

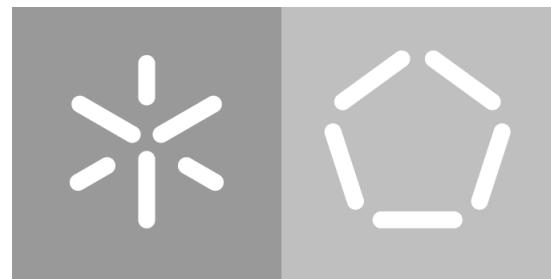
Universidade do Minho
Escola de Engenharia
Departamento de Informática

A67654 - André Soares
A83819 - Miguel Oliveira
A84727 - Nelson Faria
A85501 - José Rodrigues

LI4 - Projeto

Consulta Já

Julho 2020



Universidade do Minho
Escola de Engenharia
Departamento de Informática

A67654 - André Soares
A83819 - Miguel Oliveira
A84727 - Nelson Faria
A85501 - José Rodrigues

LI4 - Projeto Consulta Já

Laboratórios de Informática IV
Mestrado Integrado em Engenharia Informática

Projeto supervisionado por
Prof.^a Daniela Oliveira

Julho 2020

AGRADECIMENTOS

Gostaríamos de agradecer à professora Daniela Oliveira pelo apoio que nos demonstrou ao longo de todos os *sprints* para consecutivamente irmos melhorando aquilo que íamos fazendo ao longo do processo de desenvolvimento do nosso projeto.

RESUMO

Ao longo deste documento iremos descrever o contexto e o tema no qual se integra a aplicação desenvolvida pelo nosso grupo no âmbito desta unidade curricular, nomeadamente no que toca à existência de outras propostas com objetivos divergentes aos nossos e também todas as suas etapas e componentes de implementação, nas quais se incluem o **back-end**, componente que se comporta como servidor da aplicação, armazenando os dados necessários e as respetivas rotinas que garantem a funcionalidade básica da plataforma, assim como a capacidade de receber pedidos por parte de clientes e processá-los, e ainda o **front-end**, componente que representa o cliente (ou utilizador) da plataforma, cuja principal função passa por disponibilizar para o cliente essas funcionalidades da aplicação através de uma interface *user-friendly*, que permita ao cliente usufruir das capacidades da aplicação de uma forma simples e intuitiva.

CONTEÚDO

1	INTRODUÇÃO	1
2	ESTADO DA ARTE	2
3	O PROBLEMA E SEUS DESAFIOS	4
3.1	Arquitetura do Sistema	5
4	DESENVOLVIMENTO	7
4.1	Decisões	7
4.2	Implementação	7
4.2.1	Back-end	7
4.2.2	Arquitetura da solução	7
4.2.3	Desenho da base de dados	10
4.2.4	Controller	11
4.2.5	Front-end	11
4.3	Resultados	13
5	CASOS DE ESTUDO/EXPERIÊNCIA	22
6	CONCLUSÃO	24
6.1	Conclusões	24
6.2	Perspetiva para trabalho futuro	24

LISTA DE FIGURAS

3.1 Arquitetura global da aplicação	5
3.2 Salt e Hash de uma Password	6
4.1 Diagrama de Classes	8
4.2 Modelo Lógico da Base de Dados	10
4.3 Página home (primeira parte)	13
4.4 Página home (segunda parte)	13
4.5 Página registar	14
4.6 Página login	14
4.7 Página perfil do administrador	15
4.8 Página perfil do paciente	15
4.9 Página propostas de consulta de um paciente, aceites pelo médico e que aguardam aprovação final do paciente	16
4.10 Página marcar consulta por parte do paciente	16
4.11 Página histórico do paciente, onde são apresentadas as consultas realizadas pelo paciente	17
4.12 Página histórico do paciente, após selecionar uma consulta, em que aparecem os botões para o download da receita e do recibo relativos à consulta selecionada	17
4.13 Página editar perfil, que é igual tanto no paciente como no médico	18
4.14 Página logout, apresentada após o término da sessão por parte do utilizador	18
4.15 Página perfil do médico	19
4.16 Página propostas de consulta de um médico, feitas por um paciente, onde o médico tem a possibilidade de aceitar a mesma	19
4.17 Página pós consulta de um médico, onde o médico pode marcar um consulta como efetuada, passando para a tela abaixo e passar a devida receita, com os respetivos medicamentos, assim como uma pequena observação da consulta	20
4.18 Página pós consulta de um médico, onde o médico pode passar a devida receita, com os respetivos medicamentos, assim como uma pequena observação da consulta	20
4.19 Página histórico de consultas do médico, onde este pode ver as consultas que já foram realizadas por si	21
4.20 Página privacidade da aplicação	21

ABREVIATURAS

A

API Application Programming Interface

B

BD Base de Dados

C

CSS Cascading Style Sheet

D

DAO Data Access Object

H

HTTP HyperText Transfer Protocol

HTML HyperText Markup Language

N

NoSQL Not only Structured Query Language

R

REST Representational State Transfer

S

SQL Structured Query Language

SNS Serviço Nacional de Saúde

U

UC Unidade Curricular

1 INTRODUÇÃO

No âmbito da unidade curricular de **Laboratórios de Informática IV**, foi-nos solicitado o desenvolvimento de uma aplicação de modo a dar resposta a um qualquer problema delimitado e adotado pelo grupo, tendo a solução que ser capaz de garantir algumas características, nomeadamente apresentar uma interface web responsive ou mobile, fazer uso de uma base de dados (SQL ou NoSQL) para manipulação e acesso a dados, em situações que assim o exijam no contexto do nosso problema, recorrer a API e WebServices e ainda ter o seu desenvolvimento Backend concretizado recorrendo a ferramentas Microsoft, tendo sido, no nosso caso em particular, o C#.

Na secção seguinte passamos a apresentar o tema adotado para a elaboração do projeto de avaliação para a respetiva UC.

2 ESTADO DA ARTE

A ideia para a conceção da aplicação final do grupo passou pela criação de uma plataforma que permitisse agilizar toda uma gestão dos procedimentos necessários para levar a cabo a realização de **consultas ao domicílio**, em particular toda a burocracia relacionada com a solicitação de consultas por parte de **pacientes** e a sua aceitação por parte de **médicos**, entre outros.

De uma forma simples, todos acharíamos bastante interessante e até mesmo cómodo termos acesso a uma aplicação que nos permitisse marcar consultas em casa, não necessitando de nos deslocar-mos para uma clínica ou algo do género e tendo ainda acesso aos mesmos recursos, sejam elas receitas médicas pós-consulta (em formato pdf), fatura, etc. Até mesmo para os médicos seria bastante interessante ter um "emprego" extra, sem comprometer o seu posto de trabalho, permitindo faturar uma quantia extra. Para além disto, seria benéfico essa plataforma ser de **fácil inscrição** por parte quer de pacientes quer de médicos. Obviamente não poderíamos deixar de ter em conta que o registo como médico necessitaria de ser muito mais rígido e por isso, um pedido de inscrição como médico deveria ser supervisionado pelo administrador da aplicação. Foi a pensar nisto que propusemos este tema.

Apesar de a área de atuação do nosso projeto ser a área da saúde, área essa que engloba muitos serviços e especialidades, o que pretendemos é um foco na ajuda na marcação online de consultas ao domicílio de Medicina geral e familiar/Clínica Geral e as habitualmente chamadas consultas de rotina.

Uma importante parte do mercado é identificar e avaliar os nossos concorrentes. Descobrir os fortes e fraquezas da concorrência posiciona-nos em vantagem para apresentar um serviço melhor e diferente do que está já a ser feito o que vai acabar por nos destacar. Como o que nos interessa é disponibilizar facilidade em marcação e realização de consultas ao domicílio é nisso que nos vamos focar na pesquisa.

Um breve resumo de potenciais concorrentes:

- **SNS**

O serviço nacional de saúde é muito limitado em termos de consultas ao domicílio. Compreensivelmente os médicos das unidades de saúde pública não podem andar a viajar de um lado para o outro durante o dia e ao mesmo tempo dar resposta às longas filas de espera dos centros de saúde. Alguns centros hospitalares permitem a ida de médicos ao domicílio, mas apenas em casos muito especiais em que o doente não se pode deslocar pelos próprios meios.

