma avaliação do planeamento, resultando numa diminuição dos erros passíveis de ocorrer aquando da administração dos medicamentos.

Assim, o planeamento facilita a organização entre os vários responsáveis de saúde, aumentando a sintonia entre os mesmos, conseguindo mais facilmente antecipar problemas e auxiliar nas decisões a tomar.

Em relação ao processo de administração de medicação, este torna-se mais simples e com menor propensão a erros, quando é previamente elaborado um correto planeamento.

Com a utilização de uma aplicação móvel, é expectável agilizar e diminuir a probabilidade de erro dos processos acima mencionados. Uma aplicação móvel oferece mais comodidade quando comparada com os métodos utilizados nos dias que correm (papel e caneta),

maior eficiência e precisão, ajuda nas tomada de decisão, alertas e construção automática de relatórios.

1.2 MOTIVAÇÃO

Com o incremento do número de utentes em lares e dos medicamentos envolvidos no tratamento dos mesmos, torna-se indispensável a utilização de um software que auxilie em todo este processo, desde o planeamento até à administração da medicação.

Um dos métodos mais usados hoje em dia consiste na utilização de registos em papel, com a informação clínica de cada utente, e quadros, onde são colocadas todas as notas relativas ao planeamento da medicação semanal. Tendo em conta a quantidade de informação que é necessário armazenar, existe a necessidade frequente de o papel ser trocado e de o quadro ser apagado, o que faz com que estes métodos sejam pouco eficientes. Havendo a necessidade de consultar registos anteriores para verificar o histórico, com este método, este procedimento pode tornar-se bastante incómodo, além de que existe também uma probabilidade acrescida de perda e degradação de registos anteriores, incompreensão da caligrafia dos diferentes responsáveis pelas tarefas relacionadas com a medicação e o desvanecimento da grafite utilizada para registar os medicamentos temporários.

Outro dos métodos utilizado é também software, no entanto, este é pouco utilizado, quer seja por ter baixa usabilidade ou até mesmo por se encontrar num computador fixo e haver a necessidade de se deslocar ao mesmo a cada registo.

Por estas razões surgiu a necessidade de criar uma aplicação móvel a ser utilizada em telemóvel ou *tablet*, de forma a colmatar as falhas acima referidas e possibilitar o registo individual da administração da medicação dos utentes. Desta forma, será possível proporcionar informações personalizadas, que irão auxiliar no processo de tomada de decisão no momento da administração ou alertar para possíveis problemas relacionados com o planeamento e administração da medicação.

1.3 OBJETIVOS

Assim sendo, é expectável que a tecnologia torne estes métodos mais ágeis e eficazes, tornando-se imprescindível a utilização de software que auxilie todo este processo.

Através da aplicação móvel será possível fazer o planeamento da medicação semanal de cada utente, realizar a gestão dos utentes e da medicação, registar informações relativas à administração da medicação, analisar dados providenciados através das ferramentas de *Business Inteligence*, análise e organização de dados que irão fornecer indicadores aos utilizadores da mesma, de forma a retirarem conclusões que em muito podem ajudar em futuras tomadas de decisão.

Algumas questões de investigação surgiram no contexto do presente tema, de forma a corroborar o âmbito desta dissertação:

- De que forma pode esta aplicação auxiliar no planeamento da medicação semanal num lar de idosos?
  - Recorrendo à planificação elaborada automaticamente pela aplicação, que tem por base a medicação de longo-prazo e temporária de cada utente;
  - Através dos dados inseridos acerca de cada medicamento, são lançados alertas quando o mesmo estiver a acabar, evitando assim ruptura de *stock*.
- É possível o processo de administração da medicação ser agilizado?
  - Através da informação relativa à medicação do utente, que se encontra na aplicação, agilizando assim o processo de consulta da mesma de modo a esclarecer qualquer dúvida existente.
- De que forma esta aplicação pode auxiliar os profissionais de saúde no momento de acompanhar os utentes ao médico?
  - Acesso imediato e simples aos dados do utente;
  - Consulta da ficha atual e do histórico de medicação do utente;
  - Disponibilização das informações relativas à instituição.

1.4 ESTRUTURA DO DOCUMENTO

- Capítulo 1 - Introdução
  - Contém a contextualização e enquadramento, a motivação e os objetivos da presente dissertação.
- Capítulo 2 - Estado da Arte
  - Composto por duas secções: as aplicações móveis na área da saúde e a aplicação de *Business Inteligence* na área da saúde.
- Capítulo 3 - Metodologias de Investigação e Tecnologias
  - São explicadas as metodologias e tecnologias utilizadas.
- Capítulo 4 - Análise e Abordagem Proposta
  - São apresentados e explicados o levantamento de requisitos, modelo de domínio, diagramas de *use case* e por fim o modelo lógico da base de dados.
- Capítulo 5 - Desenvolvimento da Aplicação Móvel

  - Composto por três secções:
    * Base de dados;
    * Backend; Frontend.

- Capítulo 6 - Interfaces da Aplicação Móvel
  - São apresentados os principais interfaces desenvolvidos bem como uma breve explicação acerca dos mesmos.

- Capítulo 7 - Prova de Conceito
  - É elaborada a análise SWOT relativa à prova de conceito.

- Capítulo 8 - Conclusão e Trabalho Futuro
  - É apresentada a conclusão de todo o trabalho elaborado, referindo as principais contribuições e alguns temas para trabalho futuro.

2

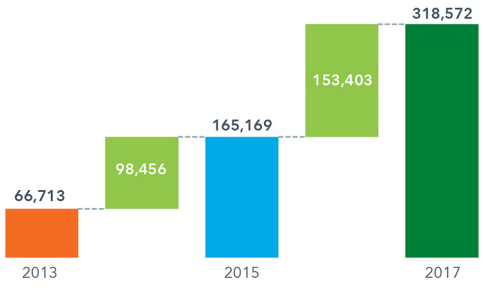
ESTADO DA ARTE

No capítulo que se segue, é apresentado o estado da arte relativo ao tema da dissertação, nomeadamente os conceitos teóricos e científicos relacionados com o desenvolvimento da mesma.

Dividido em 2 secções, este capítulo aborda numa primeira instância a relação das aplicações mobile na área da saúde. De seguida é explicado o conceito de *Business Intelligence* de uma forma geral e de seguida mais aplicado às áreas da saúde.

2.1 APLICAÇÕES MÓVEIS NA ÁREA DA SAÚDE

O uso de aplicações móveis na área da saúde, também denominado por *mHealth* (*mobile health*), tem vindo a tornar-se cada vez mais popular. Estas aplicações são utilizadas para armazenar dados, fornecer informações personalizadas, providenciar alertas, melhorar e tornar mais eficaz o acesso aos dados, reduzir custos e aumentar a qualidade dos serviços.



Sources: 42 Matters, Jul 2017; Mevvy, Jun 2015; IQVIA AppScript Database, Jul 2015; IQVIA Institute, Jul 2017
Report: The Growing Value of Digital Health. IQVIA Institute for Human Data Science, Nov 2017

Figura 1: Número de aplicações de saúde existentes - 2013, 2015 e 2017.

Este aumento do número de aplicações na área da saúde demonstra o quão úteis estas podem ser nestas mesmas áreas.

As aplicações *mHealth* facilitam o acesso a recursos informativos, o auxilio a diagnóstico de doenças e as tomadas de decisões clínicas e monitorização de utentes, razões pelas quais se tornam muito úteis. Estas facilidades originadas pelo uso deste tipo de aplicações, resultam em benefícios como a comodidade, uma maior precisão e eficiência e também uma maior produtividade. Estes fatores influenciam positivamente as tomadas de decisão clínica, fazendo com que estas possam ser mais acertadas [6].

Uma das grandes adversidades que faz com que nos dias de hoje não se explore mais as aplicações *mHealth*, consiste no facto de que apenas uma reduzida percentagem dos utilizadores de *smartphones* utiliza aplicações de saúde.[7] Uma das medidas que pode aumentar esta percentagem, reside na implementação de aplicações mais *user-friendly*, tanto a nível funcional, como a nível visual. Uma aplicação que seja elaborada tendo em conta este requisito, proporciona uma experiência de utilização mais satisfatória e agradável, facilitando o uso da mesma e permitindo também que estas tenham um maior alcance de faixas etárias, provocando assim um aumento da percentagem de utilizadores.

As aplicações na área da saúde estão a tornar-se cada vez mais populares e um forte indicador é a quantidade de aplicações disponíveis em diferentes línguas, o que leva a crer que é cada vez mais uma área com enorme evolução no mundo das aplicações.

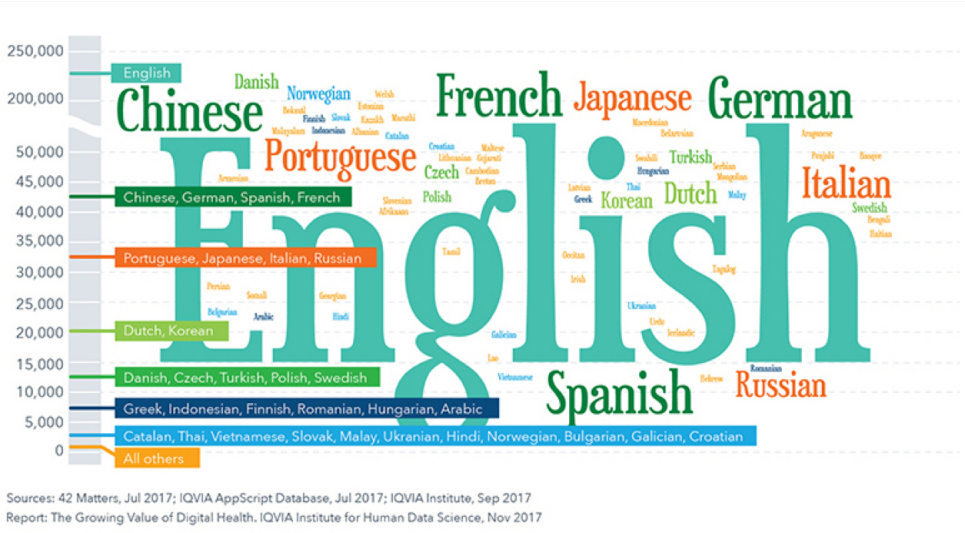


Figura 2: Quantidade de aplicações disponíveis em diferentes línguas.

No que diz respeito a aplicações móveis para a administração e planeamento da medicação em lares, não existe documentação que prove a sua existência, no entanto, existem software para computadores. Estes software são de pouca usabilidade, visto que, em termos de deslocação, estão limitados. Existe assim a necessidade de desenvolver um software que

facilite todo o processo relacionado com a medicação em lares, mas que tenha elevada usabilidade, ou seja, uma aplicação móvel.

2.2 BUSINESS INTELIGENCE

As ferramentas de *Business Inteligence*, auxiliam no processo de recolha e processamento de grandes quantidades de dados, de forma a que estes se tornem significativos para consultar e criar relatórios de dados. Desta forma, é esperado que aplicações ofereçam suporte à tomada de decisão, potencializando e otimizando processos, através dos dados extraídos com suporte a ferramentas de *Business Inteligence*. Existem diferentes ferramentas destinadas para este fim, das quais se destacam duas, o Tableau e o Power BI [8].

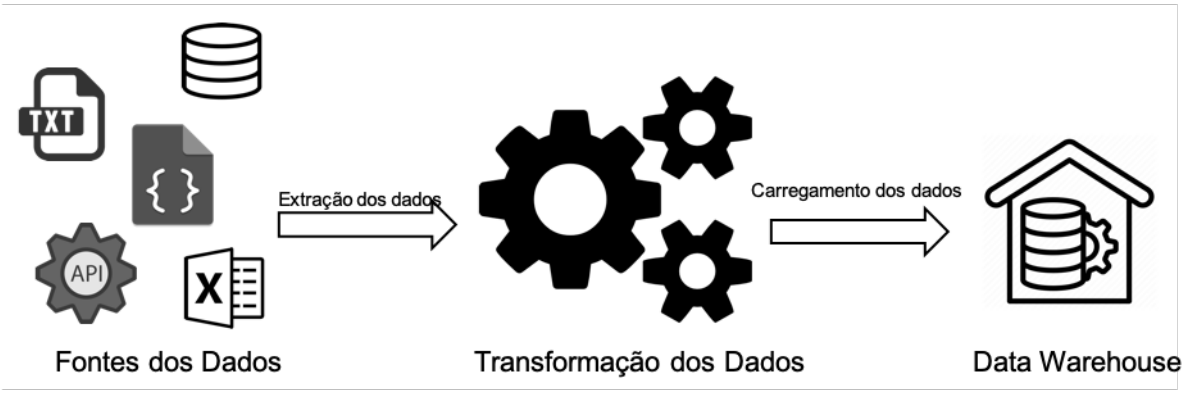


Figura 3: Processo de transformação de dados - BI.

2.2.1 *Ferramentas de Business Inteligence*

As ferramentas Tableau e Power BI, têm por base atingir os mesmos objetivos, isto é, transformar dados em resultados significativos que auxiliam nos processos de tomada de decisão. No entanto, existem certos aspetos que diferenciam estas ferramentas.

No caso da ferramenta Tableau, esta permite o acesso a inúmeras bases de dados e servidores, suporta grandes volumes de dados e assegura uma elevada performance independentemente da quantidade de dados. Esta ferramenta tem compatibilidade com o sistema operativo Windows e macOS [9]. Em comparação, a ferramenta Power BI apenas permite o acesso a um número bastante limitado de bases de dados e servidores e apenas suporta até 10 *gigabytes* de dados, sendo que a única forma de ultrapassar esta restrição de tamanho, assenta na possibilidade de ter os dados na *cloud*. Relativamente à performance, esta é muito dependente da quantidade de dados com que se está a lidar, existindo um decréscimo de

performance à medida que o volume de dados aumenta. Em termos de compatibilidade, esta ferramenta apenas roda no sistema operativo Windows [10].

