Proposta de Projeto e Seminário - LEIC – LI61N – 1718v

Sistema de partilha entre médicos de dermatologia da ficha de utente, incluindo fotos de sinais dermatológicos

Luís Moreira 40977@alunos.isel.ipl.pt

Hélio Cavudissa 41628@alunos.isel.ipl.pt

Relatório Beta

Orientadores:

Jorge Martins jmartins@cc.isel.ipl.pt

Vítor Almeida valmeida@deetc.isel.ipl.pt

Medical Skin Care

Maio 2018

Conteúdo

1. Introdução

Atualmente, não é fácil os hospitais saberem o histórico médico dos seus pacientes na totalidade, como, exames realizados em outras unidades hospitalares de outros países e não só, pois não existe um protocolo de partilha de dados entre entidades hospitalares. A nossa aplicação surge com objetivo de aproximar as unidades hospitalares de forma a poderem partilhar as fichas de utente com a devida permissão dos utentes.

Pretendemos desenvolver uma aplicação WEB de apoio ao diagnóstico por dermatoscopia diferencial com partilha entre médicos da ficha dermatológica do utente. Com especial consideração na privacidade dos dados do utente, como tal o tratamento dos dados constantes na ficha do utente de dermatologia terá em consideração requisitos fundamentais, tais como: confidencialidade, integridade, e a conformidade legal e normativa, nomeadamente tendo em consideração a privacidade dos dados.

Doravante vamos nos referir à aplicação como Medical Skin Care.

2. Enquadramento

Como foi referido acima, não existe um protocolo de partilha de fichas do utente entre unidades hospitalares, atualmente a metodologia mais próxima é o relatório médico, que não inclui muita informação que pode ser essencial para as próximas consultas.

Com a partilha das fichas de utentes pretendemos alargar a quantidade de informação sobre o paciente, que um dermatologista terá acesso, informações como fotos de sinais, histórico familiar, histórico médico, informações essa que ajudarão o dermatologista ter uma melhor perceção sobre a patologia.

2.1. Tecnologias Existentes.

Dos estudos feitos sobre os softwares existentes na abordagem do nosso problema, podemos encontrar dois softwares opensource voltados para dermatologia:

2.1.1. Dermatron

É um software de histórico médico, com as seguintes funcionalidades:

- Gerir consultas e registo de pacientes
- Pesquisa de medicamentos.
- Agendar consultas.
- Inserir diferentes tipos de lesão
- Inserir tipos diferentes de sintomas
- 28 diferentes tipos de fisiopatologia
- 59 diferentes localizações anatômicas
- 727 diferentes opções de diagnóstico
- Captura de imagem do Dermatoscópio USB
- Sugestões de diagnóstico baseadas na Web

2.1.2. OPENMRS, SkinHelpDesk.

O OpenMRS é uma plataforma de software de referência que permite o desenho de um sistema de histórico médicos personalizados sem conhecimento de programação (embora o conhecimento médico e de análise de sistemas seja necessário). O sistema é baseado em uma estrutura de base de dados conceitual que não depende dos tipos reais de informações médicas que devem ser coletadas ou de formulários de coleta de dados específicos e, portanto, podem ser personalizados para diferentes usos.

Dentre muitas funcionalidades que o OPENMRS, as mais destacadas são:

- Segurança: autenticação do utilizador
- Acesso baseado em privilégio: funções dos utilizadores e sistema de permissão
- Repositório de pacientes: Criação e manutenção de dados de pacientes, incluindo dados demográficos, observações clínicas, pedidos, etc.
- Vários identificadores por paciente: um único paciente pode ter vários números de registros médicos
- Exportação de dados: os dados podem ser exportados para um formato de planilha para uso em outras ferramentas (Excel, Access, etc.)
- Arquitetura modular: um módulo OpenMRS pode estender e adicionar qualquer tipo de funcionalidade à API e ao webapp existentes.
- Fluxos de trabalho do paciente: um serviço de fluxo de trabalho do paciente incorporado permite que o paciente seja colocado em programas (estudos, programas de tratamento, etc.) e rastreado por vários estados.
- Gerenciamento de coorte: o criador de coorte permite que você crie grupos de pacientes para exportação de dados, relatórios, etc.
- Relacionamentos: Relacionamentos entre duas pessoas (pacientes, parentes, cuidadores, etc.)
- Localização / internacionalização: suporte a vários idiomas e a possibilidade de se estender a outros idiomas com suporte total a UTF-8.
- Suporte para dados complexos: imagens radiológicas, arquivos de som, etc. podem ser armazenados como observações “complexas”
- Ferramentas de relatórios: ferramentas de relatórios flexíveis
- Atributos da pessoa: os atributos de uma pessoa podem ser estendidos para atender às necessidades locais.

O OPENMRS oferece um módulo específico para a dermatologia que oferece algumas funcionalidades adicionais e específica.

Um utilizador registado poder submeter o seu “skin profile”. O “skin profile” inclui problemas de pele, detalhes sobre o problema, como quando e onde começou. Também permite fazer upload de fotos ou capturar imagens diretamente do webcam. O perfil do utilizador está disponível para a equipa de dermatologistas, mas não para o público em geral. A equipa de dermatologistas pode opinar sobre um “skin profile”.

2.2. Medical Skin Care

Depois de termos estudados os softwares mencionados acima, decidimos começar um projeto de raiz, pelo desafio que o problema apresenta e pelos seguintes aspetos dos softwares:

Dermatron:

- Não pretendemos fazer gestão de consultas, deixando isso ao cargo da entidade hospitalar.
- Não pretendemos gerir os produtos farmacêuticos duma entidade hospitalar
- Não pretendemos fazer diagnostico de doenças na web, ficando apenas ao cargo dos profissionais da saúde registados na aplicação.

SkinHelpDesk:

- Queremos garantir a privacidade dos dados dum utente, pretendemos que o utente tenha conhecimento das equipas que acedem aos seus dados
- Os Dados de um paciente somente podem ser acedidos se o mesmo der a permissão de acesso.

Apesar das ferramentas apresentadas acima referirem alguns requisitos do projeto, como captura de imagem do Dermatoscópio via USB (Dermatron), e a análise do perfil de um utente por uma equipa de dermatologista (SkinHelpDesk), o principal objetivo da nossa aplicação é a partilha da ficha de utente e restringir a adição e analise da ficha de um utente aos profissionais autorizados, o que nos leva ao outro grande motivo para construir a aplicação de raiz, o nosso software já conta com o **novo Regulamento Geral de Proteção de Dados (RGPD)** que entra em vigor em 25 de Maio de 2018 e substitui a atual diretiva e lei de proteção de dados em vigor. Tivemos em conta os principais pontos do novo regulamento, nomeadamente:

- Informação aos titulares dos dados
- Consentimento dos titulares dos dados
- Processos de Segurança e Tratamento de Dados
- Proteção de dados desde a conceção

Nas próximas secções apresentaremos uma descrição mais detalhada de como solucionamos os problemas que foram discutidos nesta secção.

2.2.1 Objetivos

Pretende-se no final deste projeto, apresentar uma aplicação web com as seguintes características:

- Suporte de ficha de utente de dermatologia
- Ficha do utente com acesso condicionado por tipo de utilizador da aplicação.
- Tratamento especializado da informação sobre dermatologia na ficha de dermatologia do utente, nomeadamente possibilidade de inserir notas médicas, associar à ficha imagens de elevada resolução, visualização de imagens, localização no corpo humano da origem da imagem, possibilidade de comparar imagens referentes à mesma localização no corpo, etc. Suporte ao diagnóstico por dermastocopia diferencial.
- Cartão de Cidadão (CC) como forma preferencial de autenticação dos utilizadores.

2.2.2. Utilizadores

As ações possíveis de realizar pelos utilizadores relativamente à ficha devem poder ser distintas conforme o tipo de autorização de cada utilizador, como é descrito a seguir:

2.2.2.1. Administrativo

Um administrativo representa uma entidade hospitalar, com as seguintes funções:

Relativo a médicos e enfermeiros:

- Registar (criar uma conta)
- Apagar
- Atualizar informações
- Integra-los numa equipa
- Remove-los duma equipa
- Adicionar um médico como líder duma equipa

Relativo a utentes:

- Registar (criar uma conta)
- Apagar (dentro da sua unidade)
- Enviar um pedido de autorização para que uma equipe médica tenha acesso aos dados.
- Preencher questionários sobre histórico do utente.

Relativo à equipa médica:

- Criar uma equipa
- Eliminar
- Adicionar integrantes
- Remover integrantes
- Enviar questionários sobre histórico médico do utente

2.2.2.2. Doutor

Funções:

- Criar um grupo de fichas
- Criar uma ficha
- Pode adicionar uma ficha ao grupo de ficha de um utente
- Pode consultar as fichas dum utente
- Pode adicionar notas sobre a ficha dum utente
- Consultar informações sobre médicos e enfermeiros da mesma equipe

2.2.2.3 Enfermeiro

Funções:

- Pode consultar as fichas dum utente
- Pode adicionar notas sobre a ficha dum utente
- Consultar informações sobre médicos e enfermeiros da mesma equipe

2.2.2.4 Utente

Funções:

- Pode consultar as suas fichas
- Conceder permissão de acessos aos seus grupos de ficha

3. Arquitetura

Apresentaremos a seguir os módulos e como eles interagem uns com outros.

3.1. Modelo de Dados

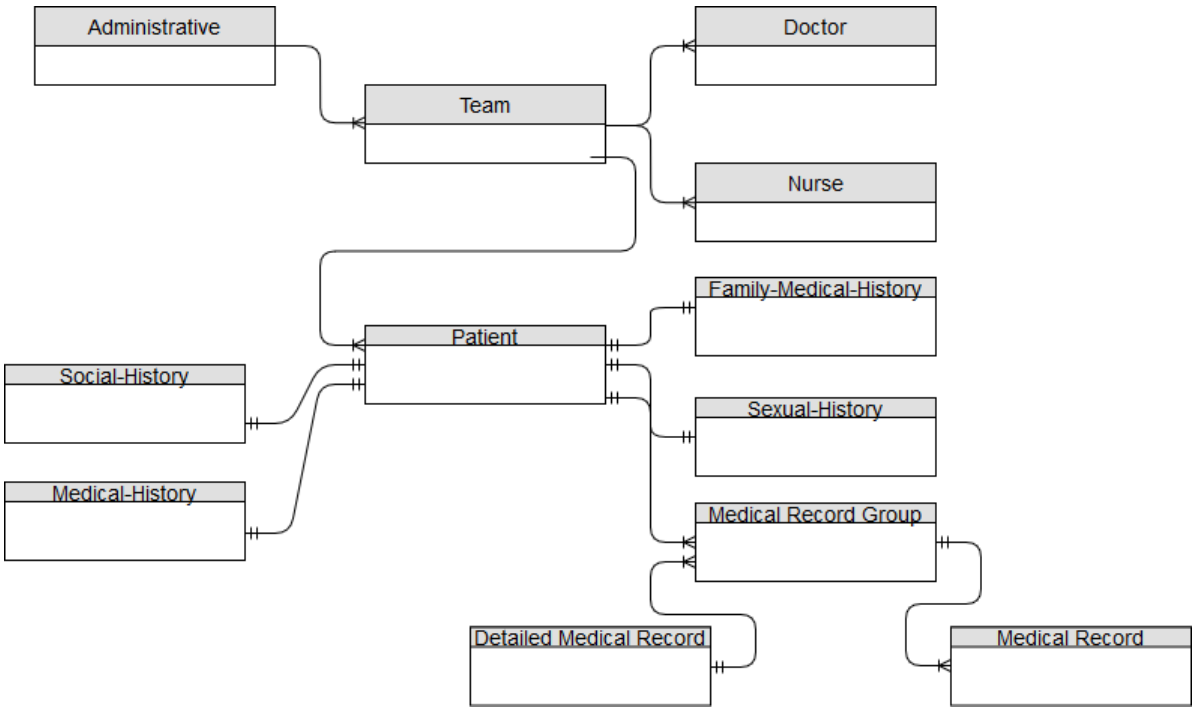

Figura 1

Como mostra a figura acima, o modelo de dados é constituído pelas entidades já acima mencionadas e pelas seguintes entidades:

3.1.1. Team

Um team é uma entidade constituída por um conjunto de médicos, enfermeiros e um médico líder, a motivação da criação desta entidade tem a ver com o facto de se poder ter um conjunto de profissionais que poderão ser consultados sem a necessidade de pedir permissão de acesso ao utente uma vez que todo profissional inserido num team, tem acesso aos dados do paciente que concede a permissão ao mesmo team.