- **Geridoc**

À primeira vista, a geridoc oferece um serviço muito parecido com o que nós queremos fazer: consultas ao domicílio. A geridoc é, no entanto, uma clínica privada física com foco na geriatria e com a sua própria equipa de médicos e enfermeiros que se disponibilizam a ir a casa, isto difere com o nosso trabalho que pretende que se inscrevam médicos de todo o país de forma independente. Um médico da geridoc trabalha na clínica deles a full time e obviamente isso trás um enorme entrave no raio de ação da clínica que apenas atua no

Grande Porto e em Lisboa.

- **Anjos da noite**

Serviço mais parecido com o nosso, os anjos da noite são uma empresa já fundada e em trabalho há muitos anos e com boa representação no mercado de trabalho, no entanto, O *ConsultaJa* destaca-se não só pela comodidade que vai oferecer aos doentes, mas também aos médicos que se poderão inscrever mesmo tendo um trabalho num outro hospital e fazer horas extras ao realizar consultar pelo nosso website. No Anjos da Noite não existe qualquer menção de como é recrutamento dos profissionais de saúde o que leva a crer que funciona como se de uma clínica privada se tratasse. Mesmo assim é este o principal concorrente, com o serviço mais idêntico.

Terminado esta análise, conclui-se que esta aplicação web é viável e será uma opção de entrada num mercado estável e de crescimento contínuo, mas ainda assim não muito explorado no mundo online.

3 O PROBLEMA E SEUS DESAFIOS

Tendo discutido já previamente todos os aspectos relacionados com a inserção da nossa aplicação no mercado e também após a análise de trabalhos e propostas semelhantes à nossa, vamos agora discutir um pouco o problema em si, incluindo os seus principais objetivos e desafios.

Como não poderia deixar de ser, foi necessário criar um model na linguagem de programação adotada de acordo com o tema adotado, fornecendo os procedimentos que, de forma implícita, representam as funcionalidades básicas que nos comprometemos inicialmente a incluir na nossa aplicação. Para tal necessitaremos de garantir armazenamento. De forma mais ou menos óbvia para qualquer um de nós, não faria sentido não o fazer, como na maior parte das aplicações que circulam nos dias de hoje na internet, visto que há necessidade de guardar informação, por exemplo, os dados pessoais de uma conta aquando do registo de um usuário na aplicação, a data e a hora de uma consulta, o preço por cada consulta, etc. A solução mais utilizada nos dias de hoje é aquela pela qual optamos foi o uso de um **sistema de base de dados**.

De seguida e com a finalidade de que a nossa aplicação apresentasse características web responsive, foi necessário, após essa implementação, recorrer à conceção de uma API REST através da construção de algumas classes responsáveis por essa ligação entre o **front-end** (cliente) e o **back-end** (servidor) designadas por **Controller**. De uma forma simples, estas classes recebem os *http requests* provenientes de clientes e tratam-nos usando para tal o model concebido em C# no contexto da nossa aplicação e explicado no parágrafo anterior, enviando para ele os resultados das operações solicitadas através de *http responses*.

Para garantir uma boa experiência de utilização do web-site quer por parte de médicos, quer por parte de pacientes, o grupo preocupou-se com o desenvolvimento de uma interface simples, atrativa e organizada (user-friendly), sendo a partir de interações com esta componente por parte de utilizadores que resultam em envios de pedidos http que por sua vez serão ”apanhados” pelo **controller** como já foi referido anteriormente.

Outros dos aspectos que se torna fundamental estar presente na nossa aplicação é o conceito de **segurança**. Cada vez se torna mais importante, nos dias que correm, garantir privacidade e confidencialidade ao aceder a um serviço web, prevenindo todo e qualquer tipo de ataques de espionagem ou até de interceção de informação trocada no processo de comunicação entre o servidor e o cliente final.

Para tal consideramos relevante a cifragem de alguns conteúdos, nomeadamente a **cifragem de passwords** e ainda a implementação de um sistema de atribuição de tokens aquando da autenticação na aplicação através de um pedido de login.

3.1 Arquitetura do Sistema

Em resposta às necessidades que fomos salientando na secção anterior, implementamos um sistema com a arquitetura ilustrada na figura seguinte. A imagem vai exatamente ao encontro de tudo o que foi descrito nessa mesma secção.

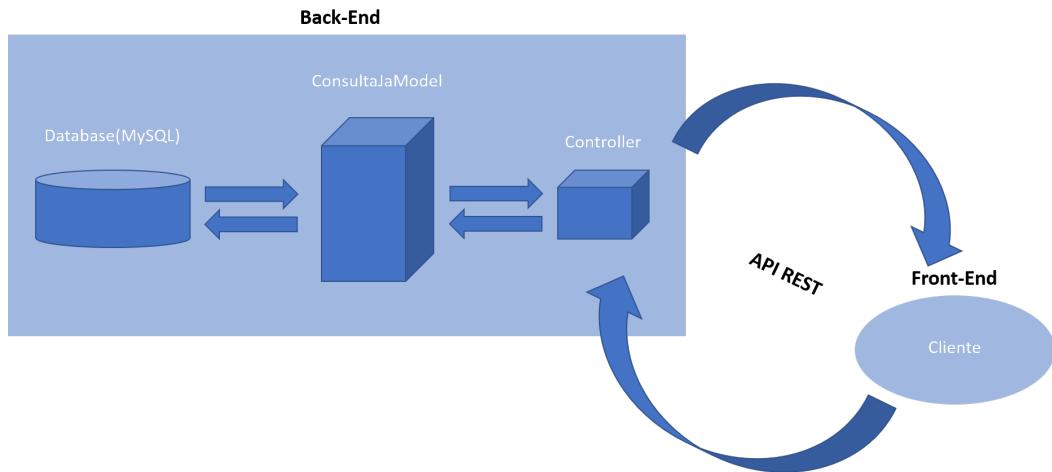


Figura 3.1: Arquitetura global da aplicação

Em jeito de síntese, o nosso produto final conta com uma componente de dados (no nosso caso optamos por um sistema MySQL), que armazena toda a informação necessária para o normal funcionamento da aplicação, uma componente que acede a esses dados de modo a dar resposta a pedidos mediante invocações dos métodos que são sua parte constituinte, a qual designamos de **Model**, uma outra componente responsável por receber pedidos de clientes (através do protocolo de aplicação designado de **HTTP**) e invoca o método respetivo do model, denominada **Controllers** e finalmente, a componente que permite ao *cliente* enviar **http requests** de modo a poder manipular dados que pretenda de acordo com a(s) funcionalidade(s) à(s) qual(ais) pretende aceder.

No que toca às questões de segurança mencionadas anteriormente, para a cifragem de passwords recorremos ao namespace System.Security.Cryptography disponibilizado pela microsoft que nos oferece serviços criptográficos de entre os quais o hashing de passwords. O objetivo pretendido é que apesar das palavras-pases dos utilizadores serem guardadas na base de dados do ConsultaJa, os gestores da BD nunca as possam ler, tendo apenas acesso à hash gerada aquando da criação da conta. Tal como referido, foi implementado um módulo responsável por estas funções de hash e verificação; Ao utilizar "salted password hashing" garantimos que a mesma palavra-passe tem diferentes hashes, isto é obtido através da utilização de um salt, que se pode resumir a adicionar à password original uma sequência de caracteres aleatórios:

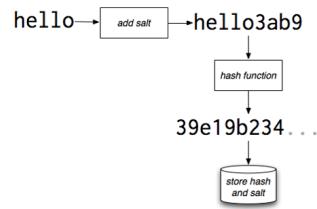


Figura 3.2: Salt e Hash de uma Password

Após a ser adicionado o Salt, é feito o hash da password, e esta é inserida na base de dados. Quando o utilizador se autentica, é feito o hash da password introduzida e este é comparado com o hash inserido aquando da criação da conta.

Quanto ao mecanismo de **autenticação** o procedimento é bastante intuitivo. Aquando do pedido de login, o utilizador fornece o seu email e a sua password. No caso de a password (que se encontra cifrada na base de dados) corresponder, é gerado um **token** que o utilizador irá guardar (de forma implícita, claro) na cache do seu **browser**. Daí em diante, sempre que uma determinado usuário pretenda aceder a uma funcionalidade da **API** que requeira *authorization*, este terá que enviar na sua mensagem *http request* o token gerado no back-end para o devido efeito. Com este token é possível ainda aceder a alguns dados do utilizador que se encontram devidamente cifrados e desta forma é possível realizar os pedidos http de forma mais segura, pois quando for necessário fazer o pedido e esse pedido necessita do id do utilizador, por exemplo, ele já se encontra no token, pelo que só fica a ser necessário descodificar o token e buscar o campo respetivo. No momento em que o utilizador der *logout* o token será descartado e deixará de poder aceder a essas funcionalidades que requeiram autorização a não ser que se volte a autenticar novamente (voltando a guardar um novo token).