2.2.2 *Aplicação de Business Inteligence na Área da Saúde*

Recolher, transformar, organizar, analisar e distribuir dados de forma a que estes auxiliem processos de tomada de decisão, são os passos do processo que se intitula por *Business Inteligence* (BI). Este processo permite transformar elevadas quantidades de dados não tratados em informação útil, que se torna muito valiosa para auxiliar em processos de tomada de decisão. [11]

Cada vez mais, a quantidade de aplicações a servirem-se do conceito de BI tem vindo a aumentar, derivado aos benefícios que este oferece. Através de várias fontes de informação, enormes quantidades de dados são retirados com vista a serem analisados. As mais variadas áreas, desde as plataformas *e-commerce*, *e-gov*, saúde, entre outras, usufruem destas análises de modo a que seja possível melhorar os seus serviços, tornando-os mais eficientes e fidedignos. [12]

Relativamente às áreas da saúde, também devido ao aumento de aplicações móveis relacionadas com estas, são gerados mais dados a cada dia que passa e utilizar BI é uma solução que permite tornar a tomada de decisão um processo mais facilitado. [13] Esta é uma grande mais valia, pois permite reduzir o risco de erro humano, salvaguardando assim todos os envolvidos em processos de saúde e também agilizar todos os processos envolvidos.

Assim, recorrendo à extração, tratamento e carregamento dos dados disponíveis, é possível proporcionar um melhor funcionamento em todas as fases que envolvem procedimentos relacionados com as áreas da saúde.

3

METODOLOGIAS DE INVESTIGAÇÃO E TECNOLOGIAS

Neste capítulo são introduzidas as metodologias seguidas ao longo da elaboração da presente dissertação. São também explicadas as escolhas tecnológicas utilizadas no desenvolvimento da aplicação.

3.1 METODOLOGIA DESIGN SCIENCE RESEARCH

De forma a desenvolver novo conhecimento relativo ao tema desta dissertação, com vista a produzir uma solução confiável, surgiu a necessidade de aplicar uma metodologia de pesquisa.
A utilização de metodologias de pesquisa auxilia no processo de recolha de informação, tornando esta mais fidedigna, e também na conceção da solução final. Assim sendo, a metodologia escolhida foi a *Design Science Research* (DSR).
Esta metodologia tem como objetivo auxiliar na construção e avaliar soluções na área das tecnologias de informação, de forma a que seja possível os profissionais destas áreas processarem informação organizacional e desenvolver acções que implementem estas informações [14].

A metodologia DSR, é composta por 6 distintas fases [15]. De seguida são apresentadas estas fases, bem como uma breve explicação do intuito de cada uma delas:

1. Definição do problema e motivação
    - Elaboração da definição do problema proposto e explicação da necessidade de explorar o mesmo.
2. Definição dos objetivos da solução
    - Através da análise de pesquisas e abordagens existentes acerca do tema, define-se os objetivos da solução.
3. *Design* e Desenvolvimento
    - Definição do protótipo e desenvolvimento da solução final. Testes regulares durante o desenvolvimento da aplicação são também expectáveis.

4. Demonstração
    - Juntamente das partes interessadas na solução final, conceber demonstrações.
5. Avaliação
    - Revisão da solução final juntamente com os especialistas da área do tema da solução.
6. Comunicação
    - Esta etapa pode seguir três diferentes abordagens: contribuições em conferências e jornais, artigos ou *workshops* de demonstração e aplicação da solução.

Tendo por base esta metodologia, nesta dissertação foram então seguidos os seis passos acima mencionados.

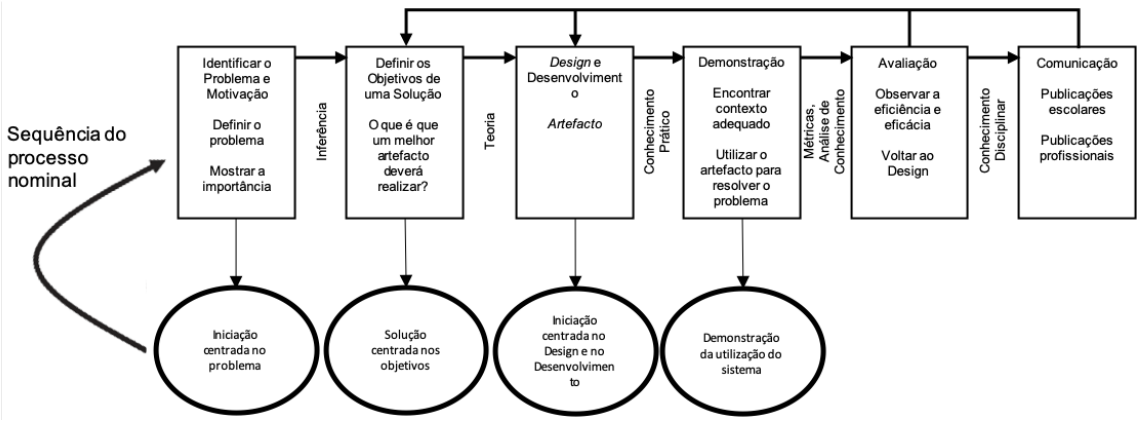


Figura 4: Metodologia Design Science Research - Adaptado de [1].

3.2 METODOLOGIA DE PROVA DE CONCEITO

A metodologia de prova de conceito (*proof of concept* - PoC) consiste em analisar uma ideia ou produto que se pretende implementar.

No que toca à presente dissertação, através desta metodologia, é possível aferir a viabilidade do produto de software desenvolvido ser implementado em contexto real, recorrendo a um protótipo. Com isto, é também possível receber *feedback* acerca do produto, antecipando assim futuros problemas técnicos e de design, que apenas se tem perceção no momento de se iniciar a utilização do mesmo, logo no início do desenvolvimento.[16]

Resumindo, a documentação elaborada através da POC, permite demonstrar o potencial de um produto ser bem sucedido, proporcionando confiança e certeza no produto em desenvolvimento. De modo a elaborar a POC relativa à aplicação móvel desenvolvida, foi feita

a respetiva análise SWOT (*Strengths*, *Weaknesses*, *Opportunities* e *Threats*). Nesta análise as letras "S"e "W"representam a análise dos pontos fortes e fracos, respetivamente, enquanto as letras "O"e "T"são relativas às oportunidades e às ameaças, respetivamente, detetadas através da análise.[17]

3.3 TECNOLOGIA DE BACKEND

A escolha da tecnologia que dá suporte à aplicação, ou seja, o *backend*, teve por base critérios como a ajuda e suporte, ferramentas de desenvolvimento e gestão de *packages*, integração com bases de dados e *performance*. Entre várias tecnologias de *backend*, a que mais se destacou nestes campos foi o Node.js [18], *framework* JavaScript que permite ter um ambiente de desenvolvimento para executar JavaScript no lado do servidor. Esta tecnologia tem vindo a tornar-se cada vez mais popular ao longo dos anos, bem como o JavaScript [19], razão que acentuou o destaque em relação a outras tecnologias, tendo assim sido tomada a decisão de utilizar esta tecnologia como *backend* da aplicação.

Através deste *framework*, foi desenvolvida uma API RESTful, de modo a permitir a comunicação cliente-servidor. Utilizando métodos como o GET, POST, PUT e DELETE, é feita a comunicação entre a aplicação e os *endpoints* da API RESTful, de modo a executar as acções pretendidas.

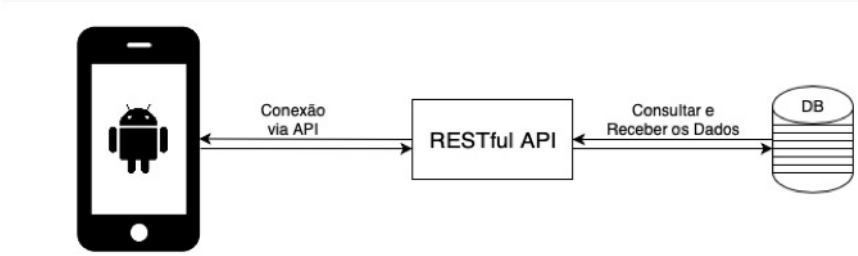


Figura 5: Arquitetura RESTful.

3.4 TIPO DE APLICAÇÃO

Existem três diferentes tipos de aplicação: aplicações *native*, aplicações *web-based* e aplicações *hybrid*.

As aplicações *native* envolvem a criação de uma aplicação para cada sistema operativo, por exemplo, uma aplicação para o sistema operativo Android e outra para o sistema operativo macOS. No desenvolvimento de aplicações para estes dois sistemas operativos, que são mais utilizados a nível mundial [20], são usadas as linguagens de programação nativas

destes mesmos sistemas operativos. No caso do sistema operativo Android, as aplicações podem ser desenvolvidas em Java ou Kotlin, já para o sistema operativo macOS as linguagens de programação nativas utilizadas são Swift ou Objective-C. Este tipo de aplicações são desenvolvidas com vista a atingir alta *performance* e a proporcionar uma interface adaptada aos diferentes sistemas operativos.

No que toca às aplicações *web-based*, estas são na realidade uma página web acedida pelo *browser*. São desenvolvidas como um *website*, porém com uma interface adaptada ao aparelho móvel. A vantagem destas aplicações é que não é necessário instalar, podendo ser acedidas através de um atalho ou URL no *browser*, não ocupando assim memória, e o desenvolvimento das mesmas é substancialmente mais rápido. Este tipo de aplicações não funciona sem acesso à Internet e não pode ser disponibilizado nas lojas de aplicações dos diferentes sistemas operativos.

Em relação às aplicações *hybrid*, estas são uma mistura entre os dois tipos de aplicações anteriormente referidos. São utilizadas tecnologias de programação web, que através do *framework* escolhido, compila e traduz o código para as diferentes plataformas sobre as quais a aplicação vai correr, tornando estas visualmente muito similares com as aplicações nativas. Comparando com as aplicações *native*, estas não possuem índices de *performance*, rapidez e otimização tão elevados, no entanto, o seu tempo de desenvolvimento e carregamento são mais rápidos e exige menos manutenção de código.

Resumindo, tendo em conta os prós e contras dos tipos de aplicações falados acima, a aplicação desenvolvida no âmbito desta dissertação de mestrado, é do tipo *hybrid*.

3.4.1 *Frameworks Hybrid*

Existem vários *frameworks* para o desenvolvimento de aplicações *mobile* do tipo *hybrid*. Entre elas, destacam-se o Flutter, React Native, Ionic, Framework 7 e Phone Gap [21]. Tendo em conta projetos já desenvolvidos, a adaptação ao *framework* React Native torna-se mais acessível, o que faz com que este seja o *framework* escolhido para o desenvolvimento da aplicação *mobile*.

3.5 BASE DE DADOS

As bases de dados, responsáveis por armazenar informação nos sistemas dos computadores, são-nos úteis no âmbito do desenvolvimento do software elaborado, de modo a disponibilizar e manter informação atualizada a todo o instante. Entre vários tipos de bases de dados, destacam-se as bases de dados relacionais (SQL) e as bases de dados não relacionais (NoSQL).

Nas bases de dados relacionais, também conhecidas por bases de dados SQL (*Structured Query Language*), são utilizadas tabelas que se relacionam entre si, através de chaves primárias e estrangeiras, para guardar a informação. Estas tabelas são compostas por colunas, que definem o conteúdo de cada tabela, e por linhas, que correspondem aos registos que cada tabela contém.[22]

As bases de dados relacionais mantêm um *schema* estático, tipos de dados fixos e são conhecidas pelas suas propriedades de atomicidade, consistência, isolamento e durabilidade (ACID). Abaixo são explicadas mais detalhadamente as propriedades ACID:

- Atomicidade: todas as operações de uma transação são executadas com sucesso, caso contrário, nenhuma é executada.
- Consistência: os dados não são afetados quando uma transação não é concluída, sendo estes revertidos para o seu estado inicial.
- Isolamento: as transações são independentes entre si, não sendo possível transações terem conhecimento do resultado de outras antes destas terminarem.
- Durabilidade: Os dados alterados por uma transação são persistidos, mesmo havendo falhas no sistema, sendo apenas possível alterar estes através de outra transação.

Em relação às bases de dados não relacionais, estas possuem *schemas* dinâmicos e diferentes tipos de dados. É composta por coleções que contêm documentos, que por sua vez podem também conter documentos. Ao contrário das bases de dados relacionais, este tipo de bases de dados não são normalizadas, o que leva a uma leitura de dados mais rápida, no entanto, torna-se mais difícil garantir a correta estrutura dos dados nelas contidos.

Tendo em conta a complexidade das *queries* a executar de forma a obter os dados, o elevado número de transações envolvidas na aplicação e a necessidade de assegurar as propriedades ACID, o tipo de base de dados escolhido para dar suporte à aplicação é o relacional (base de dados SQL).

3.5.1 *Software de RDBMS*

Os RDBMSs (relational database management systems) são utilizados para fazer a gestão das bases de dados relacionais. Sendo que existem vários, optou-se pelo RDBMS MySQL, pois este é um software *open source*, tem compatibilidade com diferentes sistemas operativos (Linux, Windows e macOS), não tem limitação de quantidade de dados e é de fácil utilização. Esta escolha também se deve ao facto de ser um RDBMS ao qual já se está ambientado.

4

ANÁLISE E ABORDAGEM PROPOSTA

Sendo a documentação um dos aspectos principais do desenvolvimento de software, a análise de Requisitos é o primeiro passo a seguir. Através desta análise é possível documentar as necessidades e restrições dos futuros utilizadores do software e que devem ser consideradas durante todo o processo de desenvolvimento do mesmo. Neste capítulo, numa primeira fase é apresentada uma breve explicação do conceito de requisitos e os seus diferentes tipos, os objetivos e as restrições do software. De seguida, tendo por base o levantamento de requisitos efetuado, é exposto o modelo de domínio e diagramas de *use-case*. Por fim, e tendo em conta as diferentes etapas anteriormente mencionadas, é demonstrado o modelo lógico da base de dados.

4.1 REQUISITOS

"Condição necessária para a consecução de um certo fim"[23], assim pode ser definido o conceito de requisito. Estes têm extrema importância na elaboração de software e podem ser classificados como requisitos funcionais e requisitos não funcionais.