3.1.2. Medical Record

Contem informações relativo a ficha dum paciente, essas informações são constituídas por notas médicas, notas de enfermagens, fotos de sinais e questionários médicos.

3.1.3. Medical Record Group

Por motivos de melhor organização decidimos dividir as fichas em patologias, cada ficha é agrupada por patologia formando um grupo de ficha.

3.1.4. Questionários

Um dos objetivos da aplicação, é a recolha de dados para efeitos estatísticos, por este motivo temos um conjunto de entidades que representam informações que serão adicionadas as fichas do utente, nomeadamente:

- Family Medical History
- Social History
- Sexual History
- Medical History

3.2. Serviços

Os serviços estão divididos de acordo as funcionalidades de cada utilizador, como mostram as figuras a seguir:

3.2.1. Administrativo

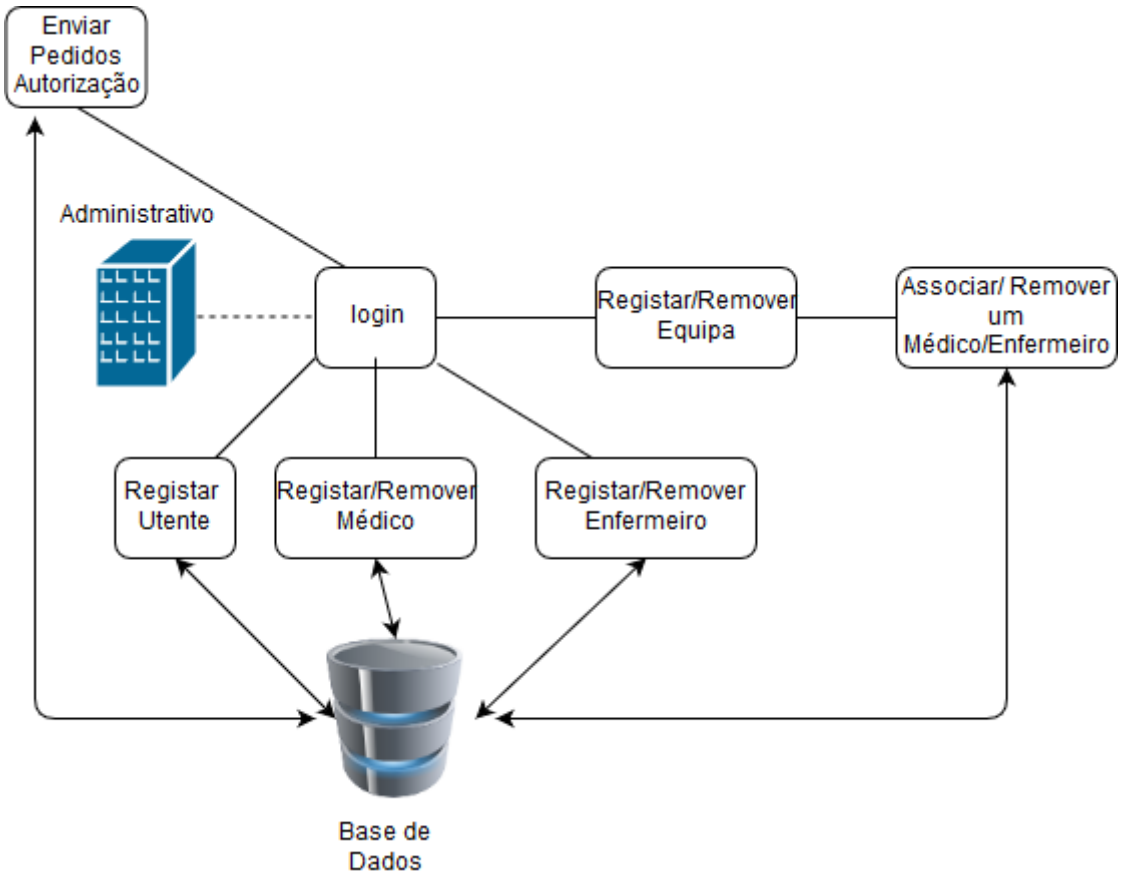

Figura 2

3.2.2 Doutor

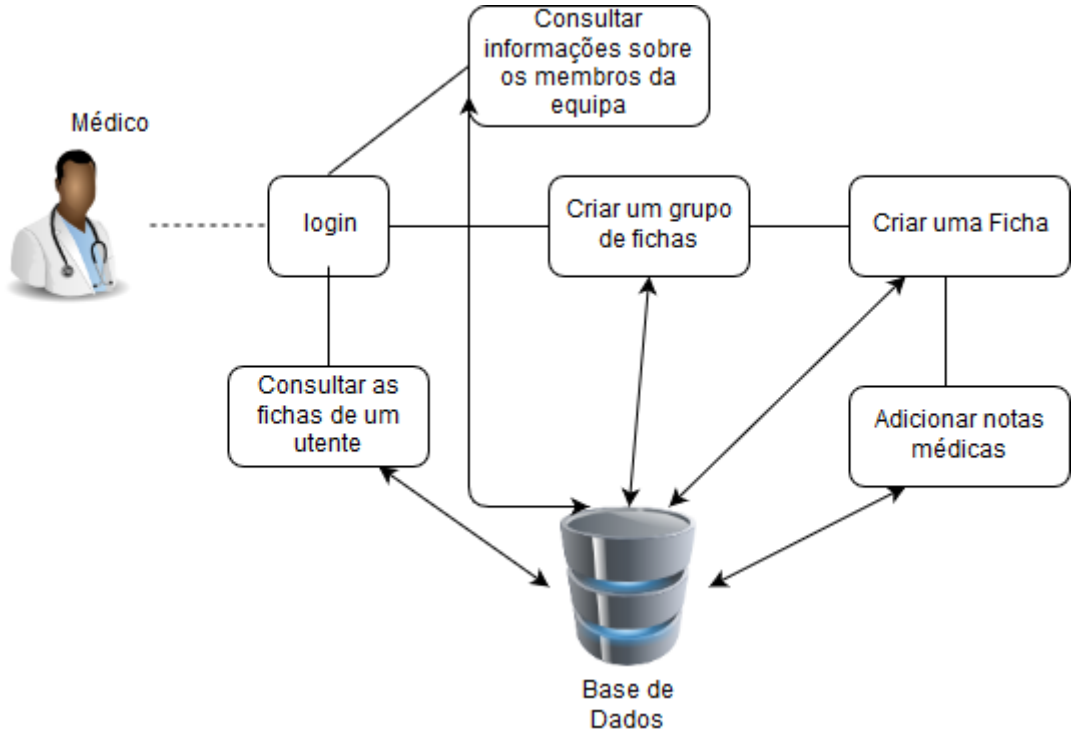

Figura 3

3.2.3 Enfermeiro

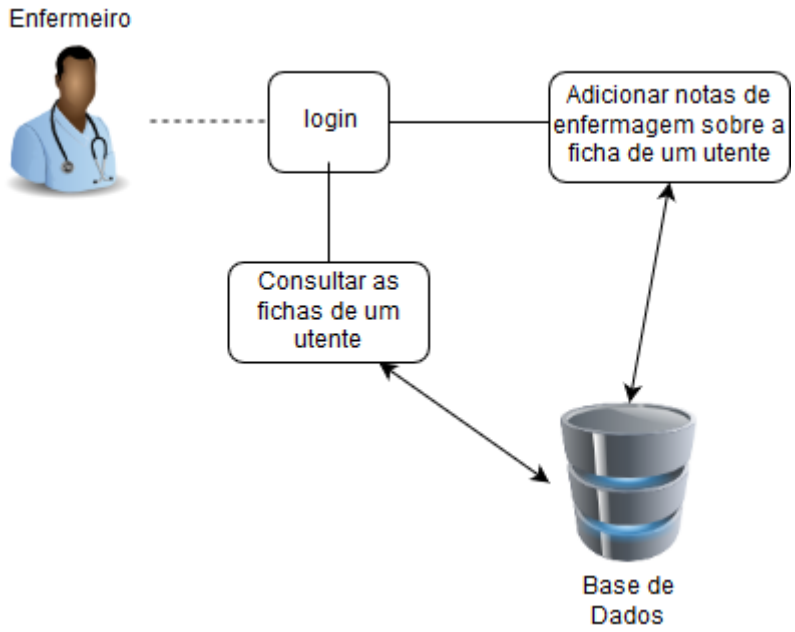

Figura 4

3.2.4 Utente

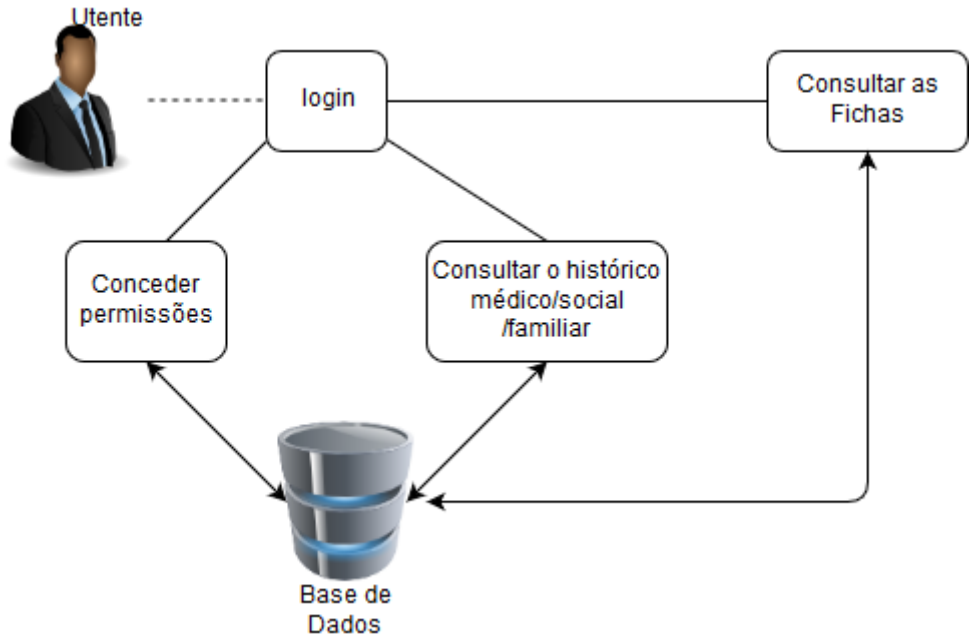

Figura 5

3.3 Routers e Views

Quando é feito um pedido à um endpoint, o pedido é recebido pelo router, que por sua vez envia ao Service, o Service acede a base de dados, a base de dados retorna a informação necessária ao service, esta informação é devolvida ao router que envia para a View, como mostra a seguinte figura.