4 DESENVOLVIMENTO

4.1 Decisões

Durante a elaboração do projeto houve algumas decisões fundamentais a serem tomadas, nomeadamente começando pela escolha das tecnologias para o desenvolver.

Assim sendo, o grupo optou, para a construção da componente backoffice da aplicação final, uma tecnologia *Microsoft*, mais concretamente e no nosso caso em particular, o **C#**. De modo a garantir o armazenamento (inevitável diga-se de passagem) de dados por parte da aplicação optamos pelo uso de uma **base de dados SQL**, mais concretamente **MySQL**, isto porque acreditamos que esta aplicação nunca apresentará um volume de dados tal que coloque em causa questões de eficiência ao ponto de ser aceitável (ou até mesmo conveniente) sacrificar a consistência dos dados armazenados num sistema MySQL, migrando para NoSQL, um paradigma de sistemas de bases de dados que procura mitigar essa ineficiência perante contextos que impliquem um número vastíssimo de dados.

4.2 Implementação

4.2.1 Back-end

Nesta secção iremos falar um pouco da vertente back-end da nossa aplicação, isto é, na arquitetura da nossa solução e ainda a base de dados que armazena todos os dados necessários para o uso das funcionalidades da aplicação.

Tendo definido as tecnologias a serem utilizadas para elaboração do modelo da aplicação e, de modo a conseguir de facto implementá-lo, seria inevitável, logo à partida, criar a "conexão" entre estas duas componentes, a parte dos **dados** e a parte dos **algorítmos ou procedimentos**. Nunca faria sentido começar a pensar no desenho dos algorítmos necessários para o correto funcionamento da aplicação sem primeiro refletir sobre que método iríamos utilizar para conseguir aceder a uma base de dados MySQL a partir da linguagem adotada para o desenvolvimento do back-end. Para tal usamos classes **DAO's - Data Access Object** - classes que implementam métodos que recorrem à linguagem SQL para aceder/manipular os dados guardados, de modo a dar a garantir a funcionalidade para a qual o método foi desenhado.

Antes de começar qualquer codificação, o grupo preocupou-se em pensar na solução para o problema ao qual se comprometeu a resolver, recorrendo à modelação quer da arquitetura final da solução *diagrama de classes*, quer das funcionalidades básicas para a aplicação *diagrama de Use Cases*.

De seguida, abordaremos cada uma das componentes (dados e procedimentos) de uma forma mais aprofundada.

4.2.2 Arquitetura da solução

Na figura que se segue ilustra-se a arquitetura do produto final de acordo com o tema com o qual nos comprometemos a resolver. É de salientar que ao longo do desenvolvimento do nosso projeto, por vezes, em certas ocasiões, consideramos necessário alterar um pouco a sua arquitetura, nomeadamente aquando da decisão de incluir a possibilidade de associar receitas às consultas após a sua realização.

O modelo foi elaborado com recurso à ferramenta *Visual Paradigm*, mais concretamente, a um **diagrama de classes**.

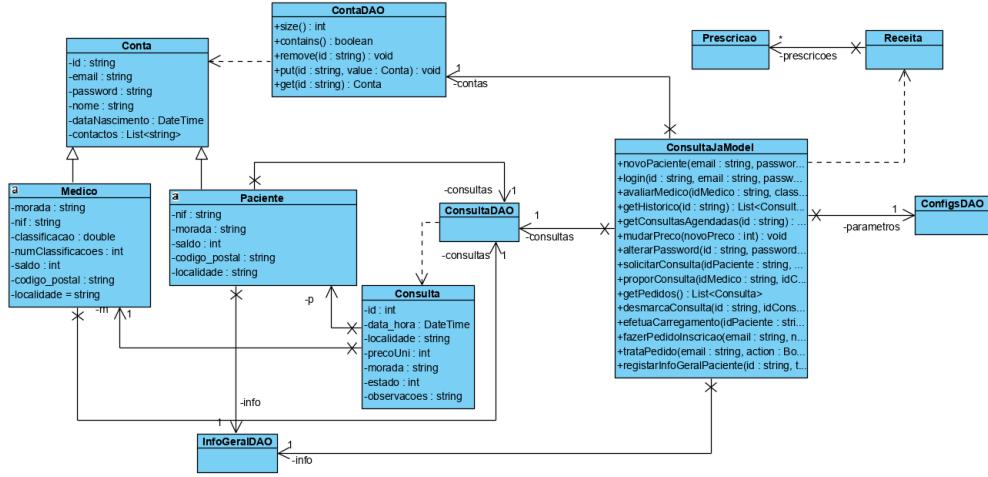


Figura 4.1: Diagrama de Classes

De um modo geral, temos quatro classes que servem de conexão à base de dados, sendo elas designadas por ContaDAO, ConsultaDAO, ConfigsDAO e ainda InfoGeneralDAO. Estas classes permitem não só se comportarem como classes de acesso aos dados armazenados na base de dados, permitindo ainda a sua manipulação tendo em conta o contexto e a natureza destes. De uma forma simplicista, estas classes comportam-se como se fossem coleções de objetos, permitindo, por exemplo, no contexto da plataforma desenvolvida por nós, "armazenar" (não podemos levar a palavra no seu sentido literal, visto que os dados não ficam guardados em classe nenhuma, mas sim na base de dados) todas as contas de utilizadores que se encontrem registados tal como alguns dos seus dados pessoais, assim como toda a informação acerca das consultas guardadas na base de dados, entre outras informações necessárias para a aplicação.

É de notar que, para cada uma destas classes, existe um único objeto de cada que é parte integrante da principal classe da nossa aplicação, sendo ela denominada de **ConsultaJaModel**. É nesta classe que conseguimos garantir os *Use Cases* definidos à partida para o nosso projeto. Por outras palavras, cada método pertencente a esta classe representa uma funcionalidade básica da aplicação, nomeadamente, registar um paciente, fazer um pedido de consulta, alterar o preço unitário da consulta, entre alguns outros.

É relevante informar que o ContaDAO lida com todas as questões relacionadas com a criação e gestão de contas na aplicação e, de modo similar, o ConsultaDAO apresenta as mesmas funções, mas lidando com consultas, cuidando do agendamento, acesso, etc. Já o ConfigsDAO permite aceder e manipular a alguns dados fulcrais para o contexto da aplicação, entre eles, o preço de cada consulta, e o número de médicos e pacientes registados na aplicação ou que manifestaram esse interesse, ao passo que o InfoGeneralDAO fica responsável pelas informações acerca dos pacientes, permitindo associar e aceder a informações de qualquer tipo relacionada com um qualquer paciente.

Por exemplo, num objeto da classe ConsultaDAO, para além de conseguir "construir" um objeto da classe Consulta, conseguimos, dado um id de uma consulta, alterar o seu estado (AGENDADA, REALIZADA, PEDIDO, etc), não sendo necessário para tal, aceder à base de dados para "criar" o objeto, alterá-lo e, no fim, voltar a escrevê-lo na base

de dados. Implementamos desta forma tendo em mente questões de eficiencia, visto que, por ser um serviço web, seria utilizado simultaneamente por várias pessoas.

Na figura seguinte mostramos o exemplo de um método que acede à base de dados, mais concretamente, o método *contains(string id)* da classe *ContaDAO* que dado um id de conta nos diz se existe um objeto da classe Conta com esse mesmo id:

```
public bool contains(string id)
{
    MySqlConnection connection =
        new MySqlConnection(this.connectionstring);

    connection.Open();
    DataTable dt = new DataTable();

    StringBuilder sb = new StringBuilder();

    sb.Append("select * from Conta where idConta='");
    sb.Append(id);
    sb.Append("'");

    MySqlDataAdapter msda =
        new MySqlDataAdapter(sb.ToString(), connection);

    msda.Fill(dt);

    connection.Close();
    return dt.Rows.Count != 0;
}
```

Outras classes das quais fazemos uso são a classe **Conta** que é superclasse de **Medico** e **Paciente**, as quais representam uma conta registada na aplicação e são constituídas pelas informações requeridas pela aplicação aquando do registo de um novo utilizador na plataforma, a classe **Consulta**, a qual representa uma consulta, quer seja ela apenas um pedido de consulta, uma consulta agendada ou até mesmo realizada, e a classe **Receita** que representa uma receita médica a qual, por sua vez contém uma coleção de objetos da classe **Prescricao**, a qual representa a prescrição de uma fármaco por parte de um médico. No diagrama de classes representado acima podemos verificar a constituição de cada uma das classes.