Os requisitos funcionais, independentes do design e de aspectos de implementação do software, caracterizam aquilo que o software deverá proporcionar aos utilizadores, as suas funções e informações. Já os requisitos não funcionais dizem respeito aos aspectos em termos de qualidade do software, tais como, *performance*, usabilidade, confiabilidade e robustez. [24]

Aquando da elaboração de requisitos, é necessário ter em atenção a sua importância, de modo a que seja possível definir a ordem pela qual os requisitos vão ser desenvolvidos. Para tal, foi seguido o método MoSCoW[25], no qual estão definidas quatro regras de prioritização:

- Must: Requisitos obrigatórios.
- Should: Requisitos que devem ser implementados.
- Could: Requisitos que não são necessários, mas são desejados.

- Won't: Requisitos que podem ser considerados posteriormente.

Tendo em conta estas regras e de forma a descrever concisamente os requisitos, estes são apresentados utilizando uma adaptação da *requirement shell* do modelo de Volere [26]:

Tabela 1: Adaptação da *requirement shell* do modelo de Volere

Requirements Shell	
Requirement:	**Requirement Type:**
Description:	
Rationale:	
Originator:	
Fit Criterion:	
Priority:	

Abaixo encontra-se uma breve explicação dos campos da tabela apresentada:

- **Requirement:** Número de identificação do requisito.
- **Requirement Type:** Tipo de requisito.
- **Description:** Descrição clara e concisa do requisito.
- **Rationale:** Justificação da existência do requisito.
- **Originator:** Quem originou o requisito.
- **Fit Criterion:** Critério em que se insere
- **Priority:** Define a prioridade de implementação do requisitos.

4.1.1 *Requisitos Funcionais:*

Tabela 2: Requisito Funcional #1

Requirements Shell	
Requirement: 1	**Requirement Type:** Funcional
Description: O auxiliar de saúde deverá poder fazer *login* utilizando credenciais válidas.	
Rationale: De modo a ser possível utilizar a aplicação móvel, é imprescindível que o auxiliar faça *login*.	
Originator: Introspeção	
Fit Criterion: Apenas utilizadores com credenciais válidas podem aceder à aplicação.	
Priority: Must	

Tabela 3: Requisito Funcional #2

Requirements Shell	
Requirement: 2	**Requirement Type:** Funcional
Description: O auxiliar de saúde deverá poder visualizar a lista de notificações não dispensadas.	
Rationale: De modo a manter a caixa e o *stock* de medicação corretos, o auxiliar de saúde deverá poder visualizar a lista de notificações não dispensadas.	
Originator: Entrevistado	
Fit Criterion: Aparece com fundo vermelho as notificações que ainda não foram dispensadas.	
Priority: Must	

Tabela 4: Requisito Funcional #3

Requirements Shell	
Requirement: 3	**Requirement Type:** Funcional
Description: O auxiliar de saúde deverá poder dispensar as notificações.	
Rationale: De modo a manter atualizada a todo o instante a lista de notificações, o auxiliar de saúde pode dispensar as notificações.	
Originator: Introspeção	
Fit Criterion: Notificação dispensada deixa de aparecer na lista de notificações.	
Priority: Must	

Tabela 5: Requisito Funcional #4

Requirements Shell	
Requirement: 4	**Requirement Type:** Funcional
Description: O auxiliar de saúde deverá poder alterar os dados da instituição.	
Rationale: No caso de alguma mudança relativa aos dados da instituição é expectável que estes permaneçam atualizados na aplicação móvel.	
Originator: Entrevistado	
Fit Criterion: Os dados da instituição são alterados com sucesso.	
Priority: Must	

Tabela 6: Requisito Funcional #5

Requirements Shell	
Requirement: 5	**Requirement Type:** Funcional
Description: O auxiliar de saúde deverá poder atribuir e remover medicação aos utentes.	
Rationale: Para que a ficha de medicação dos utentes esteja atualizada a todo o instante, é necessário que o auxiliar de saúde possa atribuir e remover medicação dos mesmos.	
Originator: Entrevistado	
Fit Criterion: É adicionado ou removido medicação da lista de medicação atual do utente.	
Priority: Must	

Tabela 7: Requisito Funcional #6

Requirements Shell	
Requirement: 6	**Requirement Type:** Funcional
Description: O auxiliar de saúde deverá poder visualizar e editar a ficha de informação individual dos utentes.	
Rationale: Para que a ficha de informação individual dos utentes esteja atualizada a todo o instante, é necessário que o auxiliar de saúde possa visualizar e editar a mesma.	
Originator: Entrevistado	
Fit Criterion: Os dados individuais de um dado utente são alterados.	
Priority: Must	

Tabela 8: Requisito Funcional #7

Requirements Shell	
Requirement: 7	**Requirement Type:** Funcional
Description: O auxiliar de saúde deverá poder visualizar o histórico e a ficha atual de medicação dos utentes.	
Rationale: Para fins informativos, é necessário que o auxiliar de saúde possa consultar o histórico e a ficha atual de medicação dos utentes.	
Originator: Entrevistado	
Fit Criterion: São apresentados os dados do histórico de medicação e a ficha de medicação atual.	
Priority: Must	

Tabela 9: Requisito Funcional #8

Requirements Shell	
Requirement: 8	**Requirement Type:** Funcional
Description: O auxiliar de saúde deverá poder visualizar o stock de medicação dos utentes.	
Rationale: Para fins informativos e de contagem, o auxiliar de saúde deverá poder verificar o número de caixas em stock de cada utente, bem como a data de validade mais próxima.	
Originator: Entrevistado	
Fit Criterion: São apresentados os dados de uma dada medicação relativo ao utente escolhido.	
Priority: Must	

Tabela 10: Requisito Funcional #9

Requirements Shell	
Requirement: 9	**Requirement Type:** Funcional
Description: O auxiliar de saúde visualiza a lista de utentes.	
Rationale: Para que o auxiliar possa escolher o utente ao qual quer consultar/editar informação.	
Originator: Introspeção	
Fit Criterion: É apresentada a lista de utentes.	
Priority: Should	

Tabela 11: Requisito Funcional #10

Requirements Shell	
Requirement: 10	**Requirement Type:** Funcional
Description: O auxiliar de saúde visualiza a lista de medicação.	
Rationale: Para que o auxiliar possa verificar os utentes que atualmente tomam uma determinada medicação, é necessário que seja apresentada a lista de todos os medicamentos que já passaram na instituição.	
Originator: Introspeção	
Fit Criterion: É apresentada a lista de medicação.	
Priority: Should	

4.1.2 *Requisitos Não Funcionais:*

Usabilidade

Tabela 12: Requisito Não Funcional #1

Requirements Shell	
Requirement: 1	**Requirement Type:** Não Funcional
Description: A aplicação móvel deve apresentar cores suaves.	
Rationale: Por forma a não prejudicar a visão do utilizador, este requisito visa uma utilização saudável da mesma para com o utilizador.	
Originator: Introspeção	
Fit Criterion: Para efeitos de teste, se 15 utilizadores utilizarem a aplicação por um período igual ou superior a 10 minutos e 5 deles apresentem dores de cabeça ou oculares, tonturas ou náuseas, então as cores da aplicação móvel são demasiado intensas.	
Priority: Must	

Tabela 13: Requisito Não Funcional #2

Requirements Shell	
Requirement: 2	**Requirement Type:** Não Funcional
Description: A aplicação móvel disponibiliza filtros de pesquisa nas listas.	
Rationale: Deste modo, o utilizador pode encontrar mais rapidamente e de forma mais fácil o resultado pretendido.	
Originator: Introspeção	
Fit Criterion: É disponibilizada uma lista de filtros aos utilizadores.	
Priority: Should	

Tabela 14: Requisito Não Funcional #3

Requirements Shell	
Requirement: 3	**Requirement Type:** Não Funcional
Description: As caixas de texto da aplicação móvel disponibilizam informação que auxilia ao seu preenchimento.	
Rationale: Apresentando informação nas caixas de texto, torna-se mais fácil para o utilizador preencher corretamente as mesmas.	
Originator: Introspeção	
Fit Criterion: As caixas de texto possuem informação auxiliar.	
Priority: Should	

Tabela 15: Requisito Não Funcional #4

Requirements Shell	
Requirement: 4	**Requirement Type:** Não Funcional
Description: O acesso às páginas da aplicação deve ser simples e intuitivo, sendo que as mesmas devem ser acedidas com um simples clique.	
Rationale: As páginas devem ser acedidas com um simples clique.	
Originator: Introspeção	
Fit Criterion: Navegação simples e intuitiva.	
Priority: Should	

Operacionais

Tabela 16: Requisito Não Funcional #5

Requirements Shell	
Requirement: 5	**Requirement Type:** Não Funcional
Description: A aplicação móvel recupera a última sessão no caso de não ter sido efetuado *logout*.	
Rationale: Deste modo não é necessário fazer *login* sempre que se abre a aplicação.	
Originator: Introspeção	
Fit Criterion: Fechar a aplicação com *login* efetuado e voltar a executar a mesma.	
Priority: Must	

4.2 Modulação do Software

Depois de ultrapassada a fase de análise de requisitos e antes de passar à implementação do software, existe ainda uma fase intermédia, a modulação do software. Nesta secção serão apresentados e explicados o modelo de domínio e o diagrama de casos de uso, tipicamente conhecido por diagrama de *Use Cases*.

4.2.1 *Modelo de Domínio*

Tendo por base o levantamento de requisitos previamente elaborado e de modo a obter conhecimento relevante acerca do domínio do tema, tais como todas as entidades envolvidas e os seus relacionamentos, torna-se fundamental a existência de um modelo que nos ajude a raciocinar sobre o problema, sendo assim possível visualizar de melhor forma as regras de negócio do software a desenvolver.

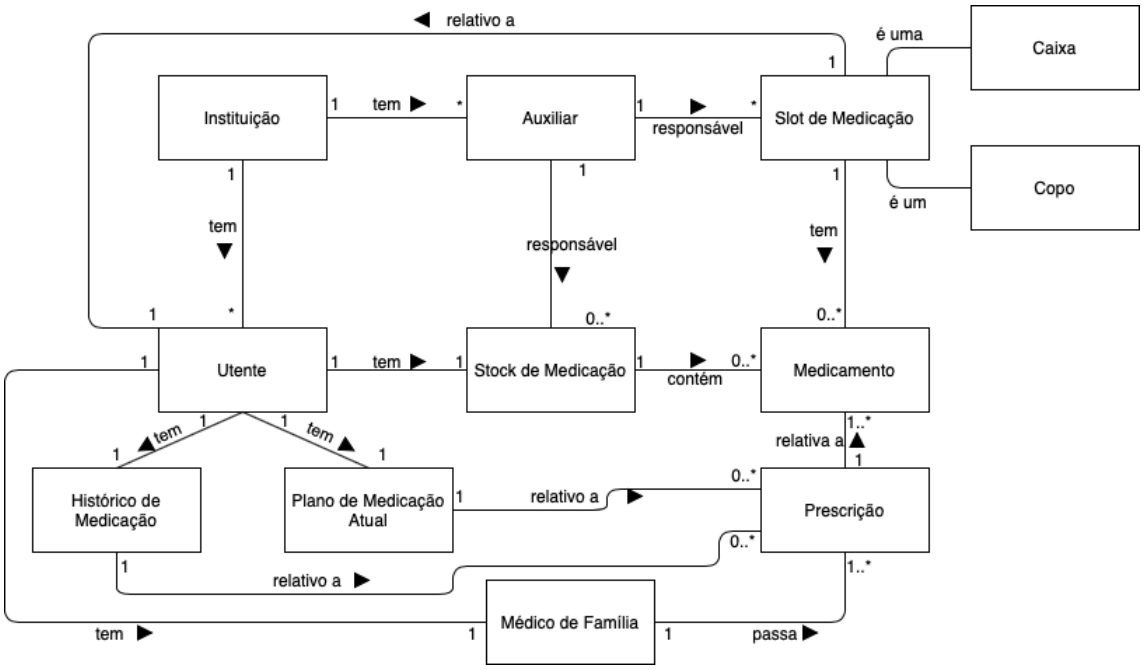


Figura 6: Modelo de Domínio.

Neste modelo, é possível retirar conclusões fundamentais para a conceção do software. Entre todas as entidades presentes no modelo de domínio existem cinco principais: Instituição, Auxiliar, Utente, Stock de Medicação e Slot de Medicação.

Abaixo são explicados os relacionamentos de todas as entidades do modelo de domínio.

1. Instituição:
   - A Instituição é a principal entidade deste modelo, sem esta não faria sentido desenvolver o software proposto;
   - Uma Instituição tem vários utentes;
   - Numa dada instituição trabalham vários auxiliares.
2. Auxiliar:
   - O Auxiliar é responsável por gerir os *slots* de medicação, de modo a preparar o planeamento semanal;
   - O stock de medicação tem de ser verificado também pelo Auxiliar, de modo a que esta não acabe, sendo sua tarefa verificar quantidades e datas de validade.
3. Utente:
   - O Utente tem a si associado o seu histórico de medicação;
   - Tem também registo do seu plano de medicação atual;

- Cada Utente tem o seu próprio stock de medicação.
- Todo o Utente tem a si associado um médico de família.

4. Stock de Medicação:
   - O Stock de medicação inclui todas as caixas de medicação associados a cada utente;
5. Slot de Medicação:
   - Um Slot de Medicação é um copo, uma caixa ou a junção de ambos;
   - Possui diferentes medicamentos;
   - Cada Slot de Medicação corresponde a um único utente.
6. Histórico de Medicação:
   - O histórico de medicação de cada utente;
   - Importante para que seja possível o médico ter o máximo de informação disponível sobre o utente.
7. Plano de Medicação Atual:
   - Plano de medicação atual utilizado para fazer o planeamento da medicação semanal;
   - É também importante par ao médico ter conhecimento do que o paciente está a tomar.
8. Médico de Família:
   - Cada utente tem a si associado um médico de família;
   - O médico de família é a principal entidade que segue regularmente o estado de saúde dos utentes.
9. Prescrição:
   - Prescrição é habitualmente feita pelo médico de família;
   - Poderá ser feita também por um médico especialista, sendo que este não tem de estar no sistema;
   - A prescrição é utilizada para manter o plano de medicação atualizado.
10. Medicamento:
    - O medicamento é utilizado nas prescrições;
    - Cada caixa do stock de medicação é relativa a um medicamento.