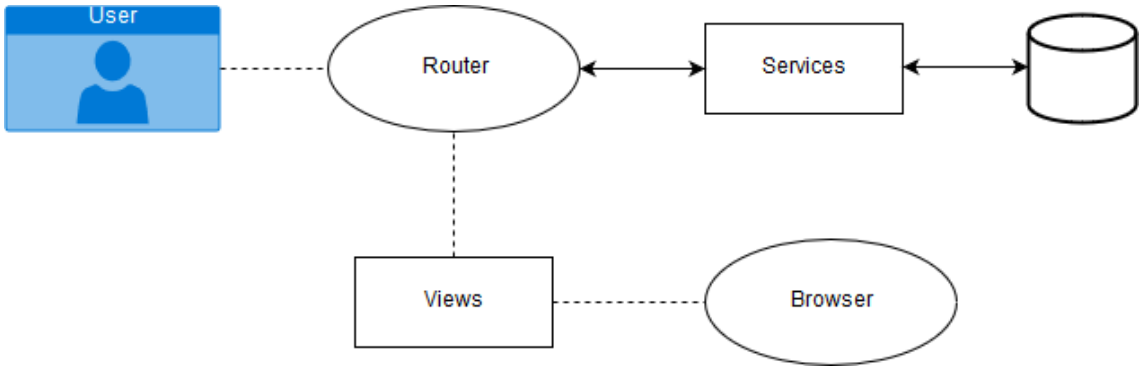

Figura 6

4. Aspetos de Implementação

4.1. Relações entre Modelos.

A relações permitem especificar como os modelos estão relacionados uns com outros. A figura 1, mostra as relações existente entre os modelos.

Tiramos partido da framework loopback (que será explicada mais a frente), que fornece relações entre modelos de modos a permitir a interação entre os mesmos.

hasMany- Permite estabelecer uma relação de um para muitos com outro modelo, logo, cada instancia do modelo declarado terá zero ou mais instancias do outro modelo.
Como exemplo temos que a relação Administrativo e Doutor na figura 7.

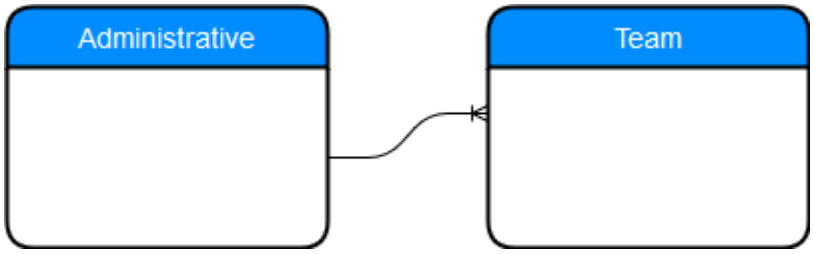

Figura 7

Depois de definir uma relação "hasMany", o Loopback adiciona os seguintes métodos com ao protótipo da class do modelo declarante (Administrative) automaticamente:

- __count__relatedModelNamePlural
- __create__relatedModelNamePlural
- __delete__relatedModelNamePlural
- __destroyById__relatedModelNamePlural
- __findById__relatedModelNamePlural
- __get__relatedModelNamePlural
- __updateById__relatedModelNamePlural

Uma vez estabelecida a relação pode-se usar o modelo Administrative para realizar a operações mostradas acima sobre um team, o exemplo 1 mostra como podemos aceder aos teams através do modelo Administrative:

```
app.models.Administrative.findOne({where: {_id:req.params.instituition}},(err,obj)  =>{
        obj.teams.find…
```

exemplo 1

hasOne - permite estabelecer uma relação de um para um com outro modelo, logo, cada instancia do modelo declarado terá uma instancia do outro modelo.
Como exemplo temos a relação User hasOne Patient.

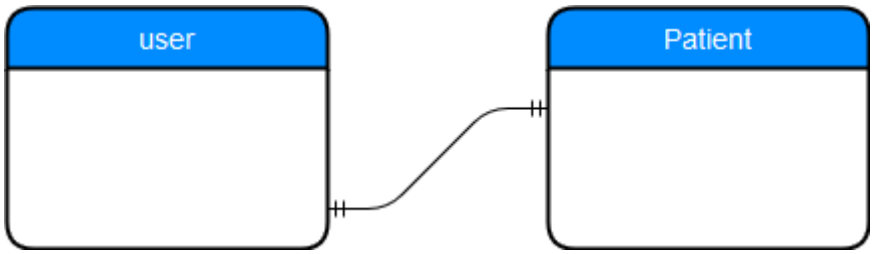

Figura 8

Depois de definir a relação *hasOne*, o Loopback adiciona automaticamente os seguintes métodos ao protótipo da classe do modelo declarante(User):

- __create__relatedModelName
- __get__relatedModelName
- __update__relatedModelName
- __destroy__relatedModelName

Uma vez estabelecida a relação pode-se usar o modelo user para criar, atualizar, apagar e mostrar um paciente, como mostra o pequeno troço de código:

```
user.patient.create(result.data, function(err,doc){
                    if (err) return cb(err) …
```

exemplo 2

onde user é uma instância de User.

belongsTo - Uma relação belongsTo estabelece uma conexão muitos para um ou um para um com outro modelo. Numa relação muitos para um, cada instância do modelo declarante "belongsTo" no máximo uma instância do outro modelo, enquanto o modelo de destino pode ter muitos dos modelos declarantes.
Como exemplos temos as relações MedicalRecordGroup belongsTo Patient, indicando que Patient hasMany MedicalRecordGroup.

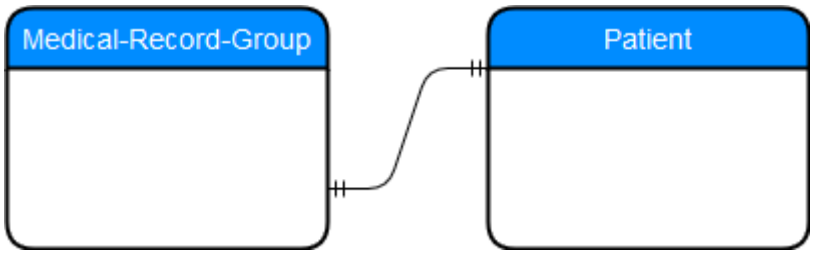

Figura 9

Depois de definir a relação belongsTo, o Loopback adiciona automaticamente o seguinte método ao protótipo da classe do modelo declarante.

- __get__relatedModelName

4.2 Serviço

Todos os dados são cifrados antes de serem guardados na base de dados, desta forma decidimos classificar os serviços em duas partes, os que acedem a base de dados (MongoDB) para armazenar os dados e os que acedem a Cloud (Azure) para gerir a gestão das chaves que são usadas para cifrar os dados.

4.2.1 Acesso à base dados (MongoDB)

Todos os objetos de domínio têm como base o modelo PersistedModel da framework loopback, que faz com que todos eles tenham acesso à um conjunto de métodos fornecidos pelo PersistedModel. Esses métodos (find,findOne,delete,create, etc) , são usados para aceder a base de dados , como mostra o exemplo 5.

```
{
  "name": "Doctor",
  "base": "PersistedModel",
…
            }
```

exemplo 4

Exemplo 4 mostra o objeto Doctor cujo a base é o modelo PersistedModel (será apresentado com mais detalhe na secção sobre tecnologias) da framework loopback

```
this.getPatients = function(cb){
app.models.Patient.find(function(err,doc){
… dados retornados da base de dados
}) }
```

exemplo 5

4.2.2 Acesso à Cloud (Azure)

De modos a garantir a segurança dos dados, os dados do utente são guardados cifrados na base de dados, usamos o sistema simétrico de chaves, tendo em conta a Data encryption key (DEK) Key encryption key (KEK).

Usamos o Azure Key Vault (será apresentado com mais detalhe na secção sobre tecnologias) para encriptar chaves (DEK) e armazena-las (KEK) na cloud.

Usamos instancia de KeyVaultClient (devolvida pelo Azure uma vez autenticados) para criar uma chave que é usada para cifrar o DEK através do método *encrypt* de KeyVaultClient, essa chave é depois guardada na base de dados MongoDB.

Quando for necessário decifrar os dados na MongoDB com a DEK, é necessário usar o método *decrypt* de KeyVaultClient para decifrar primeiro a DEK.

4.2.3 Registo e Autenticação

Embora um utilizador esteja registado, é necessário restringir o acesso do mesmo a certas informações, usamos o loopback para restringir o acesso não autorizado de médicos, enfermeiros aos dados do utente.

A quando do registo do utilizador é-lhe atribuído um Role que o permitirá efetuar determinadas ações, este role será associado ao seu identificador único gerado pela base de dados.

Ao fazer login é atribuído um *token* ao utilizador, através do qual é possível determinar o seu role de modos a garantir a restrição a ações não permitidas ao mesmo. A quando do log out é destruído o token de acesso que lhe foi atribuído anteriormente.

A *framework loopback* fornece um conjunto de modelos embutidos que derivam de *PersistedModel*, que permitem realizar as operações de CRUD.

Para utilizador particular (Paciente, Doutor ou enfermeiro), usamos o modelo User da *framework loopback.* O modelo *User* também deriva de *PersistedMode*l, porém tem mais 4 endpoints que permitem fazer *log in, log out*, confirmar o email e mudar a password.

Criamos o objeto *user* que tem como base o *User* da framework loopback, o objeto user tem relação com todos os utilizadores particular, esta relação permite aos utilizadores particular aceder os 4 endpoints de User (*log in, log out*, confirmar o email e mudar a password).

Quando um *administrativo* regista um utilizador particular são criados dois objetos, um user apenas com email e password, outro(cifrado) que representa o utilizador a ser criado (Paciente, Doutor ou enfermeiro), é estabelecida uma relação entre ambos, lhe é atribuído um role (patient, doctor, nurse), no caso do paciente, é-lhe atribuído uma equipa aquando do registo.

Os detalhes sobre a cifra de dados, role e token de acesso serão explicados na secção sobre tecnologias.

5. Tecnologias

No desenvolvimento duma aplicação com Node.js, em projetos de dimensão considerável, é necessário ter uma API com endpoints que serão consumidos por uma aplicação cliente. Atualmente existem varias frameworks para criar APIs RESTfull, o problema surge na escolha da melhor framework para o problema que enfrentamos.

Dentre as frameworks mais usadas atualmente, encontram:

- Express
- Restify
- Hapi
- LoopBack

5.1. LoopBack

Depois de algumas pesquisas concluímos que para o problema que temos a LoopBack é a melhor framework, pelas seguintes razões:

- A framework loopback fornece um conjunto de APIs para interagir com cada modelo, dando a possibilidade de fazer query e filter sobre as informações.
- Permite a criação de uma API RESTful de maneira muito rápida. Como mostram as figuras 10 e 11.

Figura 10

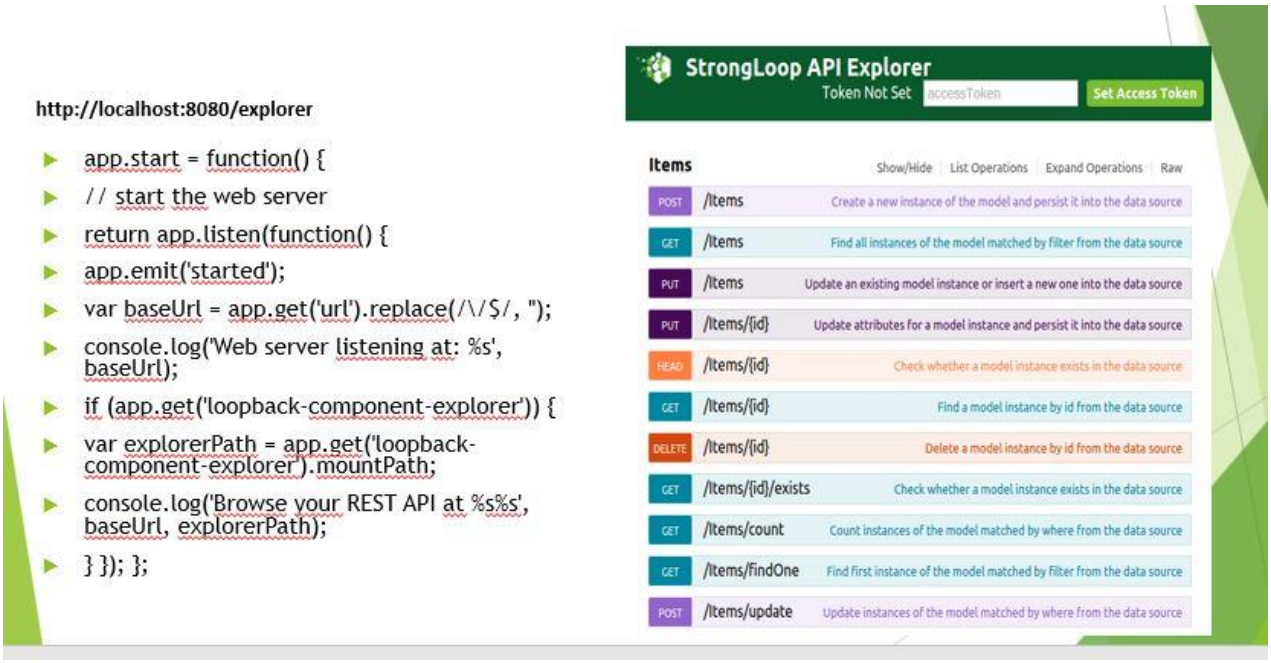

Figura 11

- O LoopBack é construído sobre o Express e vem com um conjunto de módulos do Node.js que podem ser usados independentemente e juntos para criar APIs REST para aplicativos clientes (qualquer coisa como mobile, browser, etc.) da maneira mais fácil.