Estas classes são aquelas que são construídas mediante o uso de procedimentos de objetos de classes DAO que acedem à base de dados, de modo a conseguir aceder aos dados armazenados da forma mais organizada, simples e eficiente possível. No entanto, como já foi referido, em situações que não seja necessário ou até mesmo recomendado carregar o objeto para memória, por exemplo, aquando da alteração da palavra passe de uma conta, tal não acontece. Seria ineficiente carregar todo o objeto conta para memória quando apenas pretendemos alterar um campo da sua estrutura. Ao invés disso, possuimos métodos específicos na respetiva classe de acesso à base de dados (DAO) para o efeito, identificando o objeto alvo da alteração com base no seu id único.

Outro caso ilustrativo da situação anterior é aquando da alteração do estado de uma consulta. Quando um consulta passa de agendada para realizada, seria bastante ineficiente

ir buscar todo o objeto quando o nosso objetivo é simplesmente alterar um valor inteiro que identifica o estado da mesma.

4.2.3 Desenho da base de dados

Agora vamos ilustrar e explicar o modo de implementação da base de dados que serve de suporte ao armazenamento dos dados à nossa aplicação. Na figura seguinte encontra-se ilustrado o modelo lógico para a construção do modelo físico da nossa base de dados:

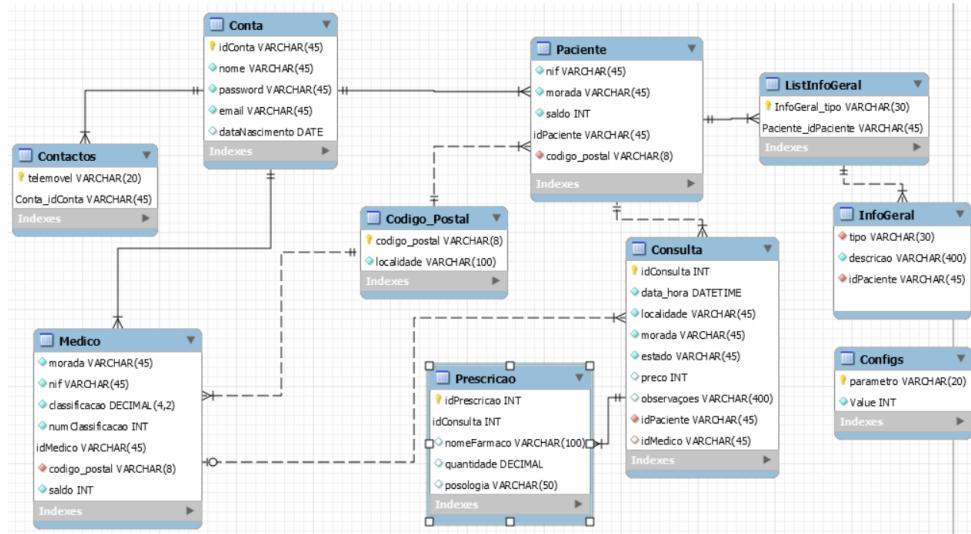


Figura 4.2: Modelo Lógico da Base de Dados

De um modo geral, a base de dados anda muito à volta de 5 tabelas principais, sendo elas a tabela **Conta**, **Medico**, **Paciente** e **Consulta** e **Configs**. Cada linha da tabela Conta está referenciada a uma linha da tabela Medico ou a da tabela Paciente, visto que uma conta ou pertencerá a um médico ou a um paciente. Estas tabelas contêm a informação associada a uma conta (ver na figura acima).

Já a tabela Consulta armazena a informação acerca de todas as consultas, quer tenham sido confirmadas ou não pelo médico e pelo paciente envolvidos, quer não, ou até mesmo se já tiverem sido efetuadas ou se ainda estiverem apenas agendadas. Esta tabela encontra-se também associada a uma linha da tabela Paciente e, simultaneamente, a uma linha da tabela Medico, sendo referenciada pelos respetivos id's de cada um. É no entanto de salientar que, caso a consulta ainda esteja à espera de aceitação por parte de um médico, não estará ainda associado um id de médico a essa consulta.

A tabela Configs guarda algumas informações imprescindíveis para o normal funcionamento da nossa aplicação, nomeadamente, o preço unitário por consulta, o número de pedidos de inscrição de médicos e pacientes (para atribuição de um id a cada utilizador aquando de um novo pedido de inscrição na plataforma) e ainda a password para direitos de acesso à plataforma como administrador, de modo a poder alterar o preço da consulta.

Existem outras tabelas que garantem alguma informação essencial, nomeadamente a tabela **Prescricao** que associa a cada consulta um determinado conjunto de fármacos que, todos juntos, formarão uma receita que ficará associada a essa mesma consulta para que seja possível, posteriormente, para o paciente fazer o respetivo download da mesma. Já a tabela **InfoGeneral** permite associar a cada paciente uma determinada informação

que seja considerada útil por parte deste próprio. Existem ainda as tabelas **Contactos** e **Código Postal** que permitem associar a uma determinada conta (de médico ou de paciente) um determinado conjunto de contactos e uma morada (Código-Postal e Distrito).

4.2.4 Controller

Relativamente ao **controller**, existem 3 classes para fazer a comunicação entre o **backend** e o **frontend**, sendo elas **AdminController**, **ContasController** e **ConsultasController**. Estas classes são responsáveis pela transmissão de dados entre o **backend** e o **frontend**, transmitindo os dados essencialmente em json, graças a classes **Model**, que representam os tipos de dados usados na comunicação da aplicação.

Assim sendo, a classe **AdminController**, é responsável pela transmissão de dados usados no perfil administrador com o backend, que permitem aceitar médicos e alterar o preço de consultas, bem como mostrar o número de médicos e pacientes inscritos na aplicação.

Em relação ao **ContasController**, é transmitida a informação necessária para que tanto o *Paciente* como o *Médico* tenham a informação pessoal das suas contas, bem como para o procedimento de login por partes de ambos os utilizadores do sistema. Esta classe permite ainda que ambos os utilizadores editem o seu perfil, na eventual necessidade de alterar algum dado do seu perfil.

De forma um pouco similar, a classe **ConsultasController**, é responsável pela transmissão de dados relativos às consultas, desde marcar uma consulta, aceitar uma consulta, marcar uma consulta como efectuada, ver as consultas agendadas, aceder ao histórico de consultas, bem como a transmissão dos dados necessários para gerar os ficheiros pdf relativos a uma consulta selecionada, sendo eles, a receita da consultas e a sua fatura/recibo.

Todos estes dados são transferidos através de HTTP, utilizando os métodos existentes para a transmissão, como por exemplo o método GET, PUT ou POST.

4.2.5 Front-end

Para o desenvolvimento do **Front-end**, foi usada a ferramenta **React-JS**, que foi uma grande ajuda para lidarmos com o javascript, o que nos permitiu uma melhor construção da interface para com o utilizador, tornando a aplicação mais amigável e mais agradável aos olhos do utilizador, contando com diversos **packages**, desta mesma ferramenta, o que facilitou em muito a estruturação desta interface.

Para a interação do **front-end** com o **back-end**, foi usado o **axios**, que facilitou o envio e a receção dos dados necessários para o correto funcionamento da aplicação, com todos os dados que o utilizador necessita. A escolha do **axios** foi porque nos pareceu a solução mais simples e eficaz de enviar e receber dados através do *Http*. Já ao nível dos tokens, para os descodificar usou-se o package **jwt-decode** que é de fácil uso e rapidamente se consegue obter o token descodificado. Para definir as rotas das nossas telas, e também definir as telas que poderão ser usadas perante uma determinada situação, como por exemplo se um cliente ainda não estiver autenticado, não pode aceder a nenhum perfil, usou-se o **react-router-dom**. Em certos casos usamos também *State Hooks*, sendo esta uma nova adição que foi feita no *React* e que permite possuir um estado sem ser necessário escrever uma classe.