11. Caixa e Copo:
    - Locais onde é colocada a medicação para os períodos definidos por cada instituição.

4.2.2 *Diagrama de Use Cases*

Os diagramas de *Use Cases*, funcionam como uma abstração, ajudando a definir o comportamento do software sem que a implementação do mesmo seja necessária. Através de um diagrama com o conjunto de todos os *use cases*, é possível definir a comunicação entre utilizadores e software, modelando assim todo o contexto geral do software. Com este tipo de diagramas, o diálogo entre o cliente e quem desenvolve o software, torna-se mais facilitado, o que leva a que seja possível reduzir o tempo de conceção do mesmo.

Em primeiro lugar foi elaborado um diagrama de *use case* geral, de modo a que seja possível identificar os diferentes grupos de interação.

Num ponto de vista geral, como se pode ver na Figura 7, identificamos que o software será composto por quatro grupos: Utente, Notificação, Medicação e Instituição. Todos estes grupos são passíveis a interações por parte de um utilizador, que no âmbito do presente tema é o auxiliar de saúde. Essas interações são descritas de um modo geral dentro dos diferentes grupos da Figura 7.

Nas figuras 8, 9, 10 e 11, são apresentados os diagramas de *use case* mais detalhados, relativos aos quatro grupos anteriormente mencionados. Nestes diagramas é possível analisar e obter uma informação geral acerca das funcionalidades que o software irá disponibilizar para o controlo de cada grupo.

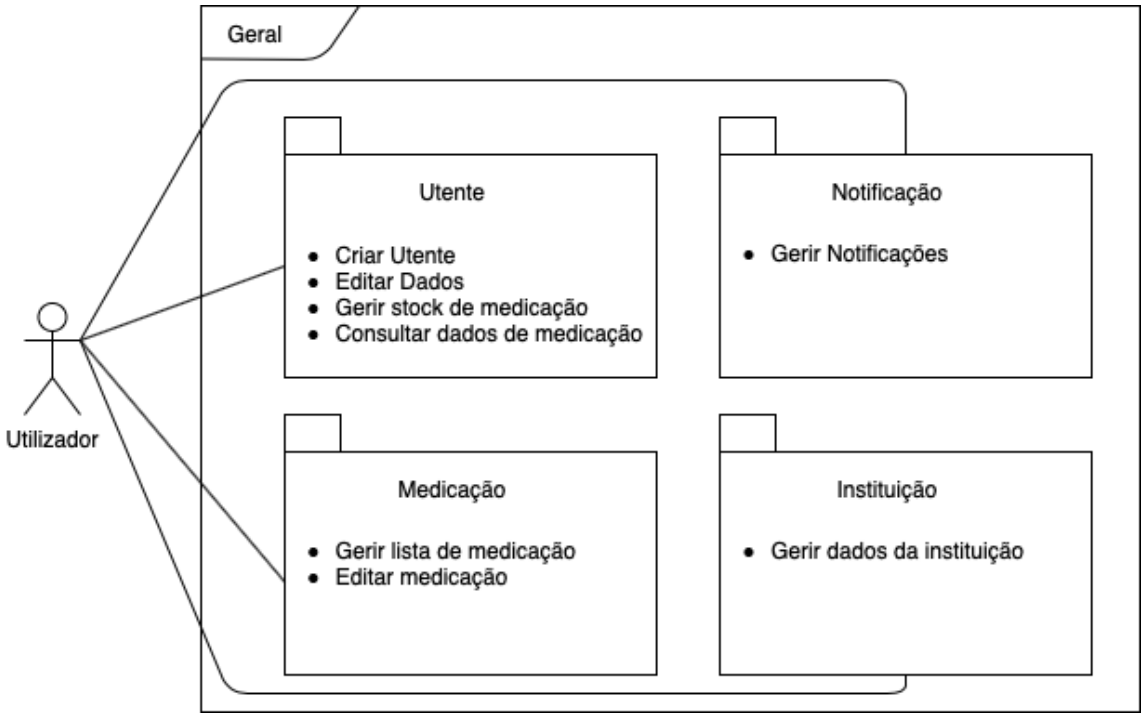


Figura 7: Diagrama de *use case* geral.

No que toca ao grupo do Utente (Figura 8), o auxiliar de saúde, representado na mesma como o Utilizador, interage com todos os dados acerca da lista de utentes e dados dos mesmos, sendo assim possível editar as informações acerca de utentes, tais como dados pessoais e atualizações na ficha de medicação individual, e também criar novos utentes. Deste modo a lista de utentes estará atualizada a todos os instantes e disponível para todos os auxiliares de saúde que tenham *login* no software. Todo o stock de medicação do utente é também controlável, de modo a adicionar novo *stock* e dar baixa de medicação já terminada ou com data de validade expirada. De modo a obter constantemente informação acerca do historial de medicação e da medicação atual, o utilizador pode também consultar dados sobre essas informações.

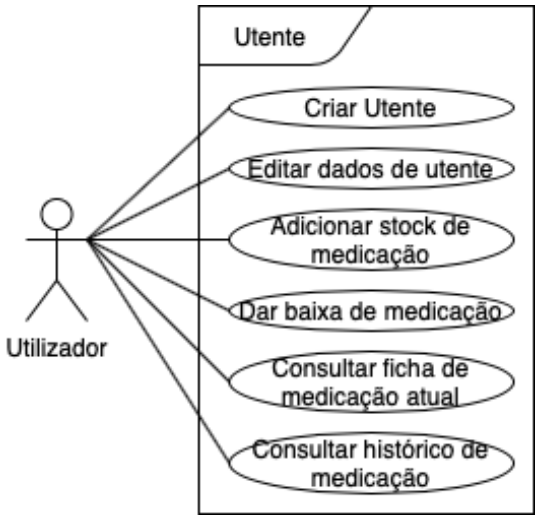


Figura 8: Diagrama de *use case* do Utente.

Na Figura 9 é evidenciado o controlo total que o auxiliar de saúde tem sobre a medicação. Adicionar nova medicação, adicionar forma farmacêutica e dosagem a medicação já existente ou eliminar medicação da lista, são as funções que o mesmo pode exercer relativamente ao grupo da medicação. Este grupo é fundamental no que toca à atualização da ficha de medicação dos utentes.

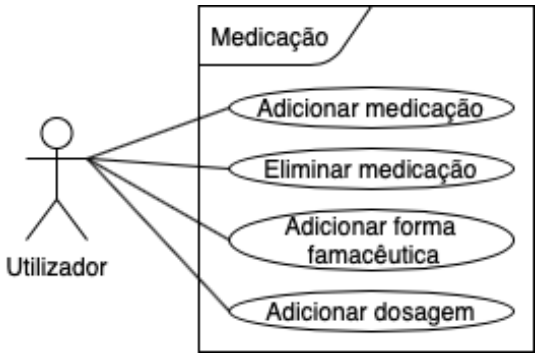


Figura 9: Diagrama de *use case* da Medicação.

Em relação ao grupo das notificações, este é marcado pela importância que tem em todo o planeamento da medicação. Aqui é possível ter acesso a toda a informação, atualizada diariamente, de modo a que toda a administração de medicação aos utentes seja feita sem que existam falhas, tanto a nível de falta de stock, como de preenchimento da caixa de medicação. Sendo assim, é possível marcar as notificações relativas ao planeamento, à medicação em fim de validade e à medicação em fim de stock, como concluídas (vistas/realizadas), tanto a nível do utente como no geral.

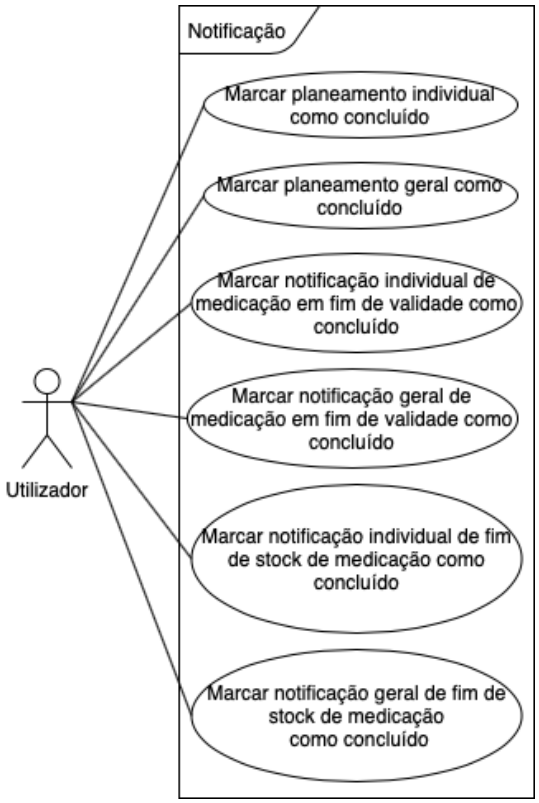


Figura 10: Diagrama de *use case* da Notificação.

O grupo da Instituição é responsável por manter atualizadas as informações acerca da instituição, tanto a nível de dados da mesma, como dos dias de planeamento e dos períodos de administração da medicação.

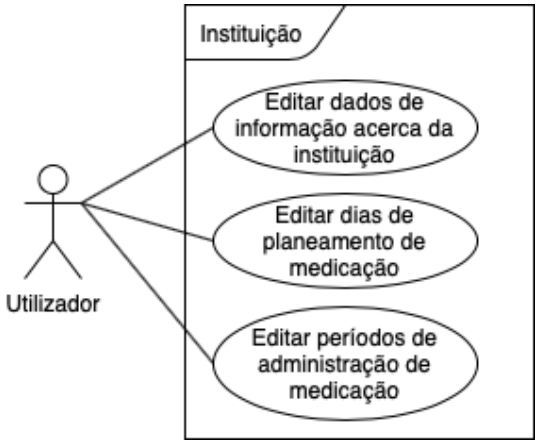


Figura 11: Diagrama de *use case* da Instituição.

4.3 ABORDAGEM PROPOSTA

De modo a dar resposta aos requisitos levantados e tendo a base de dados definida, é necessário fazer o planeamento da abordagem a implementar. Começando no *backend*, o qual foi desenvolvido em Node.js, é essencial entender a forma de como este comunica com a base de dados e com o *frontend*, bem como toda a sua estrutura.
No que toca à ligação à base de dados, esta é feita através do ORM (*object-relational mapping*) Sequelize.
No que toca à comunicação entre o *frontend* e o *backend*, esta é feita através de uma API RESTful, que recorrendo das operações CRUD (*create*, *read*, *update* e *delete*), permite um acesso dinâmico aos dados, pois esta aplicação envolve a manipulação de dados e não apenas a apresentação de dados estáticos. Assim sendo, verifica-se que existe uma separação entre o cliente e o servidor, o que leva a que seja possível que estes evoluam individual e independentemente.

5

DESENVOLVIMENTO DA APLICAÇÃO MÓVEL

Tendo em conta a abordagem proposta, de seguida são apresentadas e explicadas as diferentes partes do desenvolvimento do software e toda a sua estrutura.

Nas três secções que se seguem, em primeiro lugar é explicado o desenvolvimento da base de dados, tendo por base algumas regras do processo de normalização. Na segunda e terceira secções deste capítulo são explicadas a estrutura do *backend* e do *frontend*, respetivamente, bem como os *frameworks* e principais *packages* utilizados no desenvolvimento.

5.1 BASE DE DADOS

Como referido anteriormente na presente dissertação, a base de dados escolhida para dar suporte à camada de dados, é do tipo relacional, SQL.

Através do levantamento de requisitos, do modelo de domínio e dos diagramas de *use case* elaborados e aprovados pelos *stakeholders*, foi então definido o modelo lógico da base de dados.

Na elaboração deste modelo foram seguidas as três primeiras regras do processo de normalização de uma base de dados. De seguida é explicado cada uma destas regras, bem como a sua aplicação.

Por último, é apresentada explicação de cada tabela, bem como o modelo lógico resultante.

5.1.1 *Normalização*

A normalização de uma base de dados é o método de organização dos dados da mesma, desde a criação de tabelas até ao relacionamento entre estas. Este conjunto de regras proporciona uma base de dados mais flexível, eliminando a redundância e aumentando a integridade dos dados, e aumentando também o seu desempenho.

A redundância resulta na necessidade de alterar os mesmos dados em mais que um local e também num excesso de espaço ocupado em disco, levando a problemas de manutenção

de dados que por vezes se tornam muito difíceis de ultrapassar. São também criadas dependências que dificultam o acesso à localização dados.

Como tal, por forma a resolver estes problemas, entre outros, foram criadas algumas regras para a normalização de dados. Nesta dissertação, o modelo lógico elaborado encontra-se na terceira forma normal, o que significa que foram aplicadas as três primeiras regras.

1ª Forma Normal

Uma tabela considera-se na 1ª forma normal se e só se cada linha contiver apenas um valor para cada atributo, eliminando assim atributos multi-valor e atributos compostos. Ou seja, uma tabela nesta forma normal, apenas deverá ser composta por atributos atómicos, não podendo ser composta por relações dentro de relações. Assim sendo, os conjuntos repetidos nestas tabelas dão origem a uma nova tabela e são identificados através da sua chave primária.

2ª Forma Normal

Para que uma tabela esteja na 2ª forma normal, esta tem obrigatoriamente de estar na 1ª forma normal e todos os seus atributos descritores têm de depender funcionalmente da totalidade da sua chave primária. Entenda-se por atributos descritores, todos os atributos de uma tabela que não sejam chave candidata da mesma, que são meramente descritivos ou caracterizadores de uma entidade. Esta forma normal resulta assim na criação de novas tabelas para conjuntos de valores que se aplicam a vários registos, sendo estas relacionadas através da chave estrangeira.