- Os modelos são centro do LoopBack e representam bases de dados ou outros serviços de back-end (REST, SOAP e assim por diante). Os modelos LoopBack são objetos JavaScript com APIs Node e REST.

- O modelo base é o PersistedModel que permite as operações de CRUD, Quando criamos um modelo que deriva de PersistedModel, temos acesso à todas as operações de CRUD sobre o mesmo objecto.
- Alguns Modelos embutidos que o loopback fornece são:

    - Role- modelo onde se encontram todos os Role da aplicação.
    - RoleMapping- modelo onde se faz a associação entre o identificador do utilizador e o respetivo role.
    - Access Token – modelo que representa o token que o loopback atribui à um utilizador a quando da autenticação.

- Cada aplicação LoopBack tem um conjunto de modelos internos predefinidos, como User, Role e Aplication, não é preciso criar esses modelos comuns a partir do zero.

5.2. Base de Dados

Uma vez que o loopback oferece suporte para varias base de dados tanto documentais quanto relacionais, escolhemos dentre elas a *MongoDB*, pelas seguintes razões:

- É fácil escrever a logica da aplicação, pode-se passar os objetos de modelo diretamente para documento.

- Mecanismos de consulta e recursos de indexação muito poderosos que facilitam e agilizam a execução de várias consultas otimizadas diferentes.

- MongoDB facilita o desenvolvimento, já que suporta dados não estruturados nativamente e não requer migrações dispendiosas e demoradas quando os requisitos de aplicativos mudam.

- Os documentos do MongoDB são codificados em um formato semelhante ao JSON, chamado BSON, o que facilita o armazenamento, é um ajuste natural para metodologias modernas de programação orientada a objetos e também é leve, rápido e intercambiável.

- MongoDB suporta consultas ricas, distinguindo-os de outras bases de dados documental que dificultam consultas complexas.

5.3. Segurança dos dados

De modos a garantir a segurança dos dados, os dados do utente são guardados cifrados na base de dados, usamos o sistema simétrico de chaves, tendo em conta a Data encryption key (DEK) Key encryption key (KEK). Usamos os módulos npm UIDGenerator e KeyGenerator para gerar a DEK e a KEK, usamos o módulo npm CryptoJson para cifrar e decifrar os dados.

5.4. Microsoft Azure

5.4.1 Key Vault

No que toca ao armazenamento de chaves para cifrar dados, existem muitas soluções, umas mais seguras, outras mais baratas e menos seguras, dentre elas:

Usar um módulo de segurança de hardware externo.

- Guardar a chave de criptografia no hardware.
- Associar a chave ao login de administrador
- Guardar a chave em um servidor diferente.
- Guardar a chave em outro lugar no mesmo servidor
- Uma vez que a chave de cifra será partilhada entre o paciente e os profissionais que tiverem permissão para acede aos seus dados, resolvemos guardar a chave num servidor diferente, o KeyVault permite-nos encriptar chaves (DEK) e armazena-las (KEK) na cloud.

5.4.2 Storage Service (Blob)

Aproveitando dos serviços da Microsoft *Azure, usamos o Blob* para armazenar as imagens na cloud. O Blob service é usado para armazenar grandes quantidades de dados, dos quais texto ou dados binários.

O armazenamento de *blobs* é ideal para:

- Exibição de imagens ou documentos diretamente de um navegador.
- Streaming de vídeo e áudio.
- Escrevendo para arquivos de log.
- Armazenando dados para backup e restauração.

Aproveitamos duma das vantagens deste serviço para armazenar as imagens das fichas de utentes, em containers, um *container* organiza um conjunto de *blobs*. Todos os blobs encontram-se dentro de um *container*. Uma vez que um *storage service* pode conter um número ilimitado de *containers*, decidimos criar um *container* por cada ficha de utente, assim sempre que é adicionado uma foto à ficha de utente, a mesma é guardada num container.

Nota:

O Blob é um serviço, que contém Containers estes por sua vez contem vários blobs, é necessário haver distinção entre Blob como serviço e blobs como coleção de dados binários.

5.5 Client-Side

A aplicação client, foi construída em JavaScript puro, não recorremos à nenhuma framework, porque não houve necessidades, pois não houve complexidade no desenvolvimento e houve muita reutilização de código uma vez que os utilizadores têm funções muito parecidas, no entanto damos enfase ao módulo da parte MediaDevice e ao método escolhido para a validação dos dados recebidos via formulários.

5.5.1Captura de Foto Através dum Dermatoscópio.

Na inserção de fotos à ficha de utentes, as fotos podem ser adicionadas através dum dermatóscopio, como mostra a imagem.

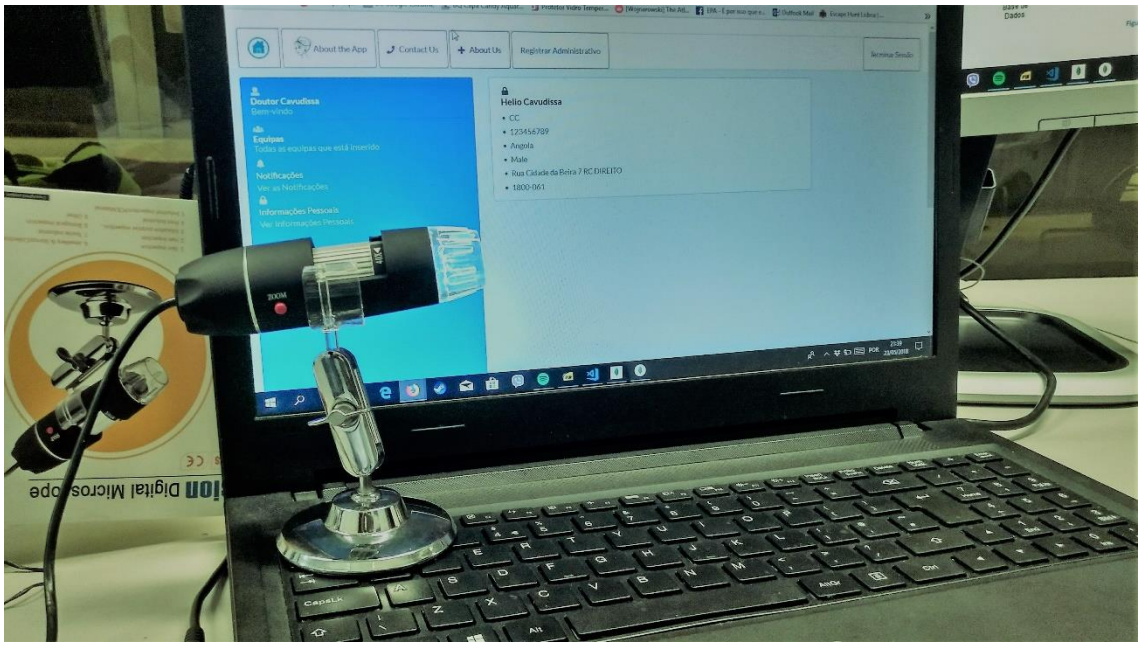

Figura 12

Na imagem acima temos um protótipo de um dermatoscopio, que ligado ao computador, podemos aceder à camara do mesmo através da API MediaDevice, uma vez capturada a imagem é convertida para base64 e de seguida é convertida para um ficheiro de formato png para ser enviada para o servidor.

5.5.2. Validação dos Dados dos Formulários.

Os dados enviados via formulários para o servidor, são recebidos pelo client-side e verificados, antes de serem reencaminhados para o servidor.

5.6. Representação dos Dados no Browser.

Usamos o HandleBars como viewEngine pelas seguintes razões:

Handlebars é um dos mais avançados (pré-compilação e afins), rico em recursos e popular de todos os mecanismos de templates JavaScript, e tem a comunidade ativa e vasta.

Handlebars é um mecanismo de criação de modelos sem lógica, o que significa que há pouca ou nenhuma lógica em seus modelos que estão na página HTML. O uso mais importante do Handlebars e de qualquer mecanismo de modelagem é manter suas páginas HTML simples, limpas e desacopladas dos arquivos JavaScript baseados em lógica, e o Handlebars atende bem a essa finalidade.

6. Aplicação em Funcionamento.

De modos a demonstrar a interação entre as tecnologias citadas acima, faremos algumas demonstrações na aplicação explicando o percurso e cada tecnologia utlizada.

6.1. Administrativo

Um administrativo como já foi referido neste documento, representa uma unidade hospitalar com as seguintes opções (ver menu figura 13).

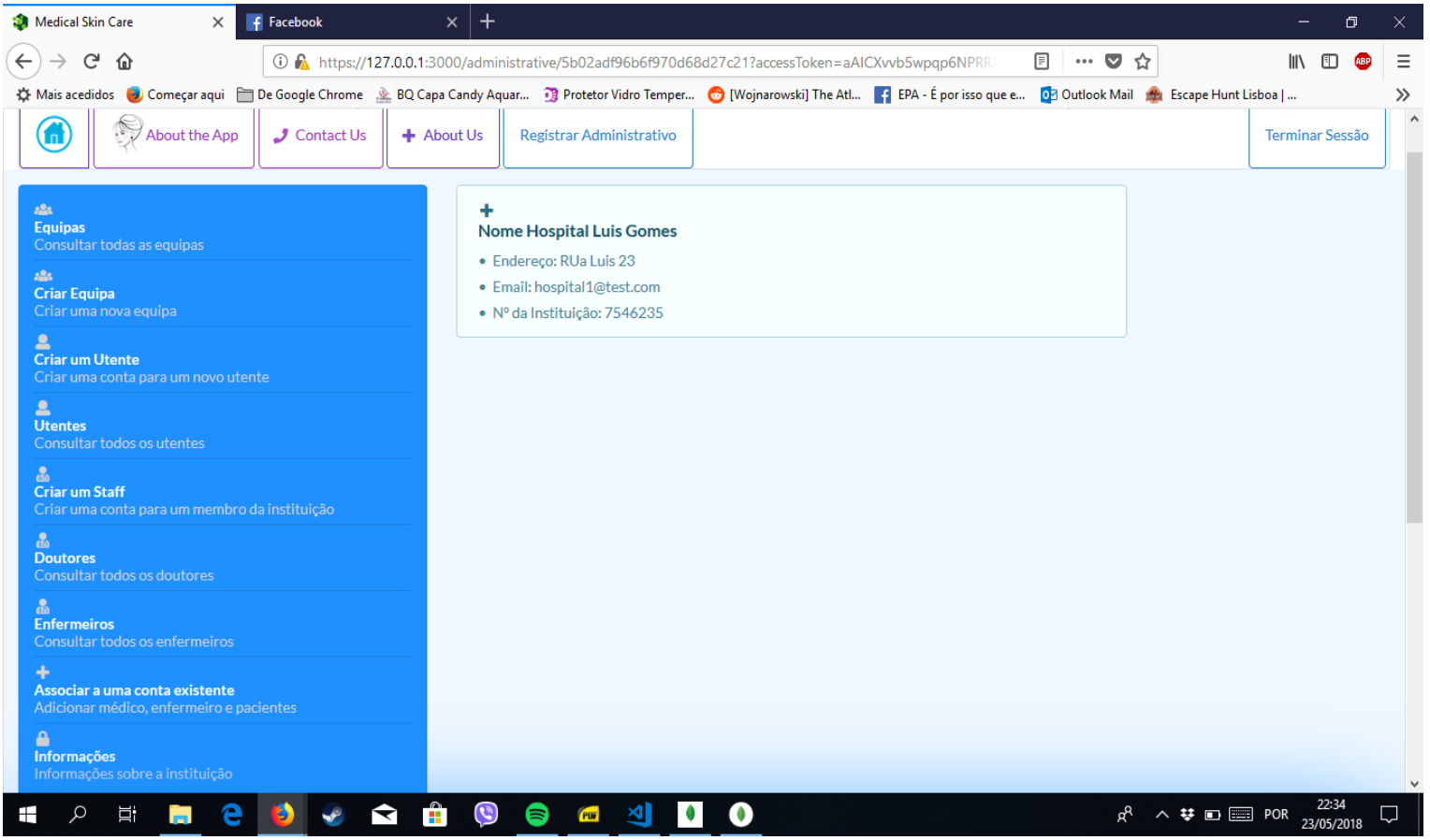

Figura 13

A primeira operação que faremos, será criar um utente, como mostra a figura a seguir.