O design do site é algo extremamente importante, tendo em conta que se trata de um website de marcação de consultas médicas de rotina é preciso ter em atenção que a idade

média do utilizador poderá muito bem ser superior ao normal comparando com outros websites. Desta maneira foi importante tomar decisões do que diz respeito ao aspeto visual do website; a interface foi desenhada de modo a ser considerada user-friendly, capaz de ser utilizado por pessoas de faixa etária superior, mas também complexo o suficiente para responder a todos os requisitos no que à funcionalidade do sistema diz respeito. Para o ”desenho” do nosso website utilizamos a framework de CSS **tailwindcss**, isto surgiu depois de uma primeira fase em que não estavamo a usar framework nenhuma e que deixou os resultados aquém da expectativa, apesar do site estar funcional não se encontrava ”elegante” e partimos à procura de soluções. Esta framework de baixo nível é altamente personalizável, o que nos permitiu construir designs sem largar o HTML; para além disto, conta com o apoio de uma grande comunidade que disponibiliza imensos templates e componentes Open source que podemos modificar completamente a nosso gosto (<https://tailwindcomponents.com/>). Deste modo, conseguimos contruir uma interface que satisfaz a vista e que se mantém prática e fácil de entender.

```
<button class="uppercase mx-auto shadow bg-indigo-800 hover:bg-indigo-700 text-white text-xs py-3 px-10 rounded"> Login </button>
```

Exemplo de snippet de código de tailwindcss que gera um botão ”Login”, com diretivas para o tamanho, forma e cores do botão.

De modo a oferecer uma maior comodidade e diversificar as funcionalidades da aplicação, às quais o paciente tem acesso, foi desenvolvido um mecanismo de geração de receitas, onde o médico após uma consulta é capaz de proceder à prescrição de medicamentos, se assim achar necessário, em que esta pode ser acedida pelo paciente através do histórico de consultas. Para gerar este pdf foi usado o componente **react-pdf**, o que permitiu gerar o mesmo através de dados provenientes da base de dados, enviados através do *controller*. Do mesmo modo, o paciente pode também fazer o download da fatura/recibo da consulta, com os dados relativos à consulta, desde a informação do médico, como também o preço da consulta, já com a informação do IVA.

De seguida, pode-se ver as imagens relativas ao front-end.

4.3 Resultados



Figura 4.3: Página home (primeira parte)

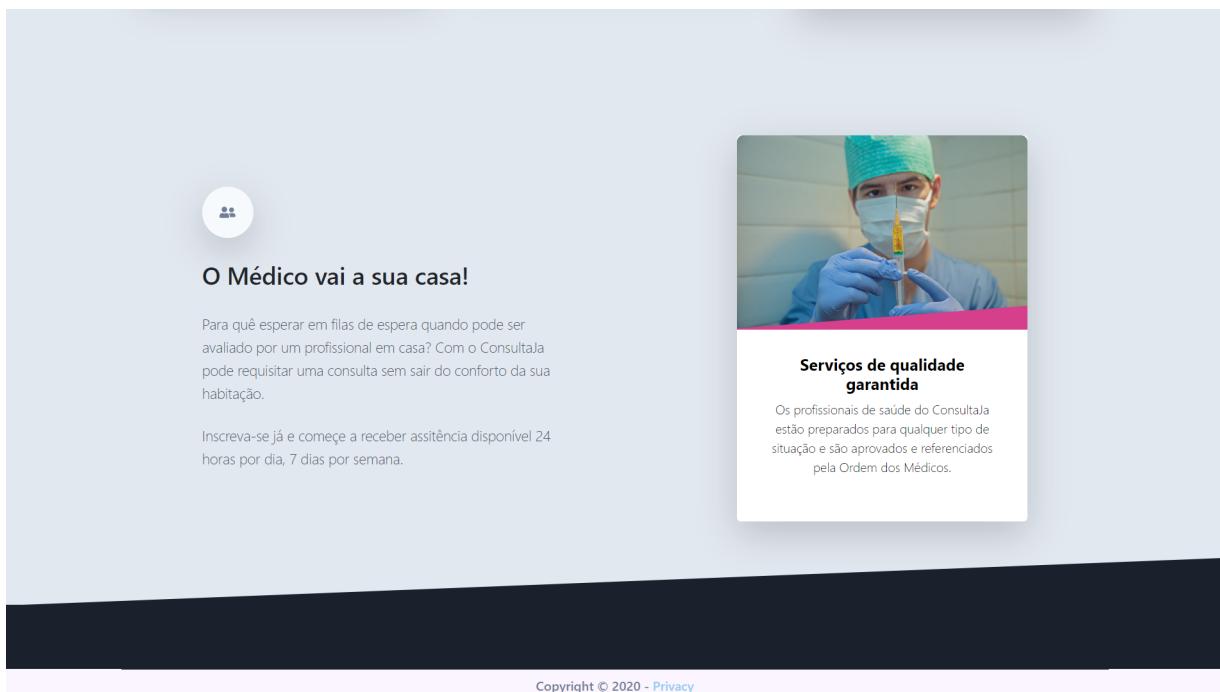


Figura 4.4: Página home (segunda parte)