3ª Forma Normal

Se uma tabela estiver na 2ª forma normal e os atributos descritores não dependerem funcionalmente uns dos outros, diz-se que a tabela está na 3ª forma normal. Posto isto, é possível afirmar que os atributos descritores devem depender unicamente das chaves candidatas da relação, caso contrário deverá ser considerado colocar esses atributos numa nova tabela.

5.1.2 *Modelo Lógico*

O modelo lógico elaborado para dar suporte à camada de dados do software, depois do processo de normalização, é composto por doze tabelas. De seguida são apresentados e explicados todos os atributos das tabelas:

Institution

Nesta tabela estão presentes as informações de todas as instituições no software.

Tabela 17: Descrição da tabela Institution - Base de Dados

Coluna	Tipo	Descrição
InstitutionId	inteiro	Chave primária para cada registo
Name	texto	Nome da instituição
Address	texto	Morada da instituição
City	texto	Cidade da instituição
PostalCode	texto	Código Postal da instituição
PhoneNumber	inteiro	Número da instituição

Log

Nesta tabela estão presentes todas as atividades relevantes efetuadas pelos utilizadores do software.

Tabela 18: Descrição da tabela Log - Base de Dados

Coluna	Tipo	Descrição
LogId	inteiro	Chave primária para cada registo
Username	texto	Chave estrangeira para a tabela User (Username), que corresponde ao utilizador que criou o *log*
Date	data	Data correspondente à criação do log
Message	testo	Informação acerca do *log* criado

User

Nesta tabela estão presentes as informações de todos os utilizadores do software.

Tabela 19: Descrição da tabela User - Base de Dados

Coluna	Tipo	Descrição
Username	texto	Chave primária para cada registo
Password	texto	*Password* da conta do utilizador
TempPassword	texto - *nullable* - depois de definida uma password, este atributo passa a ser nulo	*Password* temporária da conta do utilizador
Type	inteiro	Tipo de utilizador (administrador/normal)
IntitutionId	inteiro	Chave estrangeira para a tabela Institution (InstitutionId), que corresponde à instituição ao qual o utilizador pertence

Notification

Nesta tabela estão presentes todas as informações de notificações do software.

Tabela 20: Descrição da tabela Notification - Base de Dados

Coluna	Tipo	Descrição
NotificationId	inteiro	Chave primária para cada registo
NotificationTypeId	inteiro	Chave estrangeira para a tabela NotificationType (NotificationTypeId), que corresponde ao tipo de notificação do registo
Date	data	Data de criação notificação
Status	inteiro	Estado da notificação (ativa/dispensada)
InstitutionId	inteiro	Chave estrangeira para a tabela Institution (InstitutionId), que corresponde à instituição à qual a notificação pertence

NotificationType

Nesta tabela estão presentes os diferentes tipos de notificações lançadas pelo software.

Tabela 21: Descrição da tabela NotificationType - Base de Dados

Coluna	Tipo	Descrição
NotificationTypeId	inteiro	Chave primária para cada registo
Title	text	Título da notificação
Description	text	Descrição da notificação

Medicine

Nesta tabela estão presentes todos os medicamentos guardados no software por todas as instituições.

Tabela 22: Descrição da tabela Medicine - Base de Dados

Coluna	Tipo	Descrição
MedicineId	inteiro	Chave primária para cada registo
Name	text	Nome do medicamento
Dosage	texto	Dosagem do medicamento
PharmForm	texto	Forma farmacológica do medicamento
InstitutionId	inteiro	Chave estrangeira para a tabela Institution (InstitutionId), que corresponde à instituição à qual a medicação pertence

PlanDay

Nesta tabela estão presentes os dias de planeamento ativos em cada instituição existente no software.

Tabela 23: Descrição da tabela PlanDay - Base de Dados

Coluna	Tipo	Descrição
PlanDayId	inteiro	Chave primária para cada registo
Name	texto	Dia de planeamento
InstitutionId	inteiro	Chave estrangeira para a tabela Institution (InstitutionId), que corresponde à instituição ao qual o dia de planeamento pertence

Period

Nesta tabela estão presentes os períodos de administração de medicação para cada instituição existente no software.

Tabela 24: Descrição da tabela Period - Base de Dados

Coluna	Tipo	Descrição
PeriodId	inteiro	Chave primária para cada registo
Name	texto	Período do dia
Status	inteiro	Estado (ativo/inactivo)
Type	inteiro	Tipo de período correspondente (caixa ou copo)
InstitutionId	inteiro	Chave estrangeira para a tabela Institution (InstitutionId), que corresponde à instituição ao qual o período do dia pertence

Box

Nesta tabela estão presentes todas as embalagens relativas aos utentes das diferentes instituições.

Tabela 25: Descrição da tabela Box - Base de Dados

Coluna	Tipo	Descrição
BoxId	inteiro	Chave primária para cada registo
TotalQtt	texto - *nullable* - este atributo pode ser nulo no caso de ser uma embalagem encomendada	Quantidade total da embalagem
ExpDate	data - *nullable* - este atributo pode ser nulo no caso de ser uma embalagem encomendada	Data de validade da embalagem
PatientId	inteiro	Chave estrangeira para a tabela Patient (PatientId), que corresponde ao utente ao qual pertence a embalagem
MedicineId	inteiro	Chave estrangeira para a tabela Medicine (MedicineId), que corresponde ao tipo de medicação da embalagem
Status	inteiro	Estado (Em uso, terminada ou encomendada)
AddedDate	data - *nullable* - este atributo pode ser nulo no caso de ser uma embalagem encomendada	Data de compra da embalagem
Verified	inteiro	Valor binário para auxilio das notificações

Medication

Nesta tabela estão presentes todas as informações relativas à medicação passada aos diferentes utentes do softwate.

Tabela 26: Descrição da tabela Medication - Base de Dados

Coluna	Tipo	Descrição
MedicationId	inteiro	Chave primária para cada registo
PatientId	inteiro	Chave estrangeira para a tabela Patient (PatientId), que corresponde ao utente ao qual a medicação foi prescrita
MedicineId	inteiro	Chave estrangeira para a tabela Medicine (MedicineId), que corresponde ao medicamento da medicação prescrita
StartDate	data	Data de início do tratamento
EndDate	data - *nullable* - No caso de ser medicação crónica, este atributo é nulo	Data de fim do tratamento
Cronic	inteiro	Valor binário (não crónica/crónica)

Prescription

Nesta tabela estão presentes todas as informações relativas aos períodos passados na prescrição de uma dada medicação de um utente.

Tabela 27: Descrição da tabela Prescription - Base de Dados

Coluna	Tipo	Descrição
PrescriptionId	inteiro	Chave primária para cada registo
MedicationId	inteiro	Chave estrangeira para a tabela Medication (MedicationId), que corresponde a uma determinada medicação prescrita
PeriodId	inteiro	Chave estrangeira para a tabela Period (PeriodId), que corresponde ao período para o qual a medicação foi prescrita
Quantity	texto	Quantidade prescrita para um determinado período

Patient

Nesta tabela estão presentes todos os utentes das diferentes instituições no software.

Tabela 28: Descrição da tabela Patient - Base de Dados

Coluna	Tipo	Descrição
PatientId	inteiro	Chave primária para cada registo
Name	texto	Nome do utente
BirthDate	data	Data de nascimento do utente
DeathDate	data - *nullable* - este atributo é apenas preenchido no caso de o utente ter falecido	Data de falecimento do utente
AdmissionDate	data	Data de admissão do utente na instituição
IdCardNumber	inteiro	Número do cartão de cidadão do utente
IdCardExpDate	data	Data de validade do cartão de cidadão do utente
VatNumber	inteiro	Número de contribuinte do utente
SocSecNumber	inteiro	Número da segurança social do utente
HealthNumber	inteiro	Número de saúde do utente
FamDocName	texto	Nome do médico de família do utente
ContPerName	texto	Nome do contacto de emergência do utente
ContPerNumber	inteiro	Número do contacto de emergência do utente
Background	texto	Histórico de saúde do utente
ImageUrl	texto	Fotografia do utente
InstitutionId	inteiro	Chave estrangeira para a tabela Institution (InstitutionId), que corresponde à instituição que o utente pertence
Gender	inteiro	Género (masculino ou feminino
Box	inteiro	Valor binário que indica se a caixa de medicação semanal do utente está completa
State	inteiro	Estado do utente (inativo, ativo ou falecido)

De seguida é ilustrado o modelo lógico com todas as tabelas acima explicadas bem como as relações entre as mesmas.

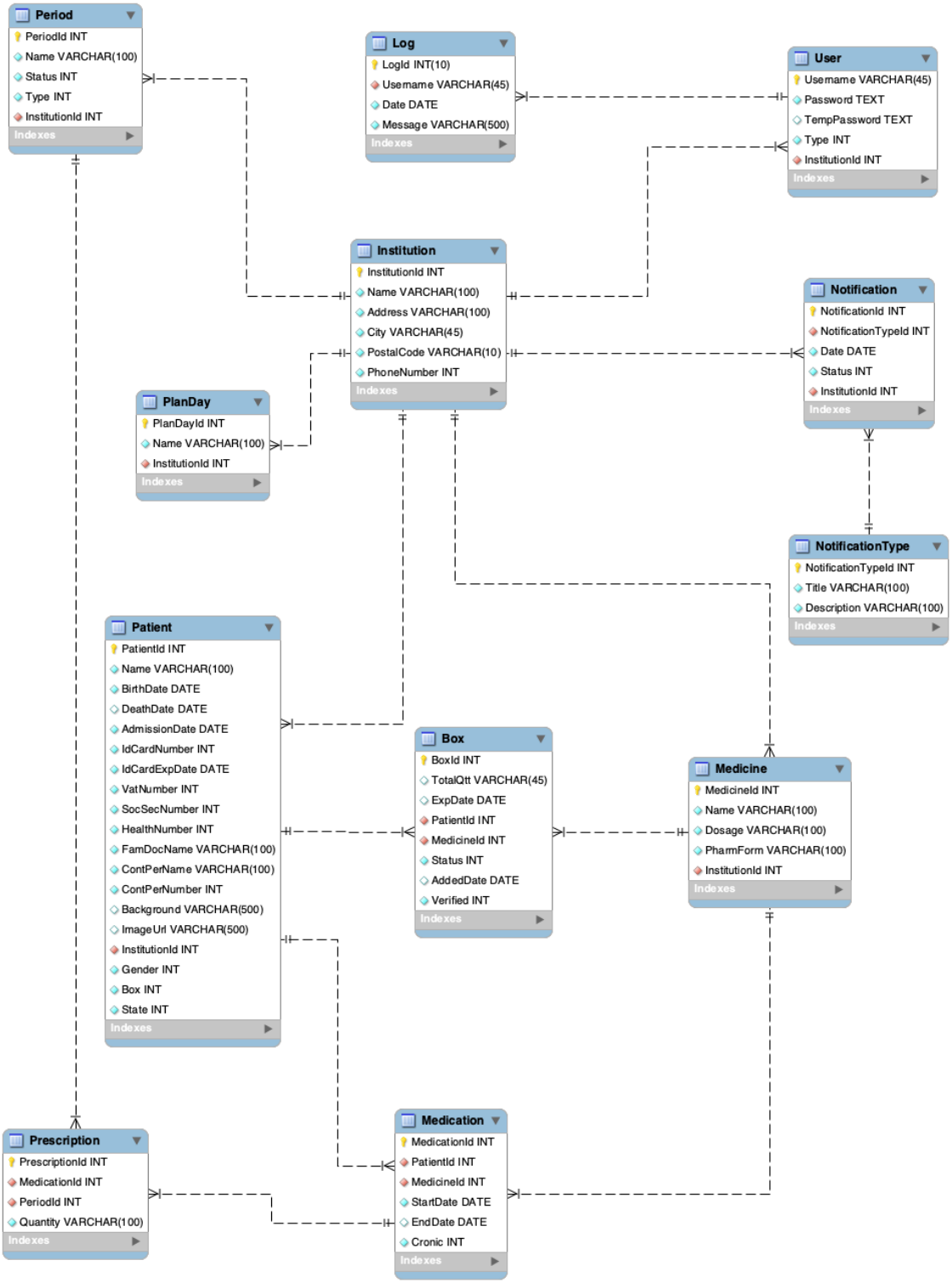


Figura 12: Modelo lógico da base de dados.

5.2 BACKEND

5.2.1 *Estrutura do Backend*

O *backend* está dividido em 10 partes fundamentais. Na imagem abaixo apresentada, podemos diferenciar todas as partes constituintes:

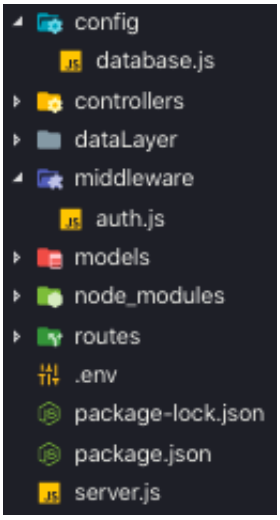


Figura 13: Estrutura do *backend*.

De seguida são explicadas as diferentes partes, bem como a sua importância no que ao *backend* concernem:

- Ficheiro server.js
  - Ficheiro onde o servidor é criado. Aqui é definida a porta onde o servidor irá correr, bem como as rotas que este irá disponibilizar;
  - As rotas disponibilizadas podem servir pedidos HTTP do tipo GET, POST, PUT, PATCH ou DELETE, sendo que no *backend* desenvolvido apenas são utilizados pedidos do tipo GET, POST, PUT e DELETE;
  - Estas rotas encontram-se divididas por responsabilidades, isto é:
    * Pedidos relativos aos pacientes: "/patients";
    * Pedidos relativos às instituições: "/institutions";
    * Pedidos relativos às notificações: "/notifications";
    * Pedidos relativos aos medicamentos: "/medicines";
    * Pedidos relativos aos períodos de administração: "/periods";

        * Pedidos relativos aos utilizadores: "/users".
    - Todas as rotas anteriormente mencionadas, têm as suas sub-rotas divididas da mesma forma na diretoria routes.
    - Neste ficheiro está também presente uma função que corre todos os dias à meia-noite, de modo a atualizar as notificações para todas as instituições do sistema.