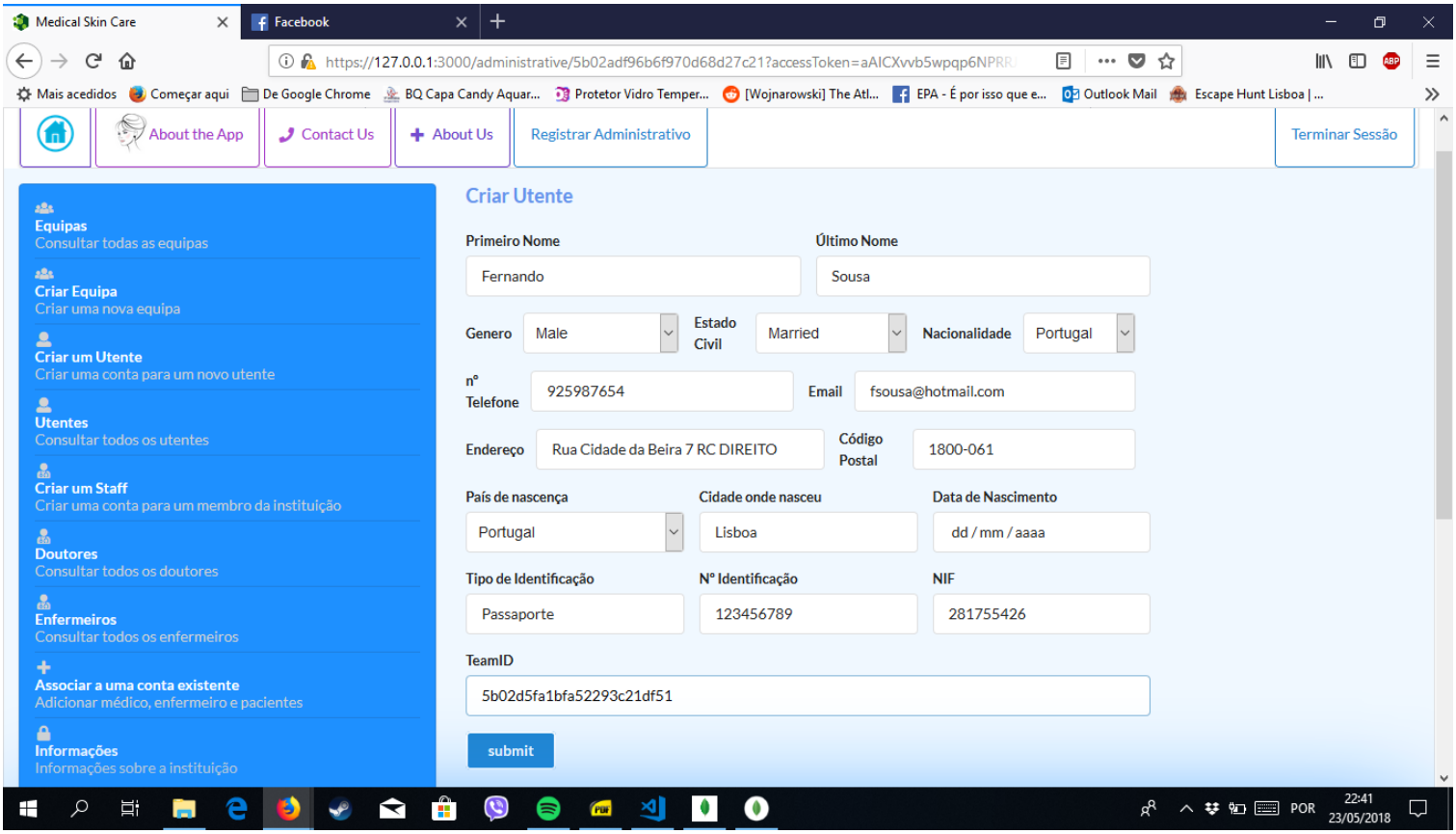

Figura 14

Como se pode ver na figura 14, quando um administrativo cria um utente, é necessário um email onde o mesmo receberá uma mensagem com a password de acesso à sua conta é também necessário anexar uma equipa médica à um utente no momento da criação (campo TeamID).

A seguir mostramos (figura 15) que um administrativo pode adicionar à sua instituição, um médico ou enfermeiro com registo existente na aplicação (registo feito por outra entidade hospitalar) , mostramos também ém que é possível pedir permissão de acesso às fichas do utente.

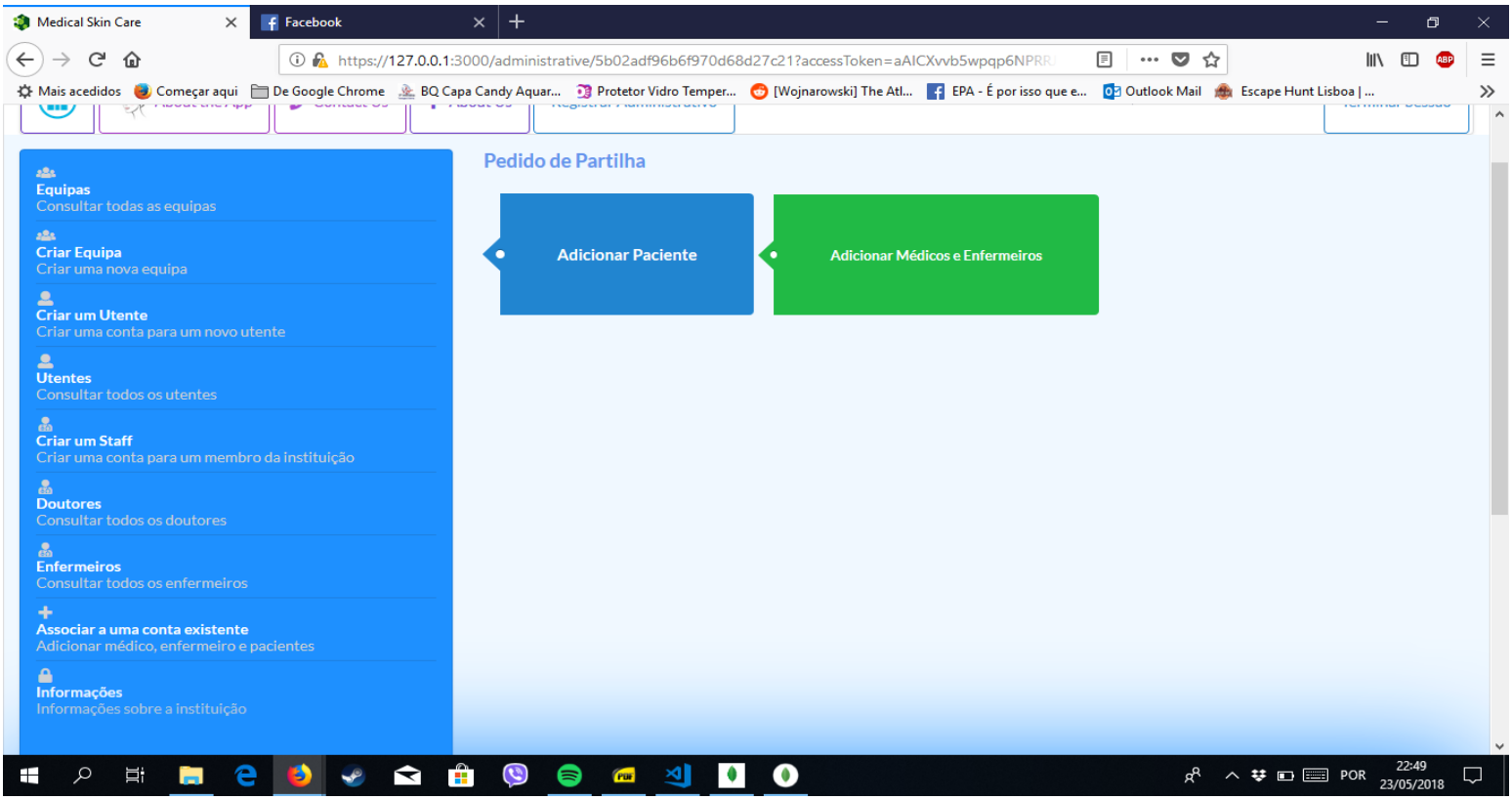

Figura 15

A opção adicionar um paciente permite-nos fazer um pedido ao paciente, como mostra a figura 16, é necessário o email onde o utente receberá a notificação e um ID da equipa que terá acesso aos dados do utente.