Figura 4.5: Página registrar

Figura 4.6: Página login

4.3 Resultados

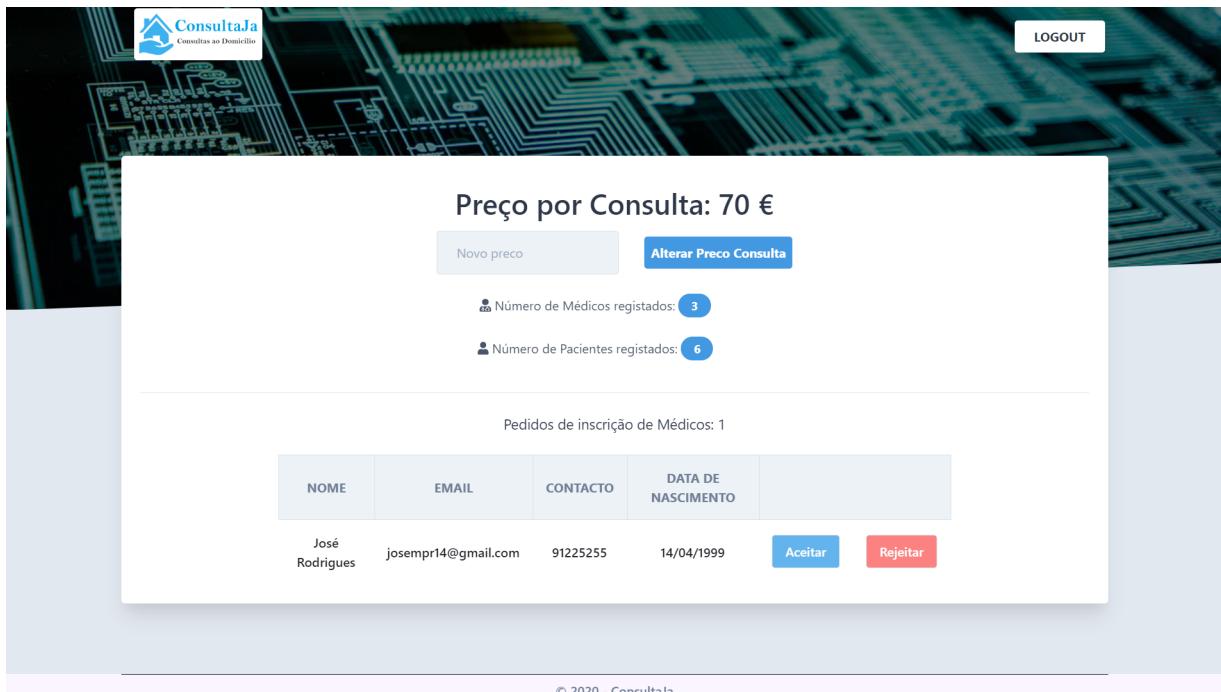


Figura 4.7: Página perfil do administrador

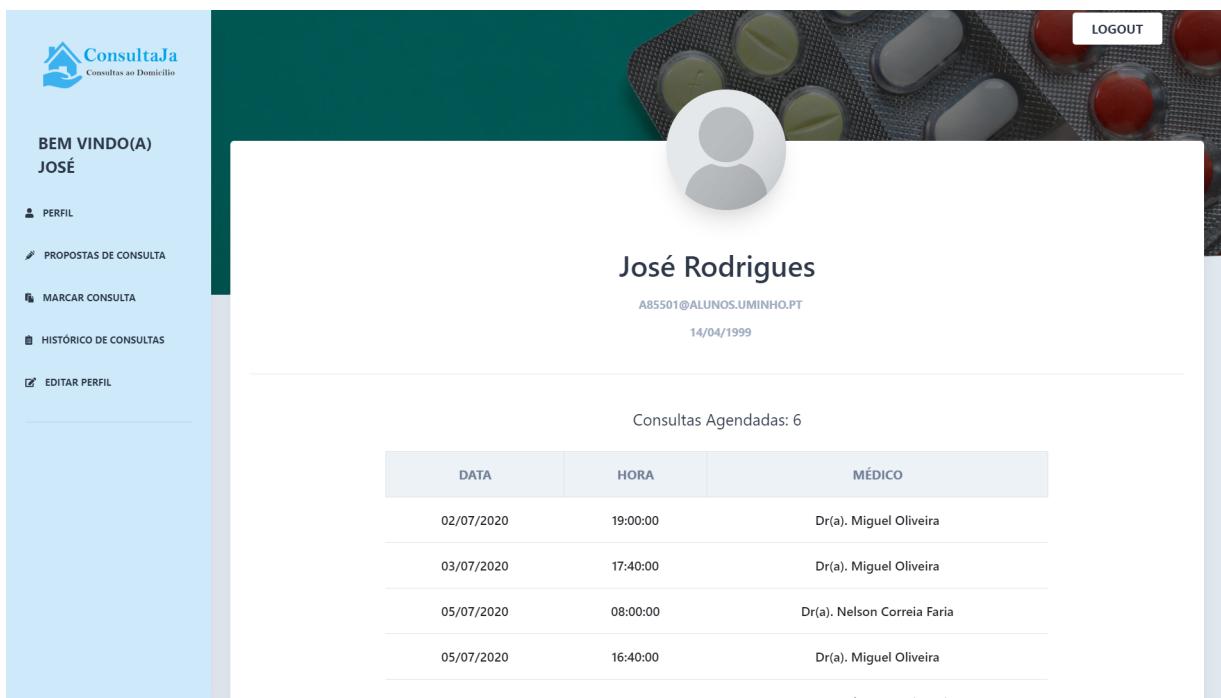


Figura 4.8: Página perfil do paciente



Figura 4.9: Página propostas de consulta de um paciente, aceites pelo médico e que aguardam aprovação final do paciente



Figura 4.10: Página marcar consulta por parte do paciente

4.3 Resultados

The screenshot shows a mobile application interface for 'ConsultaJa'. On the left, there is a sidebar with the app's logo and the text 'BEM VINDO(A) JOSÉ'. Below this, there are several menu items: 'PERFIL', 'PROPOSTAS DE CONSULTA', 'MARCAR CONSULTA', 'HISTÓRICO DE CONSULTAS' (which is currently selected), and 'EDITAR PERFIL'. The main content area is titled 'Histórico de Consultas' and displays a table of consultations. The table has columns for 'DATA', 'HORA', 'MÉDICO', and 'RECEITA/RECIBO'. Each row in the table represents a consultation, and there is a blue 'SELEÇÃO' button to its right. At the bottom of the table are 'PREVIOUS' and 'NEXT' navigation buttons.

DATA	HORA	MÉDICO	RECEITA/RECIBO
30/06/2020	18:45	Dr(a). Nelson Correia Faria	SELEÇÃO
27/06/2020	16:40	Dr(a). Nelson Correia Faria	SELEÇÃO
26/06/2020	11:59	Dr(a). Miguel Oliveira	SELEÇÃO
26/06/2020	10:56	Dr(a). Nelson Correia Faria	SELEÇÃO
25/06/2020	18:00	Dr(a). Nelson Correia Faria	SELEÇÃO
25/06/2020	17:40	Dr(a). Miguel Oliveira	SELEÇÃO
25/06/2020	17:20	Dr(a). Miguel Oliveira	SELEÇÃO
24/06/2020	15:35	Dr(a). Nelson Correia Faria	SELEÇÃO

Figura 4.11: Página histórico do paciente, onde são apresentadas as consultas realizadas pelo paciente

This screenshot is similar to Figure 4.11, showing the 'Histórico de Consultas' page. However, it shows that a specific consultation has been selected. The table now includes two additional buttons at the top: 'DOWNLOAD RECEITA!' and 'DOWNLOAD RECIBO!'. The rest of the interface, including the sidebar and the table structure, remains the same.

DATA	HORA	MÉDICO	RECEITA/RECIBO
30/06/2020	18:45	Dr(a). Nelson Correia Faria	SELEÇÃO
27/06/2020	16:40	Dr(a). Nelson Correia Faria	SELEÇÃO
26/06/2020	11:59	Dr(a). Miguel Oliveira	SELEÇÃO
26/06/2020	10:56	Dr(a). Nelson Correia Faria	SELEÇÃO
25/06/2020	18:00	Dr(a). Nelson Correia Faria	SELEÇÃO
25/06/2020	17:40	Dr(a). Miguel Oliveira	SELEÇÃO
25/06/2020	17:20	Dr(a). Miguel Oliveira	SELEÇÃO
24/06/2020	15:35	Dr(a). Nelson Correia Faria	SELEÇÃO

Figura 4.12: Página histórico do paciente, após selecionar uma consulta, em que aparecem os botões para o download da receita e do recibo relativos à consulta selecionada

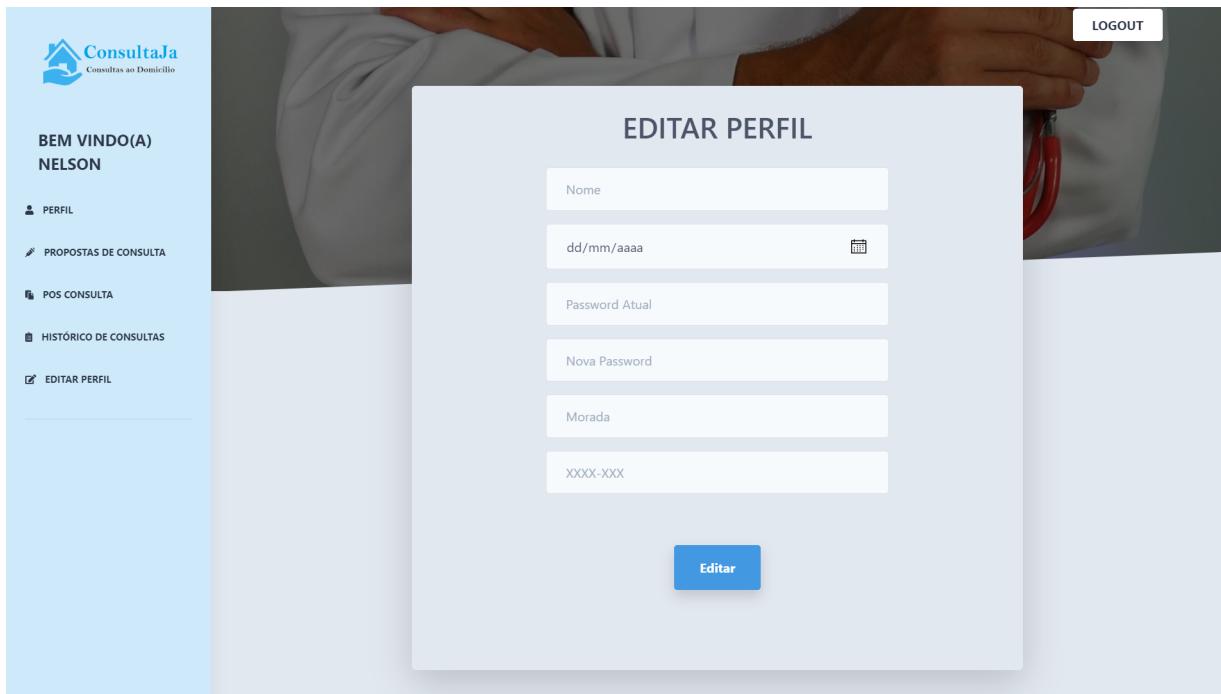


Figura 4.13: Página editar perfil, que é igual tanto no paciente como no médico



Figura 4.14: Página logout, apresentada após o término da sessão por parte do utilizador