- Ficheiro package.json
    - Neste ficheiro está definido o nome do servidor, a sua versão, descrição, o ponto de entrada da aplicação, o autor e por fim a lista de *packages* npm instalados como dependências. Este ficheiro é gerado automaticamente aquando da criação do projeto, tendo em conta os parâmetros anteriormente referidos.

- Ficheiro package-lock.json
    - Ficheiro gerado automaticamente que contém as dependências instaladas e as dependências exigidas por outros *packages*.

- Ficheiro .env
    - Ficheiro onde estão definidas as variáveis de ambiente a serem utilizadas, tais como porta do servidor, nome, dialecto, *host*, porta, utilizador e password da base de dados.

- Diretoria node_modules
    - Esta pasta contém os *packages* e as suas dependências instalados.

- Diretoria config
    - Contém o ficheiro database.js, o qual é utilizado na camada de acesso a dados, onde é feita a ligação à base de dados, fazendo a autenticação da mesma através do ORM Sequelize.

- Diretoria middleware
    - Esta pasta contém o ficheiro de autenticação responsável pela proteção das rotas;
    - A função de *middleware* presente neste ficheiro, tem acesso ao objeto de *request*, ao objeto de resposta e também à função que deverá ser executada de seguida;
    - A função seguinte apenas executa se a verificação feita através do método JSON Web Token for validada, caso contrário, retorna um código de erro;
    - Esta verificação valida se para um dado segredo, construído com recurso ao username presente no *request*, o *token*, passado também no *request*, é válido;

  - No caso da verificação ser concluída com sucesso, é então iniciada a função seguinte, que no contexto é a sub-rota protegida.

- Diretoria routes
  - Nesta pasta encontram-se todas as sub-rotas disponibilizadas, que são acedidas através das diferentes rotas definidas no ficheiro server.js;
  - Tal como mencionado anteriormente, à semelhança da divisão das rotas no ficheiro server.js, estas encontram-se também divididas por responsabilidades em diferentes ficheiros;
  - Em cada ficheiro, as sub-rotas que necessitam de estar protegidas são então precedidas pelo uso da função de *middleware*, garantindo apenas o acesso a utilizadores com credenciais válidas.

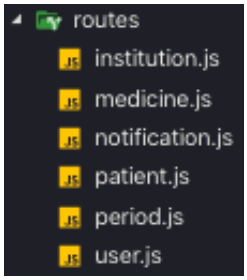


Figura 14: Diretoria 'routes'.

```
const express = require('express');
const router = express.Router();
const authMiddleware = require('../middleware/auth.js');

router.use(authMiddleware);

/* GET all patients */
router.get('/', (req, res) => {
  patient.getPatients(req.query.instId)
    .then(ls => {
      res.status(200)
        .jsonp(ls);
    })
    .catch(erro => {
      res.status(500)
        .jsonp(erro);
    });
});

/* GET patient by id */
```

```
router.get('/patient/', (req, res) => {
  patient.getPatient(req.query.id)
    .then(ls => {
      res.status(200)
        .jsonp(ls);
    })
    .catch(erro => {
      res.status(500)
        .jsonp(erro);
    });
});
```

Código 5.1: JavaScript - Exemplo da proteção de sub-rotas

- Diretoria controllers
  - Nesta pasta encontram-se as funções responsáveis pela parte lógica do sistema;
  - São elas que efetuam a manipulação dos dados recebidos das consultas à base de dados, feitas através da chamada das funções que fazem esse acesso;
  - Os dados enviados como resposta ao pedido efetuado são manipulados também nestas funções;
  - As funções encontram-se também divididas por responsabilidades em diferentes ficheiros.

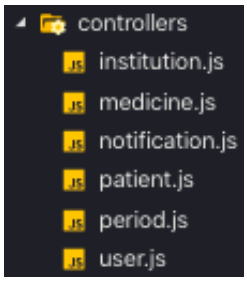


Figura 15: Diretoria 'controllers'.

```
module.exports.getMedicineById = async (params) => {
const DBResponse = await medicine.getMedicineById(params.allMedIds, params
    .date, params.patId);

let result = {};
for (const med of DBResponse.resultMeds) {
const medId = med.MedicineId;
if (params.expDate == 1) {
  const boxesDates = await box.getAllBoxExpDate(params.patId, medId,
      params.date);
```

```
    const qttNeed = boxesDates.length;
    result[medId] = { ...med, QttNeed: qttNeed, ExpDates: boxesDates };
  }
  else {
    const qttNeed = DBResponse.medIdsQtt[medId] === 0 ? 2 : 1;
    result[medId] = { ...med, QttNeed: qttNeed };
  }
  }
  return result;
  };
```

Código 5.2: JavaScript - Excerto do *controller* medicine

- Diretoria models
  - Esta pasta contém os modelos gerados pelo ORM sequelize;
  - Estes modelos são uma abstração que representam as tabelas da base de dados.

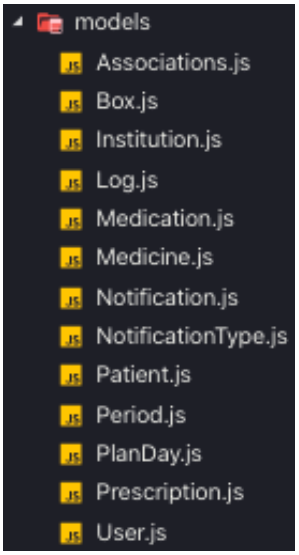


Figura 16: Diretoria 'models'.

```
const Sequelize = require('sequelize');
const db = require('../../config/database.js');

const Period = db.define('Period', {
  PeriodId: {
    type: Sequelize.INTEGER,
    allowNull: false,
    autoIncrement: true,
    primaryKey: true
```

```
    },
    Name: {
      type: Sequelize.STRING(100),
      allowNull: false
    },
    Status: {
      type: Sequelize.INTEGER,
      allowNull: false
    },
    Type: {
      type: Sequelize.INTEGER,
      allowNull: false
    },
    InstitutionId: {
      type: Sequelize.INTEGER,
      allowNull: false,
      references: {
        model: 'Institution',
        key: 'InstitutionId'
      }
    }
  }, {
      tableName: 'Period'
    }
  );

  module.exports = Period;
```

Código 5.3: JavaScript - Exemplo de modelo

- Diretoria dataLayer
  - Nesta pasta, encontram-se os ficheiros, também divididos por responsabilidades, que contêm as funções encarregues de executar as consultas à base de dados;
  - Aqui são utilizados os modelos gerados pelo ORM Sequelize.

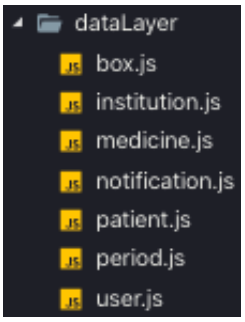


Figura 17: Diretoria 'dataLayer'.

```
const Sequelize = require('sequelize');
const { Institution, Period, PlanDay } = require('../models/Associations.
    js');
const { Op } = require("sequelize");

module.exports.getInstitutions = async () => {
  let result = [];
  await Institution.findAll({
    attributes: [
      [Sequelize.fn('DISTINCT', Sequelize.col('InstitutionId')), '
          InstitutionId']
    ],
    raw: true
  }).then(values => {
    result = values;
  }).catch(err => {
    result = err;
  });
  return result;
};
```

Código 5.4: JavaScript - Exemplo de consulta à tabela Institution da base de dados

5.2.2 *Frameworks e Principais Packages Utilizados*

Sequelize

O Sequelize é um ORM para Node.js baseado em promessas, compatível com MySQL[27], o sistema de gestão de base de dados relacionais escolhido para a gestão da base de dados. Através deste ORM, é possível diminuir o custo de mapear dados da base de dados em objetos do domínio do problema, facilitando e tornando mais rápido o processo de acesso, inserção, atualização e eliminação de dados da base de dados, pois este mapeamento é feito

de forma automática. A utilização deste ORM traz também benefícios em termos da quantidade e consistência de código escrito, da possibilidade de evitar a utilização de *queries* SQL e da abstração do mecanismo da base de dados. O maior desafio da utilização de ORM's incide no facto de que quando existe a necessidade de executar *queries* complexas, estas podem-se tornar mais lentas.

Express

O Express.js é um *framework* do Node.js que serve para criar abstrações de rotas e *middlewares* por forma a criar API's[28]. Este *framework* permite trabalhar com os diferentes verbos HTTP, bem como com diferentes bibliotecas.

Json Web tokens

O *package* jsonwebtokens do Node.js é utilizado para gerar e verificar *tokens*[29], de modo a garantir a segurança das rotas.

Node-cron

Através do node-cron, também este um *package* do Node.js, é possível agendar a execução de tarefas recorrendo à sintaxe cron[30]. No contexto do problema, este *package* é utilizado de modo a atualizar todos os dias à meia-noite as notificações de todas as instituições.

5.2.3 *Processo de Autenticação Utilizando JWT*

Quando um utilizador efetua *login*, no caso de este ser um utilizador permitido, é gerado um *token* único que fica a si associado e é enviado para a aplicação, onde é guardado no armazenamento da mesma, controlado pelo sistema operativo. Assim, em todos os pedidos feitos pela aplicação, é enviado para o *backend* no cabeçalho de requisição HTTP "*Authorization*"o *token* relativo ao utilizador com *login* efetuado. Caso este *token* seja válido, é enviada a informação pedida, caso contrário é enviada uma mensagem de erro. Aquando do *logout* de um utilizador, o *token* a si associado é eliminado, necessitando de ser gerado um novo num próximo *login*.

Assim, através dos *tokens* únicos gerados para cada utilizador, é garantido que apenas utilizadores com *token* válido podem fazer pedidos às rotas protegidas.

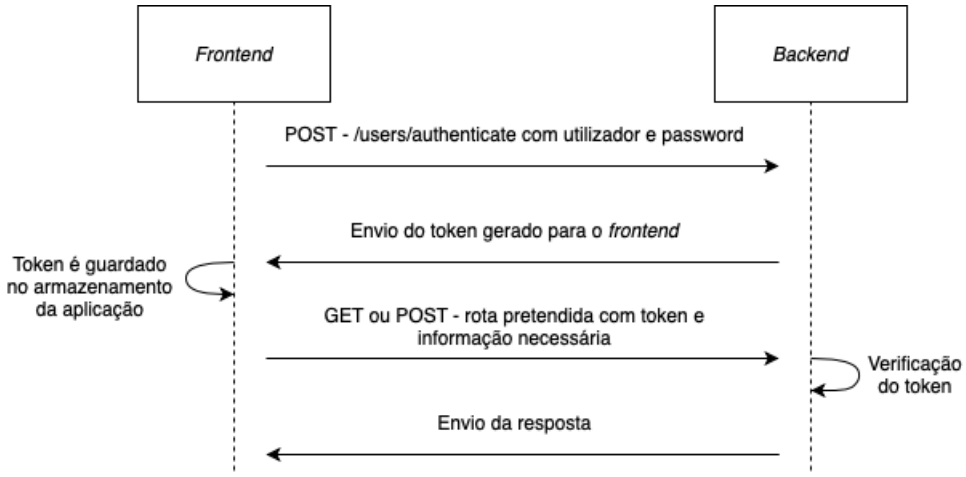


Figura 18: Processo de autenticação.

5.2.4 *Gestão de Notificações*

Como referido anteriormente, todos os dias à meia-noite corre a função que atualiza as notificações de todas as instituições. Esta foi a hora escolhida, pois o horário do planeamento da medicação, da verificação de caixas de medicação em fim de validade e de *stock* de medicação de utente a terminar, encaixa no período diurno do lar no qual foi feito o levantamento de requisitos.

Todas as instituições têm as suas notificações guardadas na tabela Notification da base de dados. Estas notificações, atualizadas diariamente, não mantêm registo histórico na base de dados, pois neste tipo de instituições não faz sentido ter acesso a notificações que deveriam ter sido dispensadas (termo utilizado quando uma notificação é vista) nos dias anteriores ao do momento. Assim sendo, no caso de uma notificação que não tenha sido dispensada no dia em que esta foi gerada, transita automaticamente para o dia seguinte, deixando no registo de *logs* essa indicação.

Existem três diferentes tipos de notificações:

- Notificação de planeamento de medicação;
- Notificação de caixa de medicação em fim de validade;
- Notificação de *stock* de medicação de utente a terminar.

Quando uma instituição é adicionada ao sistema, esta não tem referência de qualquer notificação a si associada na tabela Notification. Consoante os dias do planeamento semanal, a existência de utentes com prescrição de medicação e com stock de medicação, estas notificações são adicionadas. Uma vez dispensadas, estas notificações são apagadas da respetiva tabela.

Notificação de planeamento de medicação

As notificações de planeamento da medicação estão diretamente relacionadas com os utentes do sistema. Estes têm a si associados um atributo 'Box' que indica se a sua caixa de medicação está completa ou incompleta. Quando um utilizador é adicionado à instituição, este atributo binário tem o valor '1' por defeito, valor este que indica que o utente tem a caixa de medicação completa.

Quando a função de atualização das notificações corre para este tipo de notificação, é verificado se o dia em que está a correr é um dos dias definidos pela instituição como dia de planeamento de medicação. No caso de ser, para todos os utentes que têm medicação a fazer e que o atributo 'Box' se encontre a '1', é então feita a atualização do valor do respetivo atributo para '0', ou seja, utente com a caixa de medicação incompleta. Existindo utentes com medicação a fazer, é também criada uma entrada na tabela Notification com este tipo de notificação para a respetiva instituição com o atributo 'Status' a '0' (notificação não dispensada).

Aquando da verificação da notificação, é então apresentada a lista de todos os utentes com a caixa de medicação incompleta. Conforme a caixa de medicação de cada utente seja completa, é atualizado o atributo 'Box' de cada um para o valor de '1'. No final, tendo sido completa a caixa de medicação de todos os utentes, esta notificação é apagada da base de dados.

Notificação de caixa de medicação em fim de validade

Este tipo de notificações têm como intuito disponibilizar a lista dos utentes e as respetivas medicações com data de validade a terminar em menos de 7 dias, a contar do momento em que é feita a atualização das notificações.