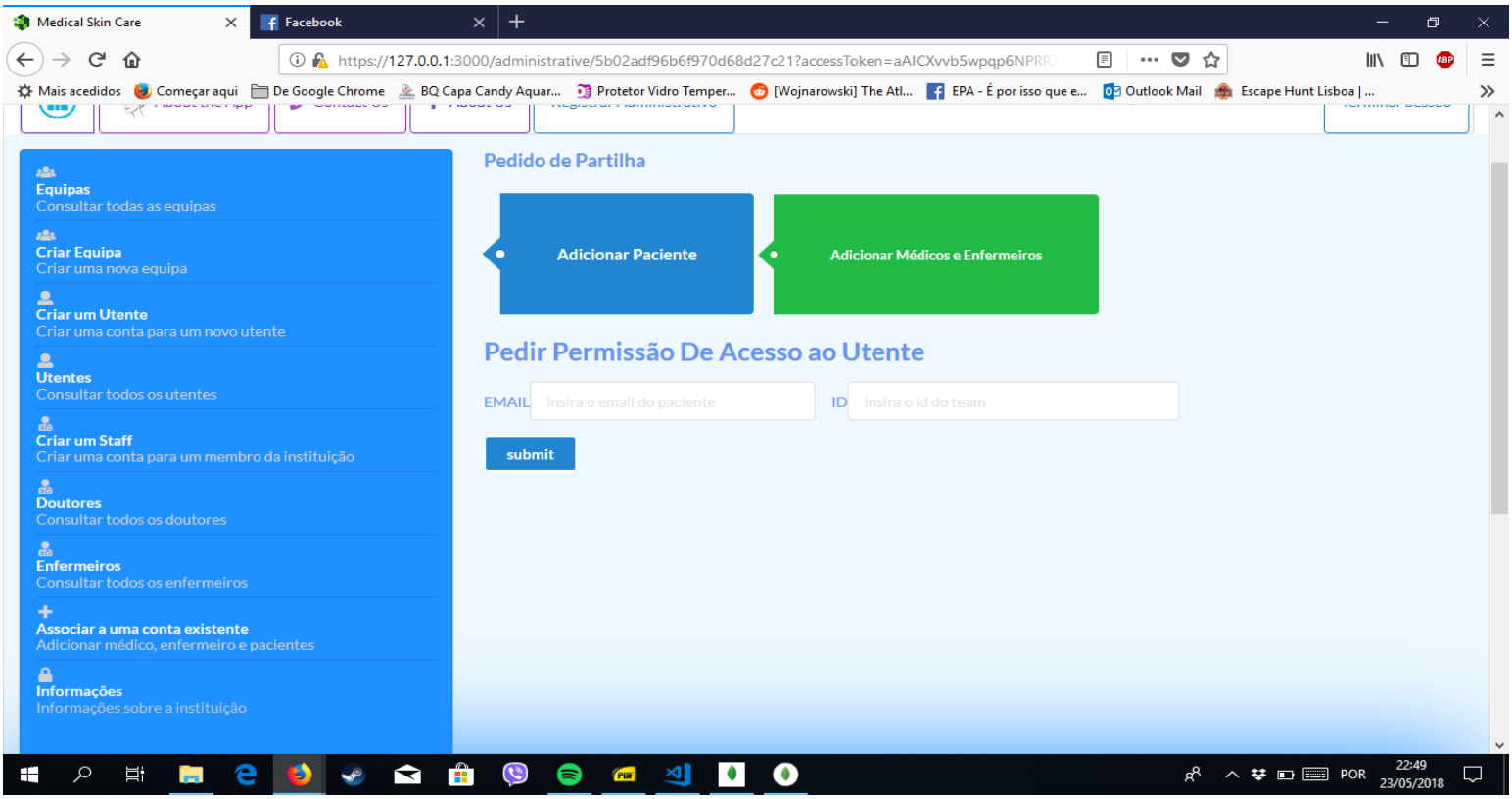

Figura 16

6.2. Utente

Um utente, como já foi referido, pode realizar as seguintes opções (ver menu figura 17)

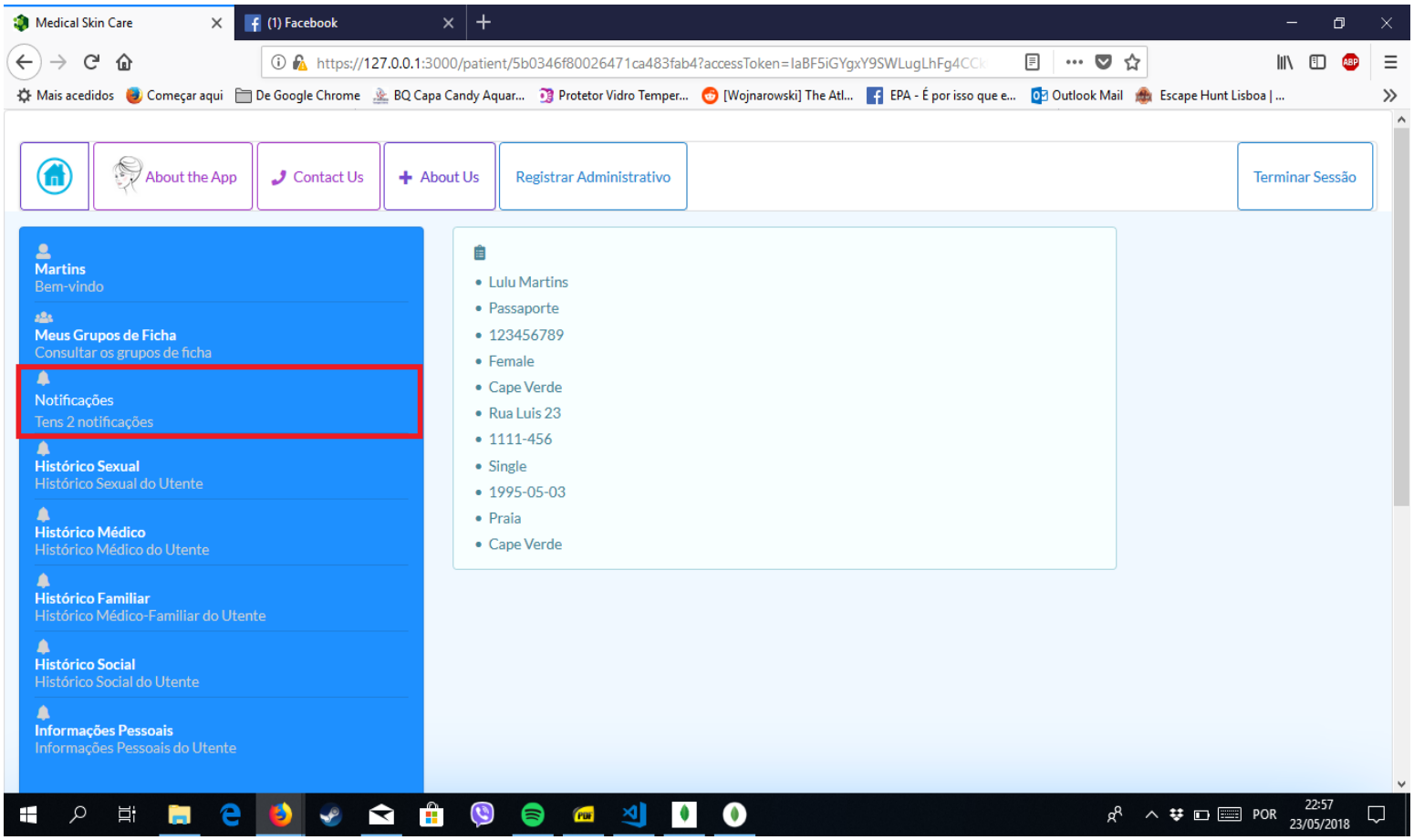

Figura 17

A figura 17 mostra o menu dum utente, com 2 notificações que são apresentadas na figura 18.

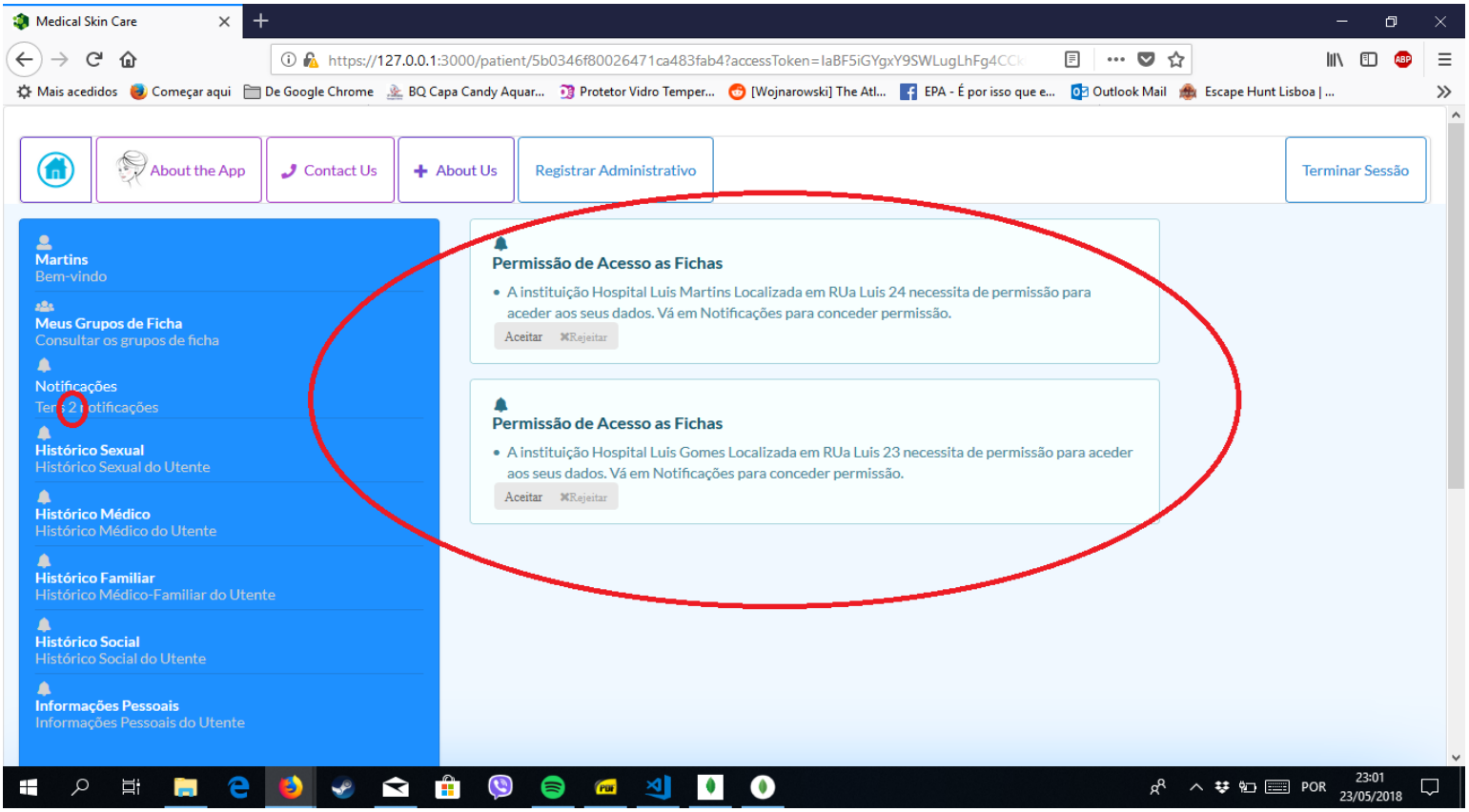

figura 18

As notificações presentes na página do utente, devem-se aos pedidos enviados pelo administrativo como mostra a figura 16.

6.2. Médicos e Enfermeiros

Os enfermeiros e médicos têm os mesmos acessos com algumas exceções em criação de fichas que apenas médicos podem criar.