4.3 Resultados

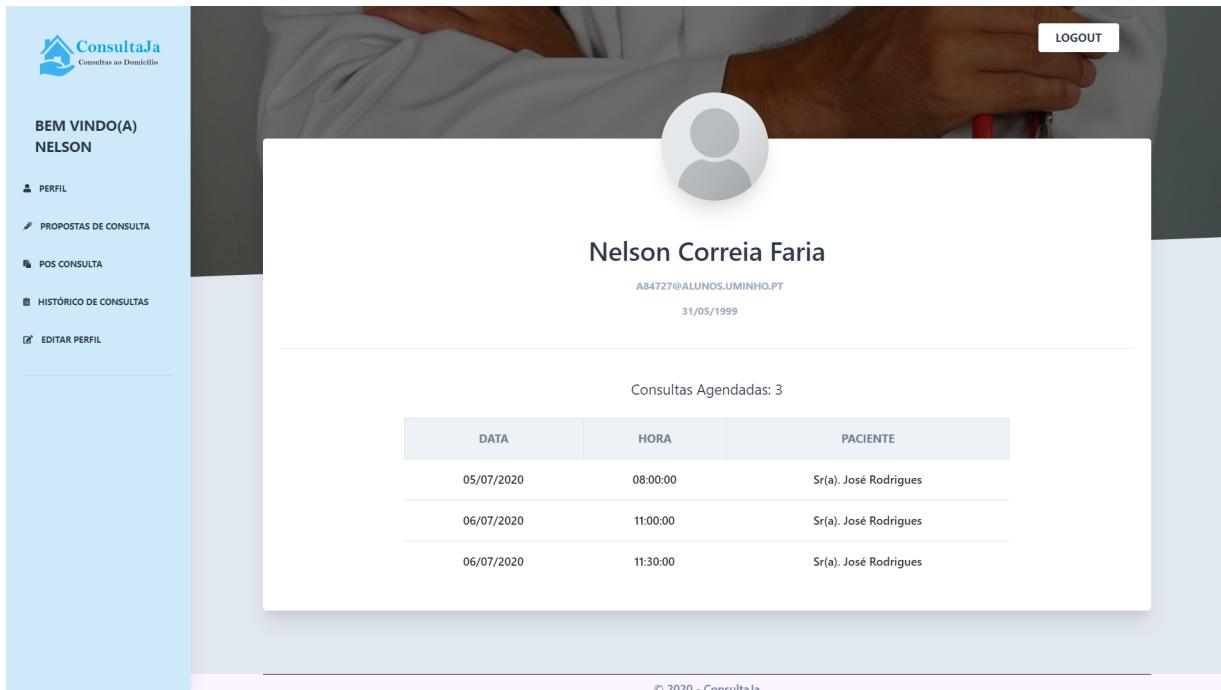


Figura 4.15: Página perfil do médico



Figura 4.16: Página propostas de consulta de um médico, feitas por um paciente, onde o médico tem a possibilidade de aceitar a mesma

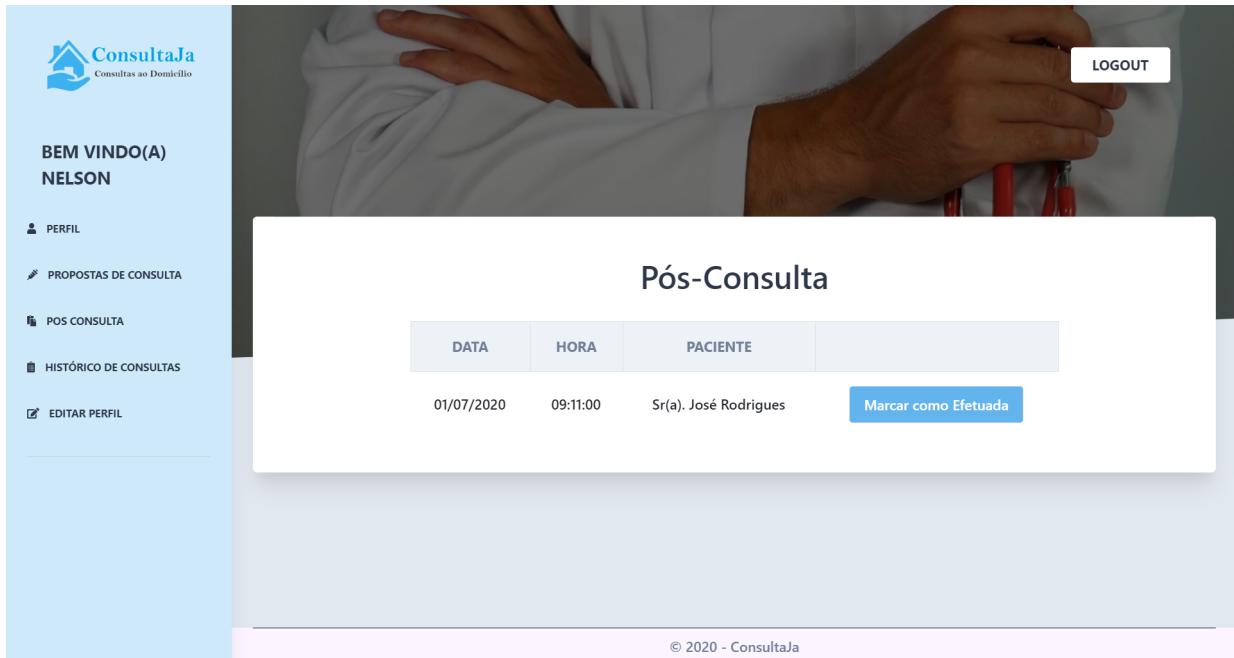


Figura 4.17: Página pós consulta de um médico, onde o médico pode marcar um consulta como efetuada, passando para a tela abaixo e passar a devida receita, com os respetivos medicamentos, assim como uma pequena observação da consulta

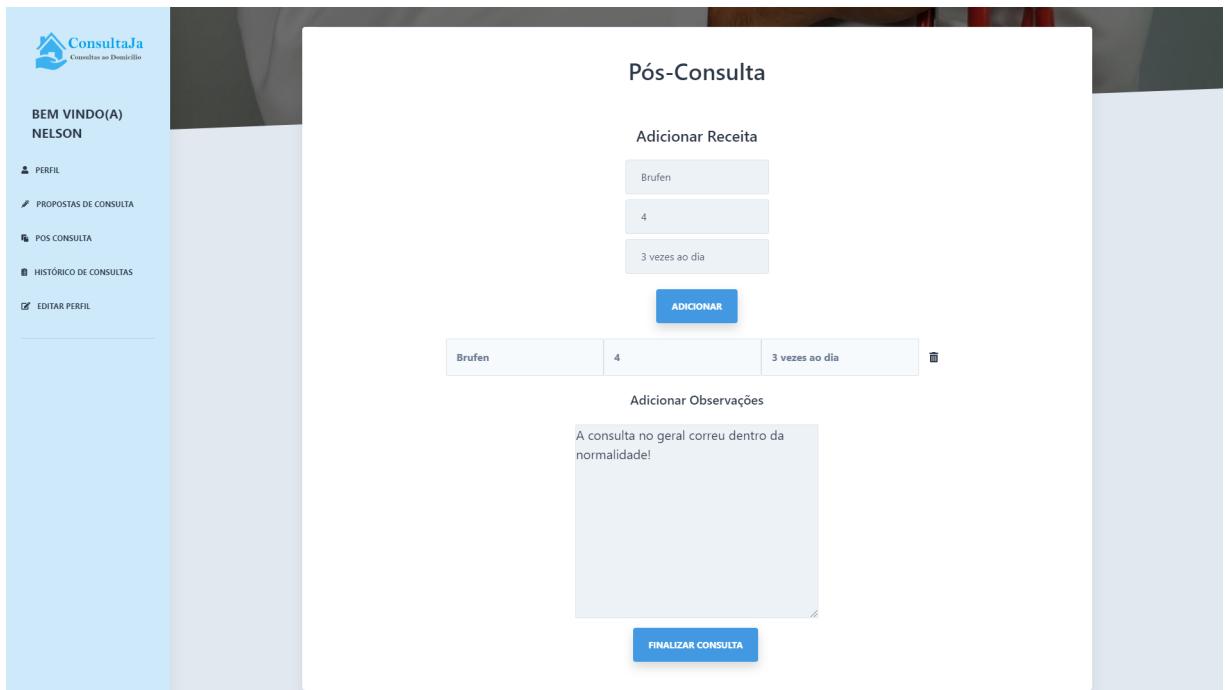


Figura 4.18: Página pós consulta de um médico, onde o médico pode passar a devida receita, com os respetivos medicamentos, assim como uma pequena observação da consulta

DATA	HORA	PACIENTE
30/06/2020	18:45	Sr(a). José Rodrigues
27/06/2020	16:40	Sr(a). José Rodrigues
26/06/2020	10:56	Sr(a). José Rodrigues
25/06/2020	23:28	Sr(a). Elsa Faria
25/06/2020	18:00	Sr(a). José Rodrigues
24/06/2020	15:35	Sr(a). José Rodrigues
24/06/2020	15:10	Sr(a). José Rodrigues
16/06/2020	09:35	Sr(a). José Rodrigues

Figura 4.19: Página histórico de consultas do médico, onde este pode ver as consultas que já foram realizadas por si

1. Questões

Caso tenha qualquer questão relacionada com o tratamento dos seus dados pessoais e com o exercício dos direitos que lhe são conferidos pela legislação aplicável contacte-nos através dos seguintes contactos:

E-mail: consultaja4@gmail.com

2. O que são dados pessoais?

Dados pessoais são qualquer informação, de qualquer natureza e independentemente do respetivo suporte, incluindo som e imagem, relativa a uma pessoa singular identificada ou identificável.