No caso de um utente ter em stock pelo menos uma caixa de medicação em fim de validade e com o atributo 'Verified' a 0 (atributo que indica que até ao momento a caixa de medicação em questão ainda não foi verificada), é criada uma entrada na tabela 'Notification' relativa a este tipo de notificação e associada à respetiva instituição.

Quando um utilizador seleciona a opção de ver este tipo de notificações é apresentada a lista de utentes com pelo menos uma caixa de medicação em fim de validade e com o atributo 'Verified' a 0. Selecionando um utente, é apresentada a lista da medicação que está prestes a terminar, dando indicação da quantidade de caixas de cada medicamento com a validade a terminar, bem como a sua data de validade.

Dispensando a notificação das caixas selecionadas, o atributo 'Verified' destas mesmas caixas é colocado a '1' na tabela 'Box' e são adicionadas, nessa mesma tabela, tantas caixas quantas as que foram dispensadas, com o atributo 'Status' a 2 (status que indica que a caixa apenas entra nas contas de uma próxima atualização de notificações de stock a terminar, contando como caixa verificada para encomenda).

Notificação de stock de medicação de utente a terminar

Sempre que um utente tem menos de 2 caixas de um dado medicamento, que está incluído no seu plano de medicação atual, e que tenha o atributo 'Verified' a 0, é adicionada uma entrada deste tipo de notificação na tabela 'Notification', associada a uma instituição.

Aquando da seleção de um dos utentes apresentados na lista de utentes com stock de medicação a terminar, é apresentada a lista de medicamentos a entrar em rutura de stock que tenham o atributo 'Verified' a 0. No caso de o medicamento não ter em stock nenhuma caixa são necessárias encomendar pelo menos duas caixas, se tiver uma caixa ainda em stock é apenas preciso encomendar, pelo menos, uma caixa do respetivo medicamento.

O fluxo de dispensar a notificação, é análogo ao anterior, com a exceção de que o número de caixas de medicação a adicionadar é relativo ao número de caixas existentes.

5.3 FRONTEND

5.3.1 *Estrutura do Frontend*

O *frontend* está dividido em 8 partes diferentes:

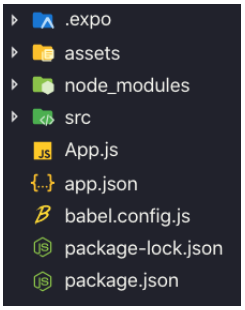


Figura 19: Estrutura do *frontend*.

Como é possível identificar pela imagem acima, tal como no backend existem os ficheiros node_modules, package-lock.json e package.json.

Os dois ficheiros mencionados em primeiro lugar têm um comportamento coincidente com o explicado na estrutura do *backend*.

O ficheiro package.json difere em alguns aspetos, sendo este explicado de seguida, bem como as restantes partes:

- Diretorias .expo e ficheiro babel.config.js
  - Esta diretoria e este ficheiro são gerados automaticamente aquando da inicialização do projeto através da framework Expo.

- Diretoria assets
  - Contém recursos necessários à aplicação. Neste caso, contém as diferentes imagens utilizadas.
- Diretoria src
  - Diretoria com o código relativo aos diferentes ecrãs da aplicação.

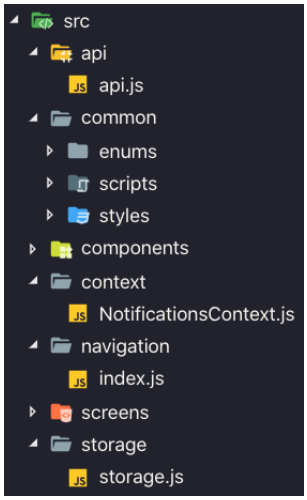


Figura 20: Estrutura da diretoria src.

  * Diretoria api
    · Nesta diretoria encontra-se o ficheiro api.js, o qual contém todos os pedidos HTTP feitos ao *backend*, necessários nos diferentes ecrãs da aplicação.

```
import axios from "axios";

const url = "http://ec2-13-58-68-17.us-east-2.compute.amazonaws.
    com:65124";

const { getData } = require('../storage/storage.js');

const setAxiosHeaders = async () => {
  const token = await getData('token');
  const username = await getData('username');
  axios.defaults.headers.common['Authorization'] = token;
  axios.defaults.headers.common['Username'] = username;
}

//GET
export const getPatients = async (instId) => {
  try {
```

```
        await setAxiosHeaders();

        const params = {
          instId: instId
        };

        const response = await axios.get(`${url}/patients`,
          {
            params
          });

        return response.data;
      } catch (error) {
        const statusCode = error.response ? error.response.status :
            500;

        throw new Error(statusCode.toString());
      }
    };
```

Código 5.5: JavaScript - Exemplo de pedido HTTP - getPatient()

* Diretoria common
  - Contém as diretorias enums, scripts e styles;

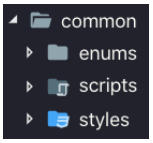


Figura 21: Estrutura da diretoria common.

  - O conteúdo destas diretorias é utilizado em vários locais do desenvolvimento da aplicação;
  - Na diretoria enums encontram-se os diferentes enums utilizados para declarar um conjunto de valores e constantes pré-definidos;
  - Na diretoria scripts estão presentes as funções utilizadas para apresentação do nome dos utentes, normalização de fontes entre diferentes ecrãs, validação de parâmetros e também construção de datas;
  - Na diretoria styles são definidos os estilos gerais da aplicação.

* Diretoria components

        - Todos os componentes, num total de 31, necessários nos diferentes ecrãs da aplicação, encontram-se nesta diretoria divididos em diferentes pastas;
        - Os componentes são genéricos de modo a poderem ser utilizados em mais que um ecrã;
        - Cada componente tem a si associado os respetivos estilos dentro da respetiva pasta.
    * Diretoria context
        - No ficheiro NotificationContext.js desta diretoria é definido o contexto relativo ao número de notificações não lidas, transversal a todos os ecrãs da aplicação.
    * Diretoria navigation
        - Nesta diretoria está presente o ficheiro index.js, no qual são configuradas as navegações entre ecrãs da aplicação, bem como o seu ecrã inicial.
    * Diretoria screens
        - Aqui encontram-se os 22 ecrãs da aplicação;
        - Cada ecrã consome os componentes necessários para apresentação.
    * Diretoria storage
        - Dados persistentes são gravados, acedidos e apagados no ficheiro storage.js desta diretoria;
        - Estes dados são relativos a informação do utilizador, de modo a manter a sessão iniciada mesmo que este saia da aplicação, até que faça *logout*.

- Ficheiro App.js
    - Ponto de entrada da aplicação, no qual é chamado o contexto geral da mesma e a sua configuração de navegação.

```
import React from 'react';
import { NotificationsProvider } from './src/context/NotificationsContext.
    js';
import AppNavigator from './src/navigation';

export default function App() {
  return (
    <NotificationsProvider>
      <AppNavigator />
    </NotificationsProvider>
```

```
    )
  }
```

Código 5.6: JavaScript - Ficheiro App.js

- Ficheiro app.json
  - Esta diretoria é gerada automaticamente aquando da inicialização do projeto através da framework Expo.
  - Contém configurações tais como nome, versão, orientação e ícone da aplicação, plataformas suportadas, entre outras.

5.3.2 *Framework e Principais Packages Utilizados*

Expo

O Expo é um framework utlizado para o desenvolvimento de aplicações móveis utilizando React Native. Este framework contém o Expo SDK (*kit* de desenvolvimento de software) que permite o acesso a funcionalidades do dispositivo onde a aplicação executa, tais como, por exemplo, câmara, contactos e armazenamento local.

Este framework auxilia nos processos de desenvolvimento, *build* e *deploy* da aplicação. Nos processos de desenvolvimento e de build é assim possível testar continuamente, pois é disponibilizada a hipótese de executar a aplicação num simulador no próprio computador, ou então diretamente num dispositivo recorrendo da aplicação Expo. Relativamente ao processo de deploy, o framework Expo gera automaticament os executáveis para os diferentes sistemas operativos, deixando apenas a encargo do desenvolvedor o carregamento do executável para a respetiva loja de aplicações do sistema operativo.[31]

Axios

Para que seja possível ter uma aplicação móvel com informação dinâmica, é utilizado o package axios do npm, que permite fazer pedidos HTTP ao servidor desenvolvido.[32]

React Native Core Components

No desenvolvimento desta aplicação foram utilizados os componentes disponibilizados pelo React Native.[33] Estes componentes são depois transformados no código nativo do sistema operativo onde a aplicação executa, utilizando um sistema de comunicação assíncrono.[34]

6

INTERFACES DA APLICAÇÃO MÓVEL

Neste capítulo são apresentados os principais ecrãs da aplicação, bem como uma breve explicação dos mesmos. Para realizar todas as capturas de ecrã presentes neste capítulo, foi utilizado o emulador do iPhone 11 Pro de 13.7 polegadas.

6.1 LOGIN

Na Figura 22 é apresentado o ecrã inicial da aplicação para utilizadores sem *login* efetuado. No caso de o utilizador introduzir credenciais válidas, este é direcionado para o ecrã de notificações, apresentado na Figura 23, caso contrário é apresentada uma mensagem de erro e o utilizador tem de introduzir novamente as credenciais.

Figura 22: Ecrã de Login.

6.2 NOTIFICAÇÕES

Na Figura 23 é apresentado o ecrã inicial da aplicação para utilizadores com *login* efetuado. Este ecrã contém as notificações que até ao momento não foram dispensadas por nenhum utilizador da instituição. Para que seja possível dispensar as notificações gerais, é necessário que todas as notificações individuais relativas a cada utente estejam dispensadas, como apresentado nas figuras 24, 25 e 26.

Figura 23: Ecrã de Notificações.

6.2.1 *Planeamento de Medicação*

Na Figura 24 está presente o ecrã relativo a um dos utentes presentes na lista da notificação "Planeamento de Medicação". Esta notificação relativa ao planeamento da medicação nos dias definidos pelo lar, apenas pode ser dispensada quando todos os utentes tiverem a sua caixa completa, sendo que para o efeito na aplicação, é necessário clicar sobre o botão "Feito".

Figura 24: Ecrã de Notificação Individual - Planeamento de Medicação.

6.2.2 *Medicação em Fim de Validade*

Na Figura 25 é apresentada a lista da medicação em fim de validade de um utente. Para que seja possível marcar a notificação geral como dispensada, é necessário que seja colocado o visto em todas as linhas da lista de medicação apresentada para cada utente e posteriormente se clique no botão "Dispensar"(este botão apenas é apresentado quando pelo menos uma linha é selecionada).

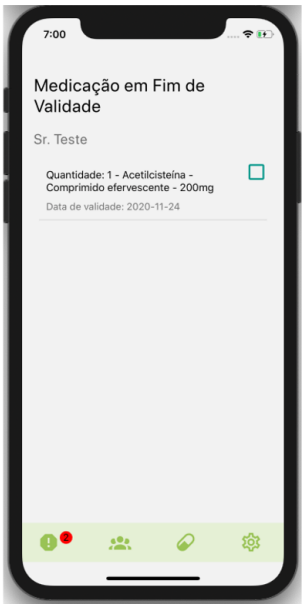


Figura 25: Ecrã de Notificação Individual - Medicação em Fim de Validade.

6.2.3 *Falta de Medicação*

O funcionamento do fluxo da Figura 26 é igual ao que acontece com a notificação explicada relativa à Figura 25. Neste ecrã apenas não aparece a informação relativa à data de validade da medicação.

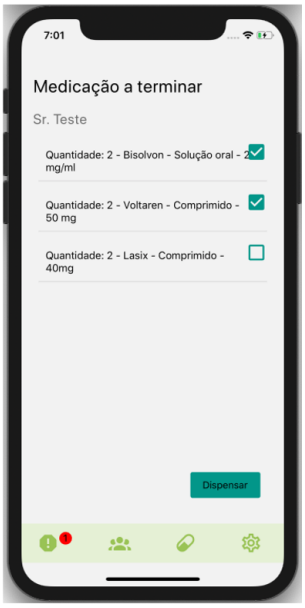


Figura 26: Ecrã de Notificação Individual - Planeamento de Medicação.

6.3 LISTA DE UTENTES

Na Figura 27 é apresentado o ecrã relativo à lista de utentes da instituição. É possível filtrar a lista por nome de utente e estado. É também neste ecrã que temos a possibilidade de adicionar um novo utente à lista clicando no botão "+".

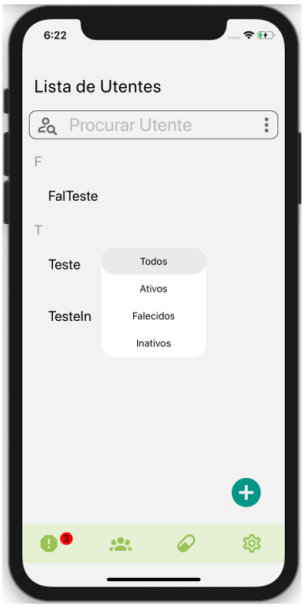


Figura 27: Ecrã de Lista de Utentes.

6.4 FICHA DE IDENTIFICAÇÃO E INFORMAÇÃO DO UTENTE

Na Figura 28 é apresentado o ecrã com toda a informação do utente escolhido através da lista aprsentada no ecrã da Figura 27. Neste ecrã é ainda possível navegar para os ecrãs de "Histórico de Medicação Individual", "Ficha de Medicação Atual"e de "Editar Informação do Utente".

Figura 28: Ecrã de Ficha de Identififcação e Informação do Utente.

6.5 Histórico de Medicação Individual

No ecrã da Figura 29 é apresentada a informação relativa ao histórico de medicação do utente. É possível verificar a medicação crónica do utente e também a temporária. Na secção da medicação temporária, além da informação da data de início de tratamento, é também apresentada a data de fim.

Figura 29: Ecrã de Histórico de Medicação Individual.