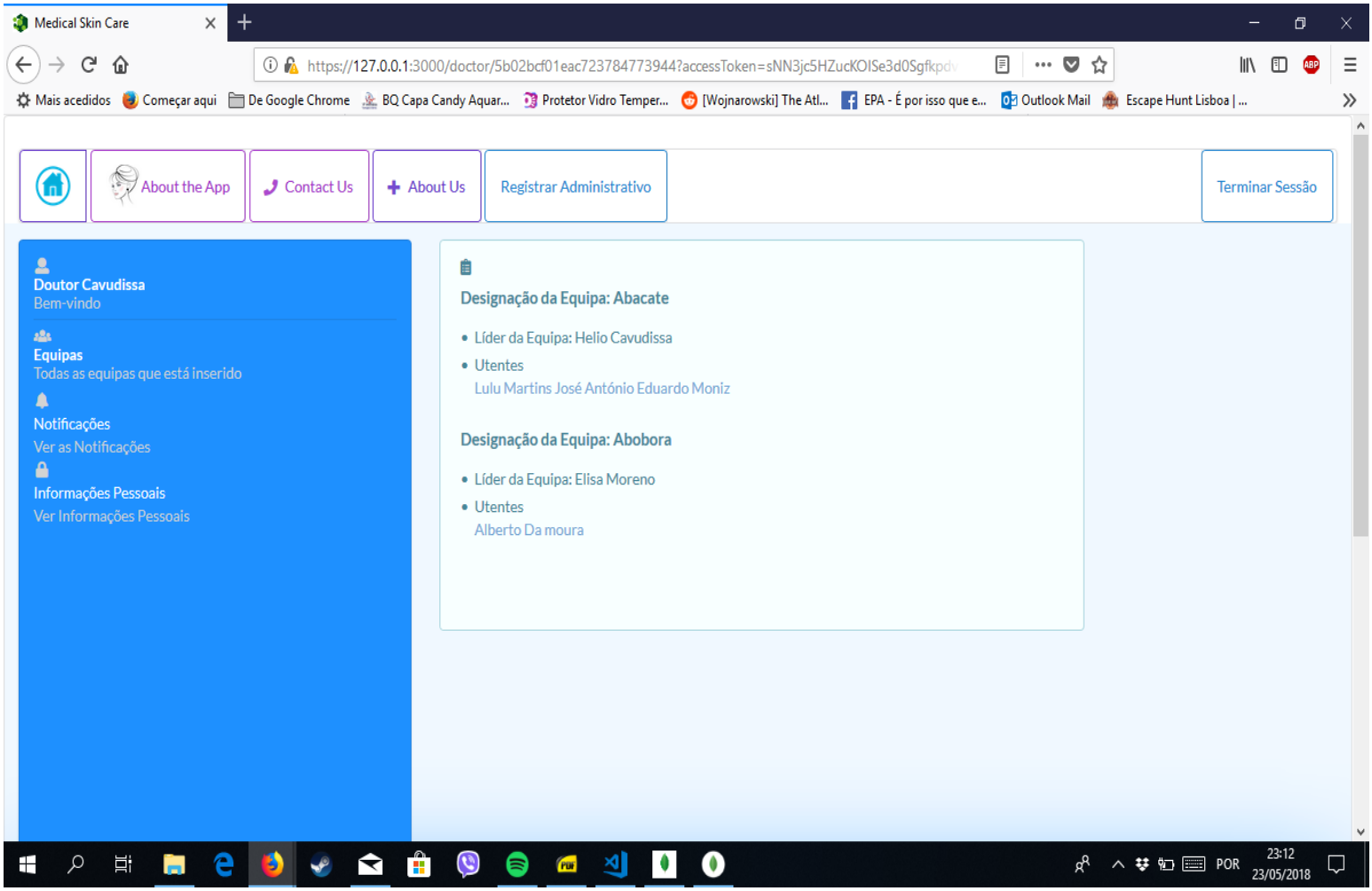

Figura 19

A figura 19, mostra o perfil dum médico, também podemos ver as suas equipas no lado direito, onde constam o team leader e os utentes.

Mostraremos a seguir na figura 20 que um médico pode consultar utentes das equipas em que está integrado e sobre os mesmos pode ler e alterar informações.

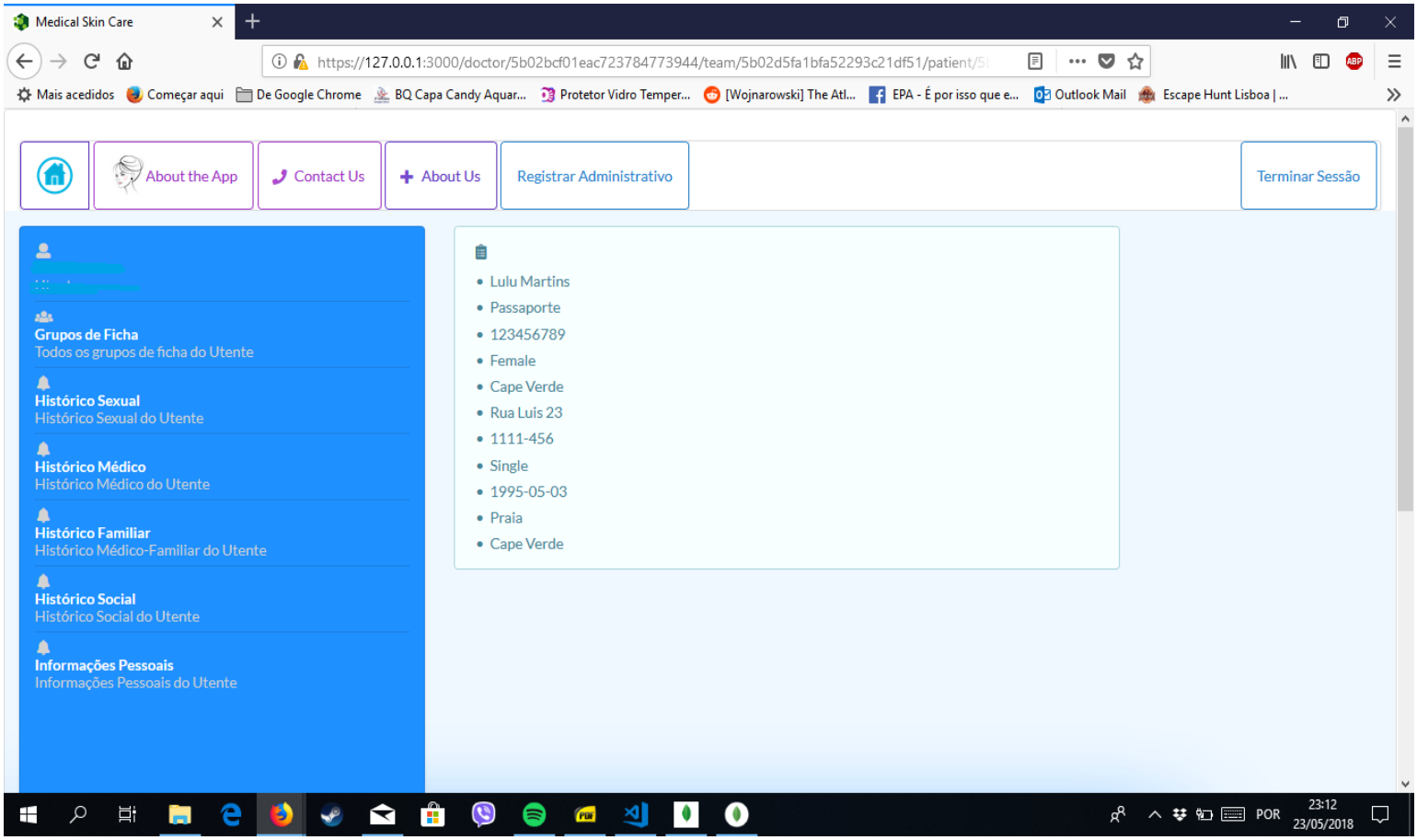

Figura 20

Pode consultar os grupos de fichas (agrupados por patologia), como mostra a figura 21

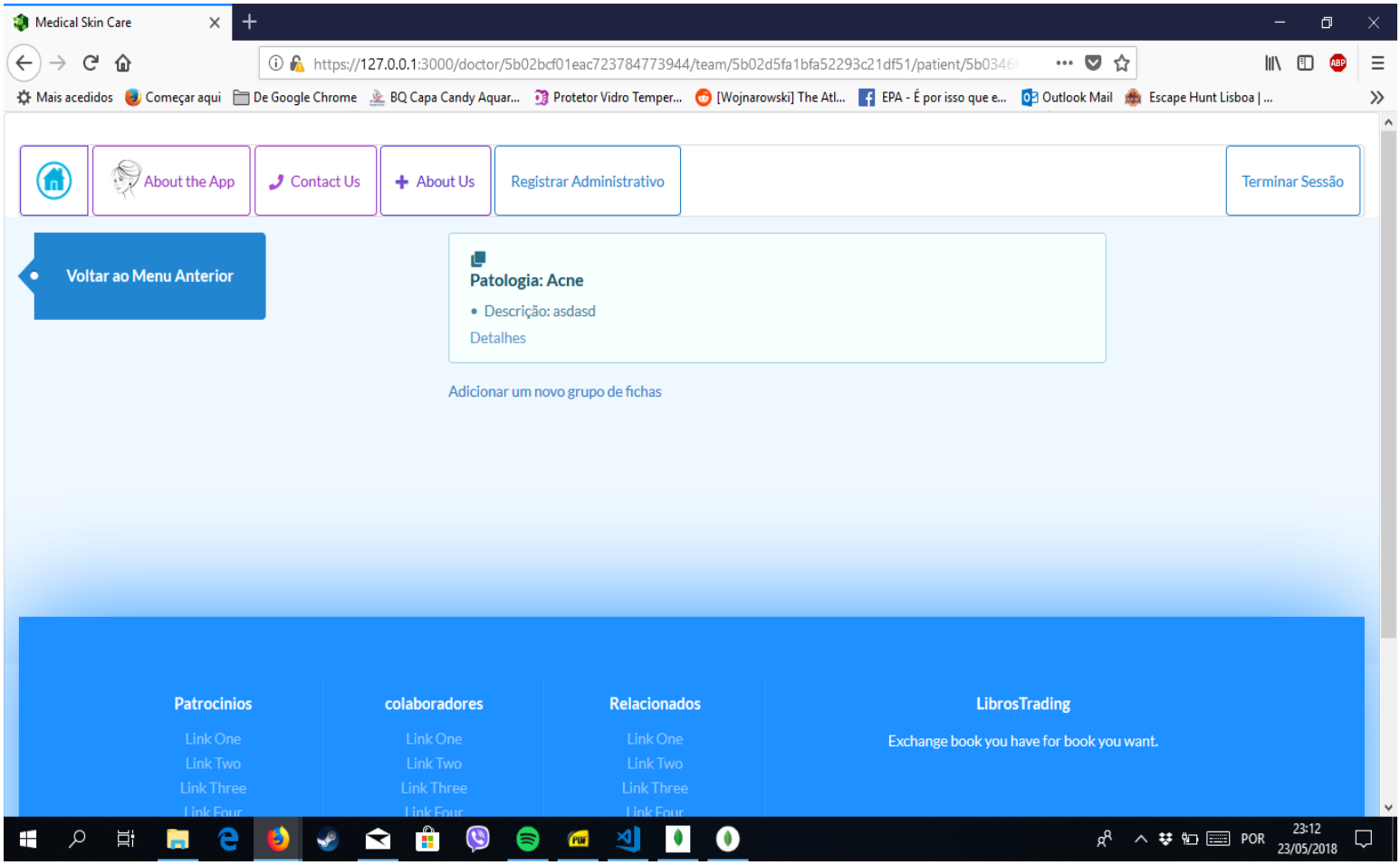

Figura 21

Figura 22 mostra as fichas existente dentro do grupo de fichas ordenadas por data.

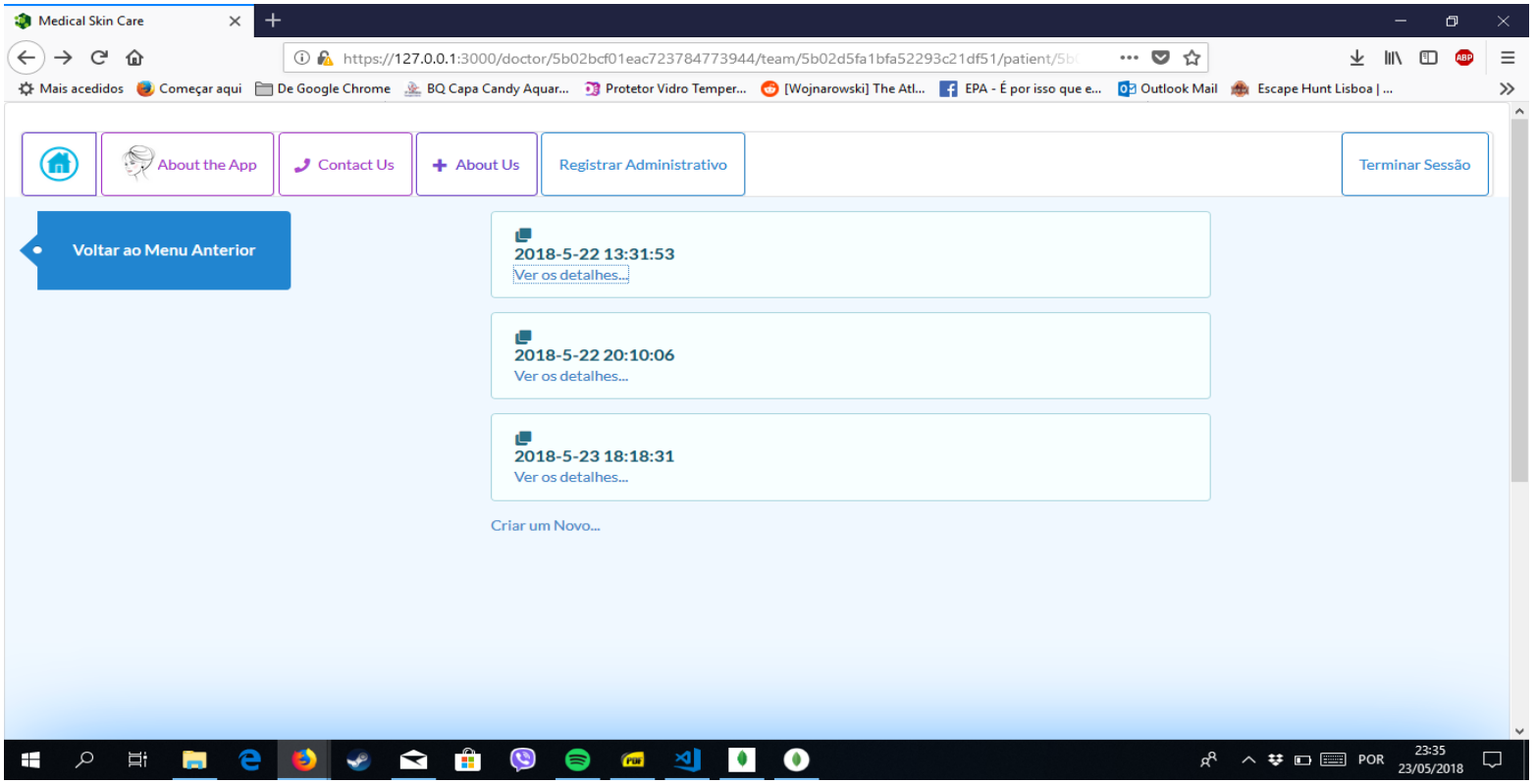

Figura 22

Figura 23 mostra as notas médicas duma ficha e também, a possibilidade de um médico a adicionar uma nota médica sobre à de um utente.

