universidade de aveiro

Escola Superior de Tecnologia e Gestão de Águeda

REABILITAÇÃO CARDIOVASCULAR

Projeto Temático em Desenvolvimento de Aplicações

Escola Superior de Tecnologia e Gestão de Águeda

Universidade de Aveiro

1º semestre

2017-2018

Grupo 4:

Daniel Martins	nº 83645
Luís Pinho	nº 83926
Pedro Martinho	nº 76480
Rafael Faustino	nº 80914
Rui Duarte	nº 29979

Escola Superior de Tecnologia e Gestão de Águeda

universidade de aveiro

Escola Superior de Tecnologia e Gestão de Águeda

REABILITAÇÃO CARDIOVASCULAR

Projeto Temático em Desenvolvimento de Aplicações

Escola Superior de Tecnologia e Gestão de Águeda

Universidade de Aveiro

1º semestre

2017-2018

Grupo 4:

Daniel Martins	nº 83645
Luís Pinho	nº 83926
Pedro Martinho	nº 76480
Rafael Faustino	nº 80914
Rui Duarte	nº 29979

Orientador: Joaquim Ferreira

Índice

Índice de tabelas

Índice de figuras

1 Introdução

1.1 Visão geral do sistema

No âmbito da unidade curricular Projeto Temático em Desenvolvimento de Aplicações, foram-nos apresentados diversos temas para escolher e desenvolver um deles, dos quais o nosso grupo escolheu o "Sistema de reabilitação cardiovascular", baseado em Activity Tracker e supervisão/acompanhamento médico.

Um hospital pretende adquirir um sistema com objetivo de gerir a reabilitação cardiovascular de diversos pacientes a partir de Activity tracking e supervisão médica, recolhendo dados médicos em tempo real do paciente: pressão arterial e frequência cardíaca, com recurso a uma pulseira usada pelo mesmo.

A aplicação funcionará com a utilização de uma pulseira que o utente irá utilizar. Esta pulseira irá medir a frequência cardíaca e pressão arterial. Em seguida, enviará os dados médicos para a base de dados que será consultada pelo médico. O médico terá acesso a todas as informações e poderá também criar notas informativas, aplicar tratamento, entre outras. O fisioterapeuta irá criar planos de treino especializados para o paciente com base nos tratamentos prescritos pelo médico. O utente terá apenas acesso a dados pessoais.

1.2 Cliente

O cliente deste sistema será um hospital com um departamento de reabilitação cardiovascular. Será usado especificamente por médicos, fisioterapeutas, pacientes e administradores do sistema dentro de cada hospital, cada um destes com diferentes papéis e permissões.

1.3 Objetivos

O principal objetivo deste sistema é fornecer maior facilidade e fiabilidade no acompanhamento fora do hospital a pacientes que estejam a praticar um programa de reabilitação cardiovascular.

Para além disto, o sistema irá permitir uma melhor comunicação entre os vários elementos da equipa médica, agilizar e encurtar tempos de espera de comunicação entre os pacientes e profissionais de saúde e melhorar a organização e acesso da informação pessoal relativa ao paciente.

A implementação deste sistema irá também permitir a redução de custos em recursos humanos, dado que muitos dos assuntos tratados entre o médico, fisioterapeuta e paciente poderão ser tratados digitalmente através da troca de mensagens e partilha de planos de treino, o que irá permitir ter menos custos a médio-longo prazo em relação a outras instituições hospitalares sem este sistema.

1.4 Repositório Git

De forma a manter sob controlo e registadas todas as contribuições e alterações deste projeto, foi criado um repositório remoto *Git* na plataforma *BitBucket*. Foi dado o acesso ao repositório a cada um dos membros do grupo assim como ao orientador do projeto. Ao longo do tempo, vários membros foram contribuindo e fazendo alterações a este repositório, pelo que é possível visualizar uma lista de todos os *commits* feitos por cada um dos membros desde o inicio do projeto até agora (disponível nos anexos).

2 Planeamento

2.1 Atividades e Tarefas

Nas tabelas 1 e 2 abaixo referidas encontram-se descritas as tarefas e atividades planeadas para o projeto em causa, assim como a respetiva duração, descrição e dependências. A duração das mesmas é expressa em semanas.