É considerada identificável a pessoa singular que possa ser identificada, direta ou indiretamente, designadamente por referência a um nome, número de identificação, dados de localização, identificadores por via eletrónica ou a um ou mais elementos específicos da sua identidade física, fisiológica, genética, mental, económica, cultural ou social.

3. No que consiste o tratamento de dados pessoais?

O tratamento de dados pessoais consiste numa operação ou conjunto de operações efetuadas sobre dados pessoais ou conjuntos de dados pessoais, através de meios automatizados, ou não, nomeadamente a recolha, o registo, a organização, a estruturação, a conservação, a adaptação, a recuperação, a consulta, a utilização, a divulgação, difusão, comparação, interconexão, a limitação, o apagamento ou a destruição.

Copyright © 2020 - Privacy

Figura 4.20: Página privacidade da aplicação

5 CASOS DE ESTUDO/EXPERIÊNCIA

Ao longo do desenvolvimento do nosso website tivemos que fazer vários estudos sobre tecnologias, ferramentas, público alvo e até mesmo possíveis concorrentes. De modo algo resumido passamos a expor de seguida várias experiências e casos de estudos que fomos efetuando nos últimos 4 meses:

- **Estudo de Mercado**

Uma vez que a nossa aplicação é toda construída em português o nosso website terá Portugal e ilhas como zona de aplicação. Neste contexto espacial é essencial perceber quem é a nossa concorrência e quem é o nosso público, garantindo que temos algo único dentro da nossa área de ação. O estudo de mercado deverá dar-nos uma ideia sobre a viabilidade comercial da nossa atividade. De modo a efetuar uma eficaz análise dividimos o estudo em 2 elementos fundamentais: Mercado e Concorrência.

Chegamos à conclusão que as consultas de medicina geral são largamente as mais utilizadas: Segundo o instituto nacional de estatística em 2012 foram dadas mais de 26 milhões de consultas médicas em Portugal, desses 26 milhões, mais de 21 foram de Medicina Geral. Em 2018 este número ultrapassou os 40 milhões. Isto aliado ao facto da pouca concorrência, que apesar de algumas semelhanças não tem exatamente o mesmo conceito, e que na sua maioria trabalham em áreas mais localizadas e não no país todo (falado no "State of the art"), ao pouco material médico necessário para uma consulta de rotina e da maior queixa dos portugueses relativamente ao Serviço Nacional de Saúde ser as longas filas de espera resolvemos avançar com o projeto ConsultaJa.

- **Pesquisa de Tecnologias**

Antes de partirmos para o desenvolvimento da aplicação, e com plena consciência de que o mundo do desenvolvimento web move-se bastante depressa devido às novas tecnologias emergentes (especialmente no frontend), vimo-nos na obrigação de recolher informação sobre as potenciais tecnologias a usar. O objetivo foi perceber melhor o que se costuma usar nesta área de desenvolvimento e escolher algo que não seja obsoleto daqui a um ano. Como vamos tivemos que lidar com a parte do cliente e servidor (full stack) recolhemos dados sobre estas duas vertentes:

- As essenciais: HTML, CSS e JavaScript:

Tecnologias extremamente importantes para qualquer tipo de trabalho de desenvolvimento front-end. JavaScript é a base incondicional e é utilizado por mais de 95% dos sites. Para desenvolver o ConsultaJa, tivemos que saber um pouco destas 3 componentes.

- Vue, JQuery, BootStrap, Angular, React

Frameworks na sua maioria de JavaScript, que tornam mais fácil a tarefa do programador no que a trabalhar com JavaScript diz respeito. Para o nosso projeto tínhamos que utilizar uma e acabamos por escolher React devido às suas vantagens: re-usabilidade de código, programação simples (fácil de ler), boa performance e uma comunidade grande. No entanto qualquer uma destas tecnologias seria adequada.

-
- ASP.NET, PHP, Python, Ruby

Fizemos uma pesquisa de tecnologias Back-End em linguagens que ainda não utilizamos no nosso curso. Mais uma vez, qualquer uma das linguagens de programação listadas seria adequada para o desenvolvimento de um website, no entanto, a equipa docente desafiou-nos a utilizar tecnologias microsoft no nosso projeto o que tornou esta escolha bastante simples e acabamos por utilizar C# (.NET CORE).

- NoSQL, SQL

Como vamos ter que guardar informação sobre utilizadores vamos ter que ter uma Base de Dados. Há dois tipos de base de dados: NoSQL e SQL. Tal como mencionado no capítulo 3, a nossa aplicação nunca atingiu um grande volume de informação portanto não se justificou sacrificar consistência de dados para utilizar uma BD não relacional, como tal resolvemos aproveitar os nossos conhecimentos de MySQL e implementamos uma base de dados relacional.

- **Reformulamento do FrontEnd**

A meio do desenvolvimento da aplicação deparamo-nos com alguns precalços que foram sendo apontados pela professora. Um deles foi o facto de apesar do nosso site estar funcional, não se encontrava apelativo visualmente. Como a funcionalidade é importante mas o design também tivemos que arranjar maneira de reformular o visual do ConsultaJa. Para isso, seguimos uma recomendação da professora e utilizamos uma framework de CSS chamada TailwindCSS. Apesar de ser um desafio a meio do desenvolvimento de aprender a usar esta nova framework, tornou-se numa importante ferramenta que ajudou a melhorar imenso o aspecto do site.

6 CONCLUSÃO

6.1 Conclusões

Este projeto foi, sem dúvida, o projeto mais extenso que a nossa equipa alguma vez se deparou e o balanço final é bastante positivo. Desde a fase inicial onde tivemos que contextualizar o projeto a desenvolver e o porquê de o desenvolver, tentando justificar o desenvolvimento do sistema. Isto permitiu decidir se realmente há uma necessidade do ConsultaJa no mercado e o contexto onde esta aplicação será inserida, para poder construir um sistema voltado aos seus utilizadores.

Numa segunda fase foi tratado a recolha dos requisitos e modelação do sistema a projetar tendo em conta esses mesmos requisitos. Desenvolvemos ainda a modelação da base de dados tendo em conta esses requisitos.

Na última fase, tratamos então do desenvolvimento do software, criando a base de dados e povoando-a e desenvolvendo a aplicação web. Este projeto tornou-se ao longo do tempo num projeto bastante enriquecedor, na medida que deu aos alunos liberdade de escolha no tema e em várias tecnologias, incentivando uma aprendizagem contínua e autodidata de tecnologias microsoft e de, no nosso caso, ReactJs.

Para terminar, resta dizer que o grupo considera o resultado final positivo, pelo que *LI4* proporcionou uma excelente introdução ao desenvolvimento full stack de websites garantindo que os alunos levem uma experiência bastante relevante para a vida profissional futura.

6.2 Perspetiva para trabalho futuro

Este projeto pode ainda ser melhorado, implementando novos mecanismos que ofereçam mais funcionalidades aos utilizadores, de modo a tornar a aplicação mais cómoda e mais apelativa, para que cada vez mais seja usada pelo seu público alvo. Para isso, pode-se implementar uma carteira digital, que permitisse aos utilizadores pagarem diretamente as consultas através da aplicação, como também a implementação de um sistema de pagamento que suportasse diferentes pagamentos, para oferecer uma maior variedade de opções de compra aos seus clientes.

Dessa forma, haveria também mudanças no sistema de faturação, o que iria obrigar a uma mudança da estrutura apresentada ao cliente, de modo a suportar todas estas atualizações do sistema.

Referências

- [1] Tailwind Components, <https://tailwindcomponents.com/>
- [2] Tailwind, <https://tailwindcss.com/>
- [3] React JS, <https://reactjs.org/>
- [4] Microsoft, Visual Studio, JavaScript, <https://docs.microsoft.com/en-us/visualstudio/javascript/?view=vs-2019>
- [5] Microsoft, Visual Studio, C#, <https://docs.microsoft.com/en-us/visualstudio/get-started/csharp/?view=vs-2019>