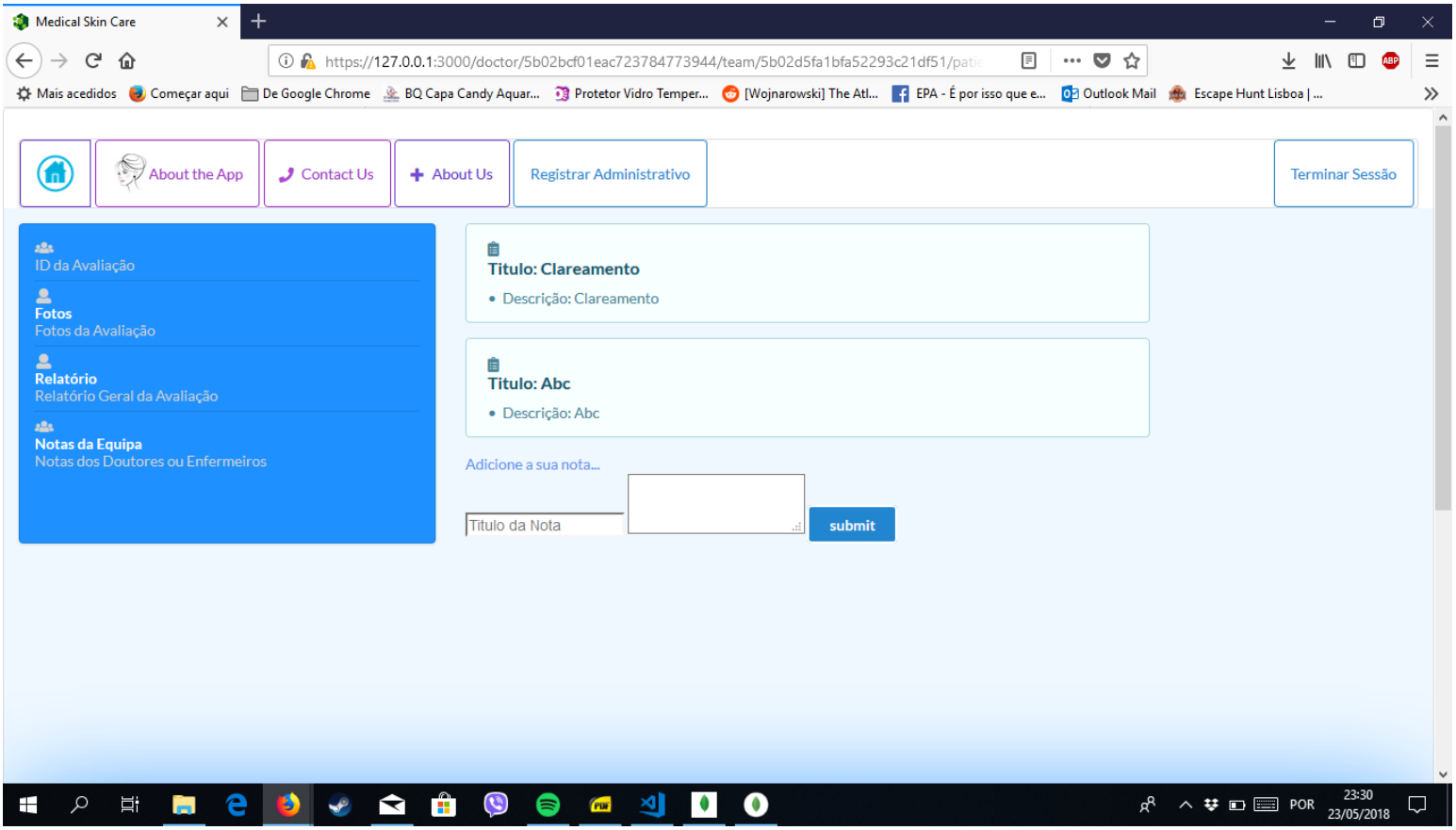

Figura 23

A figura 24 mostra as fotos duma ficha do utente.

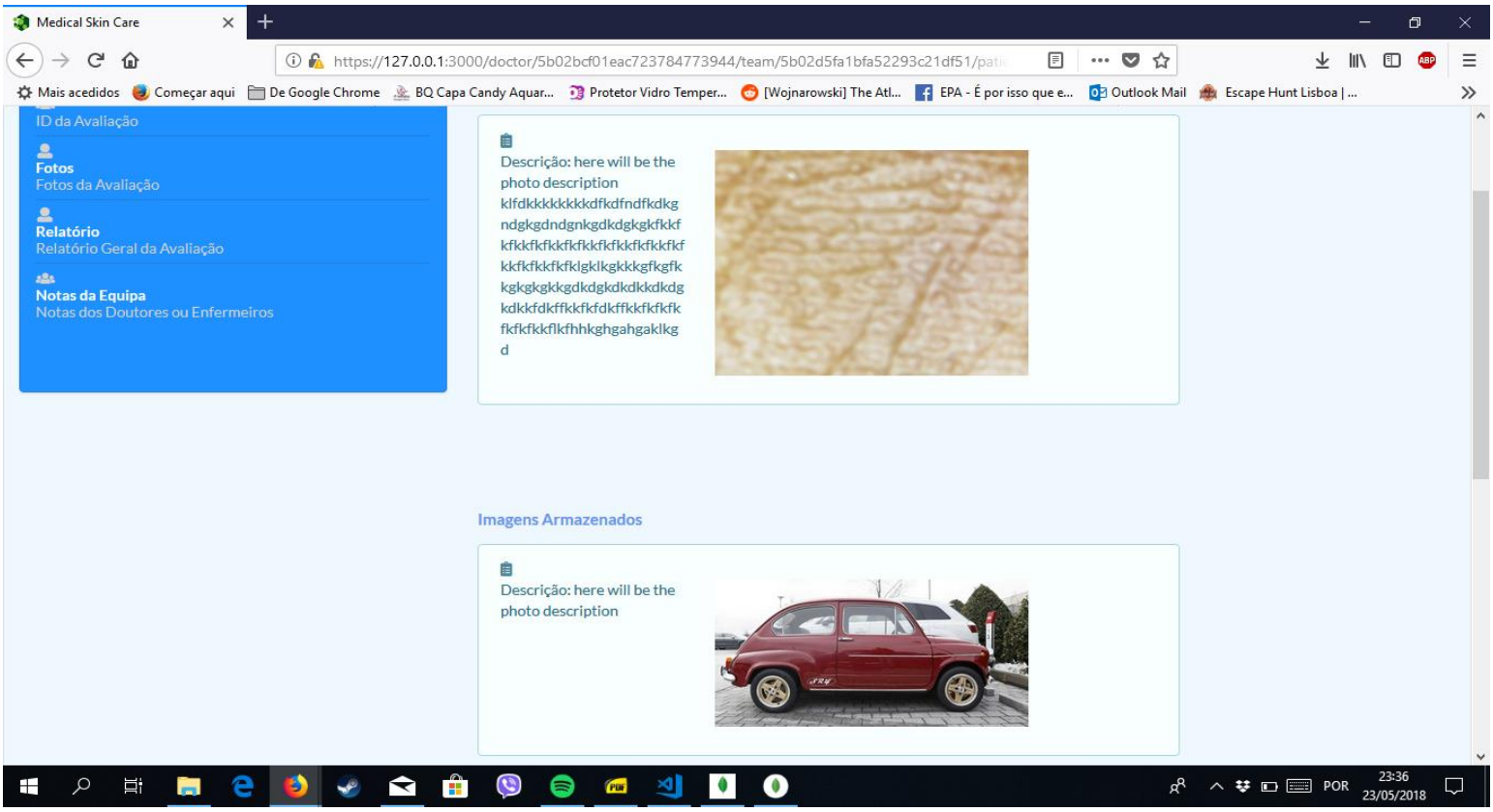

Figura 24

7. Calendarização

Data	Nº de semana	Tarefas
21/02/18 **28/02/18** **7/03/18** **14/03/18**	1 2 3 4	Discussão dos Detalhes, Pesquisa de Ferramentas, Aspetos Legais e Elaboração da Proposta de Projeto
21/03/18	5	Consulta com os Profissionais (Médicos e Enfermeiros) e Desenhos da Arquitetura do Sistema
28/03/18 **04/04/18** **11/04/18**	6 7 8	Desenvolvimento da Componente Servidora
18/04/18	9	Implementação da interface do Utilizador – Parte 1
25/04/18	10	Elaboração e Entrega do Relatório de Progresso
02/05/18 **09/05/18** **16/05/18**	11 12 13	Implementação da interface do Utilizador – Parte 2
23/05/18	14	Criação e Entrega do Cartaz e Relatório BETA
30/05/18 **06/06/18** **13/06/18** **20/06/18**	15 16 17 18	Conclusão das componentes Servidora e interface do utilizador
27/06/18	19	Correção de Erros e Melhorias
03/07/18 **14/07/18**	20 21	Conclusão do Relatório Final e Entrega da Versão Final

Marcos / Entregas	**Data Limite**
Proposta de Projeto	19 de março de 2018
Relatório de Progresso	30 de abril de 2018
Cartaz e Versão Beta	28 de maio de 2018
Entrega Versão Final	14 de julho de 2018

8. Conclusão

Nesta versão, apresentamos uma aplicação cumprindo os requisitos funcionais, ficando assim para fase final, detalhes não funcionais, melhorias nos aspectos visuais, melhorias no código da aplicação, como mostra a seção da calendarização

9. Bibliografia Consultadas

https://loopback.io/doc/index.html

https://nodejs.org/en/

https://www.npmjs.com/

https://handlebarsjs.com/

https://azure.microsoft.com/pt-br/

https://www.mongodb.com

F.Guerra Rodrigo e al., Dermatologia - Ficheiro Clinico (inclui CD-ROM) - 4.ª edição, Fundação Calouste Gulbenkian, junho de 2010, ISBN: 9789723113167

Kail, Robert V (2011), Children and Their Development (6th Edition) (Mydevelopmentlab Series), Englewood Cliffs, N.J: Prentice Hall, ISBN: 0205034942

Klaus Wolff, Richard Allen Johnson, Arturo Saavedra, Ellen K. Roh, Fitzpatrick's Color Atlas, 8th Ed, – 16 Mar 2017, McGrawHill Education, isbn: 9781259642197