6.6 FICHA DE MEDICAÇÃO INDIVIDUAL

No ecrã da Figura 30 é apresentada a lista de medicação atual do utente. Neste ecrã é também possível adicionar nova medicação ao utente clicando no botão "+"e também aceder à informação relativa aos medicamentos presentes na lista, sendo para isso apenas necessário clicar sobre a linha do medicamento a consultar.

Figura 30: Ecrã de Ficha de Medicação Individual.

6.7 FICHA DE MEDICAMENTO INDIVIDUAL

No ecrã da Figura 31 é apresentada a informação relativa ao medicamento selecionado de um utente. É também possível neste ecrã editar a prescrição do medicamento ao utente, clicando no símbolo do lápis. Dar baixa de uma caixa terminada ou aceder ao ecrã de adicionar caixa do respetivo medicamento ao utente é também feito neste ecrã.

Figura 31: Ecrã de Ficha de Medicamento Individual.

6.8 LISTA DE MEDICAMENTOS

Na Figura 32 é apresentado o ecrã relativo à lista de medicamentos já adicionados pelo lar. Esta lista pode ser filtrada pelo nome do medicamento. É também aqui que é possível adicionar novos medicamentos.

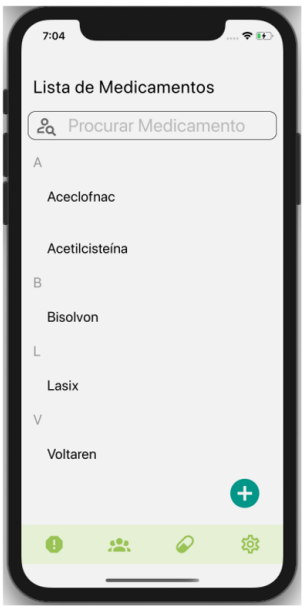


Figura 32: Ecrã de Lista de Medicamentos.

6.9 INFORMAÇÃO DE MEDICAMENTO - GERAL

Na Figura 33 é apresentado o ecrã relativo à informação de um medicamento selecionado da lista de medicamentos. Aqui é possível verificar as diferentes formas farmacêuticas e dosagens do medicamento escolhido, bem como os utentes que tomam esse medicamento nas diferentes variantes (formas famacêuticas e dosagens).

Figura 33: Ecrã de Informação de Medicamento Geral.

6.10 ADMINISTRAÇÃO

Na Figura 34 é apresentado o ecrã relativo à informação da instituição, dias de planeamento da medicação e diferente períodos de administração de medicação. Todas estas informações são editáveis e variáveis entre instituições. É possível repor algum dado editado incorretamente, desde que este ainda não tenha sido salvo. É também neste ecrã que o utilizador pode fazer *logout* da aplicação clicando no símbolo presente no topo à direita.

Figura 34: Ecrã de Administração.

7

PROVA DE CONCEITO

No presente capítulo é apresentada a prova de conceito recorrendo à análise SWOT feita à aplicação móvel desenvolvida.

7.1 ANÁLISE SWOT

A aplicação móvel desenvolvida foi submetida à análise SWOT tendo em conta aspetos de desenvolvimento e também a experiência de utilizadores da área da saúde em lares de idosos. Deste modo foi então possível melhorar alguns pontos da aplicação e ter uma perspetiva do que pode vir a ser melhorado.

Nesta análise, as diferentes letras que constituem o seu nome têm diferentes caraterísticas. De seguida são explicadas de forma sucinta as respetivas caraterísticas:

- **S**: Esta letra corresponde às Forças (*strengths*), sendo esta letra relativa a fatores internos e positivos, analisando vantagens internas da aplicação móvel relativas a concorrentes;
- **W**: *Weaknesses*, que em português significa Fraquezas, tem a ver também com fatores internos, mas neste caso negativos. Em oposição às forças, esta letra tem em conta as desvantagens internas da aplicação móvel relativas a concorrentes;
- **O**: O de Oportunidades (*Opportunities*), prevê aspetos externos que possam influenciar positivamente a aplicação móvel;
- **T**: As Ameaças, *Threats*, são relativas a fatores externos e negativos relativos à aplicação móvel.

	Fatores Positivos	Fatores Negativos
Fatores Internos	Strengths (Forças)	Weaknesses (Fraquezas)
Fatores Externos	Opportunities (Oportunidades)	Threats (Ameaças)

Figura 35: Análise SWOT.

De seguida são apresentadas as forças, fraquezas, oportunidades e ameaças, identificadas relativas à aplicação móvel:

- Forças:
  - Escalabilidade: A arquitetura do desenvolvimento da aplicação móvel foi elaborada de forma a permitir escalar horizontalmente, permitindo maior flexibilidade dos processos;
  - Centralização: Diversas funcionalidades integradas numa só aplicação, tais como ficha de utente, medicação e informações da instituição;
  - Usabilidade: Aplicação *user-friendly*, o que permite evitar problemas futuros relativamente ao desenvolvimento, aumento da produtividade do programador e do utilizador da mesma e também um menores custos relativamente a desenvolvimento e apoio ao cliente.
- Fraquezas:
  - Conectividade: É necessário ter uma ligação de dados móveis ou *wireless* ativa de modo a usufruir da aplicação;
  - Versão do Dispositivo: O dispositivo onde corre a aplicação tem de ter versão 5 ou superior no caso de software Android e versão 10 ou superior se iOS.
- Oportunidades:
  - Minimização de erros: Com recurso à aplicação é possível diminuir o erro dos profissionais de saúde;
  - Produtividade: Acesso simples e rápido a todas as informações relativas aos utentes.
- Ameaças:
  - Novas tecnologias: Dificuldade em aderir a novas tecnologias por parte dos profissionais de saúde;

- Concorrência: Novos concorrentes;
- Conectividade: Fraca ligação de dados móveis ou *wireless*.

8

CONCLUSÃO E TRABALHO FUTURO

Neste capítulo é apresentada a conclusão de todo o trabalho elaborado. Num primeiro momento são referidas as principais contribuições e de seguida são abordados temas para trabalho futuro.

8.1 PRINCIPAIS CONTRIBUIÇÕES

Com esta dissertação foi desenvolvida uma aplicação móvel que visa agilizar o processo de planeamento e administração da medicação em lares de idosos. Através da aplicação desenvolvida é possível também diminuir o erro humano, ter a todo o instante acesso a informação relativa ao stock de medicação de cada utente, bem como do seu plano atual e histórico de medicação. Também através da aplicação é possível fornecer informação relativa a dados do utente e da instituição aquando do acompanhamento do utente ao médico.

Todo o trabalho desenvolvido torna mais fácil o trabalho dos auxiliares de saúde dos lares de idosos e previne a perda de informação.

Na secção 1.3 do capítulo 1 foram levantadas algumas questões de investigação, que podem agora ser respondidas relacionando as mesmas com a aplicação móvel desenvolvida.

- De que forma pode esta aplicação auxiliar no planeamento da medicação semanal num lar de idosos?
  - A aplicação móvel disponibiliza notificações nos dias definidos de planeamento de medicação, sendo assim possível visualizar os utentes e a respetiva medicação;
  - A aplicação móvel disponibiliza notificações quando o *stock* de medicação está a terminar ou a validade está a chegar ao fim, impedindo assim que os utentes fiquem sem medicação.
- É possível o processo de administração da medicação ser agilizado?
  - Através da aplicação móvel é possível consultar notas relativas aos utentes, sendo assim possível antever impasses na hora da administração da medicação.

- De que forma esta aplicação pode auxiliar os profissionais de saúde no momento de acompanhar os utentes ao médico?
  - A aplicação móvel disponibiliza o histórico da medicação dos utentes;
  - A aplicação disponibiliza a medicação atual dos utentes;
  - A aplicação disponibiliza dados pessoais, tais como, número do cartão de cidadão, número de utente do serviço nacional de saúde, data de nascimento, antecedentes, entre outros.

Por último, apesar de a aplicação ter por base o levantamento de requisitos relativos a uma instituição, a mesma foi concebida de modo a que seja possível aplicar em qualquer lar do país.

8.2 TRABALHO FUTURO

Todo o trabalho inicialmente proposto foi desenvolvido, no entanto existem pontos de melhoria para trabalho futuro, tais como:

- Uso de HTTPS de modo a garantir uma camada adicional de segurança;
- Aplicar *Business Inteligence*;
- Permitir envio de informação relativa aos utentes através de e-mail;
- Melhorar a responsividade de alguns componentes de modo a proporcionar uma boa experiência de utilização em qualquer tamanho de ecrã.

BIBLIOGRAFIA

[1] S. Chatterjee K. Peffersm T. Tuunanen, M. Rothenberger. A design science research methodology for information systems research. *Journal of Management Information Systems*, 24:54, 2007.

[2] PORDATA INE. Esperança de vida à nascença: total e por sexo. https://www.pordata.pt/Portugal/Esperanca+de+vida+a+nascenca+total+e+por+sexo+(base+trienio+a+partir+de+2001)-418, 2020. Acedido: 2020-07-17.

[3] PORDATA INE. Indicadores de envelhecimento. https://www.pordata.pt/Portugal/Indicadores+de+envelhecimento-526, 2020. Acedido: 2020-07-17.

[4] Ana Cristina Pereira. Lares cheios. https://www.publico.pt/2018/12/15/sociedade/noticia/centros-dia-atraem-menos-lares-estao-cheios-1854827, 2018. Acedido: 2020-10-26.

[5] LUSA/SO. Instituições de idosos em portugal têm falta de profissionais. https://saudeonline.pt/2018/03/23/instituicoes-de-idosos-em-portugal-tem-falta-de-profissionais/, 2018. Acedido: 2020-10-26.

[6] C. Lee Ventola. Mobile devices and apps for health care professionals: Uses and benefits. *Pharmacology and Therapeutics*, 39:356–364, 2014.

[7] Oedekoven M OŚullivan JL Kanzler M Kuhlmey A Gellert P Ernsting C, Dombrowski SU. Using smartphones and health apps to change and manage health behaviors: A population-based survey. *Journal of Medical Internet Research*, 19, 2017.

[8] Top 5 ferramentas de bi. https://towardsdatascience.com/top-5-bi-tools-that-you-must-use-for-data-visualization-7ccc2a852bd3, 2019. Acedido: 2020-08-19.

[9] Especificações - ferramenta tableau. https://www.tableau.com/products/techspecs, 2020. Acedido: 2020-10-18.

[10] Especificações - ferramenta power bi. https://docs.microsoft.com/pt-pt/power-bi/fundamentals/desktop-get-the-desktop, 2020. Acedido: 2020-10-18.

[11] V. Vimarlund T. Mettler. Understanding business intelligence in the context of healthcare. *Health Informatics Journal*, 15:254–264, 2009.

[12] C. Veda C. Hsinchun, H. Roger. Business intelligence and analytics: From big data to big impact. *MIS Quarterly*, 36:1293–1327, 2018.

[13] E. Caritá E. Morais, S. Silva. Business intelligence utilizando tecnologias web para análise de fatores de risco na ocorrência de doença arterial coronariana business intelligence using web technology for analysis of risk factors for coronary artery disease. *JHI Journal of Health Informatics*, 2:7–13, 2010.

[14] X. Dong R. Davison, S. Sia. Design science in the information systems discipline: An introduction to the special issue on design science research. *Information Systems Journal*, 18:325–330, 2008.

[15] M. Rothenberger A. Gupta V. Yoon M. Tremblay, D. VanderMeer. *Advancing the Impact of Design Science: Moving from Theory to Practice*, chapter Towards a Design Science-Driven Product-Service System Engineering Methodology. 2014.

[16] A. Sergey A. Sergey, D. Alexandr. Proof of concept center — a promising tool for innovative development at entrepreneurial universities. *Procedia - Social and Behavioral Sciences*, 166:240–245, 2015.

[17] E. Valentin. Swot analysis from a resource-based view. *Journal of Marketing Theory and Practice*, 9:54–69, 2001.

[18] T. Hussain T. Crawford. A comparison of server side scripting technologies. *International Conference Software Engineering Research and Practice*, pages 69–76, 2017.

[19] Most popular technologies. https://insights.stackoverflow.com/survey/2020, 2020. Acedido: 2020-12-05.

[20] Utilização de sistemas operativos móveis em todo o mundo. https://gs.statcounter.com/os-market-share/mobile/worldwide, 2020. Acedido: 2020-12-01.

[21] Top de frameworks para desenvolvimento de aplicações móveis híbridas em 2020. https://www.websoptimization.com/blog/hybrid-mobile-app-frameworks/, 2020. Acedido: 2020-12-06.

[22] Base de dados relacionais. https://www.oracle.com/database/what-is-a-relational-database/, 2020. Acedido: 2020-11-16.

[23] Definição de requisito. https://www.infopedia.pt/dicionarios/lingua-portuguesa/requisito, 2020. Acedido: 2020-08-22.

[24] R. Machado J. Fernandes. *Lecture Notes in Management and Industrial Engineering: Requirements in Engineering Projects*, chapter Requirements. 2015.

[25] M. Qasem A. Alzaqebah A. Hudaib, R. Masadeh. Requirements prioritization techniques comparison. *Modern Applied Science*, 12:64–65, 2018.

[26] Requirement shell - volere. https://www.reqview.com/blog/2019-02-27-news-volere-requirements-specification-template, 2019. Acedido: 2020-08-16.

[27] Sequelize. https://sequelize.org, 2020. Acedido: 2020-10-15.

[28] Express. https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction, 2020. Acedido: 2020-10-15.

[29] Package jsonwebtoken. https://www.npmjs.com/package/jsonwebtoken, 2018. Acedido: 2020-10-15.

[30] Package node-cron. https://www.npmjs.com/package/cron, 2020. Acedido: 2020-10-15.

[31] Package axios. https://www.npmjs.com/package/axios, 2020. Acedido: 2020-11-02.

[32] Framewrok expo. https://docs.expo.io/, 2020. Acedido: 2020-11-02.

[33] React native core components. https://reactnative.dev/docs/intro-react-native-components, 2020. Acedido: 2020-11-02.

[34] Comunicação entre react native e sistema operativo. https://dev.to/mfrachet/understanding-the-react-native-bridge-concept-1k90, 2019. Acedido: 2020-11-02.