Identificação	Atividades/Tarefas	Duração (semanas)	Dependências
A1	Gestão do projeto	16	
A2	Descrever / Identificar requisitos do sistema	2	
T2.1	Descrição geral e do propósito do sistema	1	
T2.2	Definir requisitos	1	
A3	Modelação do projeto	3	A2
T3.1	Criação do Diagrama de Casos de Uso	1	
T3.2	Criação do Diagrama de Classes	1	T3.1
T3.3	Criação do Diagrama de Atividades	1	T3.2
A4	Implementação	10	A3
T4.1	Base de dados	7	
T4.2	Comunicação com a Base de dados	7	
T4.3	Interface	7	
A5	Instalação	1	A4

Tabela 1 - Atividades e tarefas planeadas

Identificação	Descrição
A1	Atividade que decorre desde o início até ao final do projeto. Consiste em definir e gerir as diferentes tarefas ao longo do projeto (calendarização de tarefas, cálculo de esforço e análise de riscos)
A2	Identificação/descrição do propósito do sistema e definição dos requisitos
T2.1	Descrição do objetivo e propósito do sistema (funcionamento geral) e do que incentivou à sua criação.
T2.2	Definição dos requisitos funcionais e não funcionais dos requisitos do projeto
A3	Atividade que consiste na modelação do sistema
T3.1	Criação de casos do diagrama de casos de uso
T3.2	Criação do diagrama de classes com base no diagrama de casos de uso
T3.3	Criação do diagrama de atividades referente ao sistema inteiro
A4	Atividade que consiste na implementação prática do projeto em termos da base de dados e das várias interfaces gráficas.
T4.1	Modelação e criação da base dados do sistema
T4.2	Criação do software responsável por estabelecer a comunicação entre as interfaces gráficas e a base de dados.
T4.3	Processo de criação das várias interfaces para diferentes utilizadores (médico, fisioterapeuta, paciente e administrador)
A5	Implementação do projeto depois de completamente criado e testado

Tabela 2 - Descrição das atividades e tarefas

2.2 Calendarização

Na tabela 3 que se segue encontra-se indicado a calendarização de cada uma das atividades, com a data de início planeada e sua respetiva duração em semanas.

Logo abaixo, a tabela 4 mostra o diagrama de Gantt com a distribuição das tarefas pelo tempo disponível com relação a duração do projeto.

Identificação	Início Planeado	Duração
A1	25/09/17	16
A2	25/09/17	2
T2.1	25/09/17	1
T2.2	02/10/17	1
A3	09/10/17	3
T3.1	09/10/17	1
T3.2	16/10/17	1
T3.3	23/10/17	1
A4	30/10/17	10
T4.1	30/10/17	7
T4.2	13/11/17	7
T4.3	20/11/17	7
A5	08/01/18	1

Tabela 3 - Calendarização das atividades e tarefas

Data Início / Tarefa	25-09-17	02-10-17	09-10-17	16-10-17	23-10-17	30-10-17	06-11-17	13-11-17	20-11-17	27-11-17	04-12-17	11-12-17	18-12-17	25-12-17	01-01-18	08-01-18
A1																
A2																
T2.1																
T2.2																
A3																
T3.1																
T3.2																
T3.3																
A4																
T4.1																
T4.2																
T4.3																
A5																
1º Entregável – Relatório intermédio – 30/10/2017																
2º Entregável – Relatório final + Aplicação final – 17/01/2018																

Tabela 4 - Diagrama de Gantt

2.3 Participação nas tarefas

Na tabela 5 encontra-se representado a participação de cada um dos elementos do grupo em cada tarefa.

	Daniel Martins	Luis Pinho	Pedro Martinho	Rafael Faustino	Rui Duarte	TOTAL
A1	20%	20%	20%	20%	20%	100%
A2	0%	25%	25%	25%	25%	100%
T2.1	0%	50%	0%	50%	0%	100%
T2.2	0%	0%	50%	0%	50%	100%
A3	30%	7%	13%	0%	50%	100%
T3.1	0%	10%	10%	0%	80%	100%
T3.2	0%	10%	30%	0%	60%	100%
T3.3	90%	0%	0%	0%	10%	100%
A4	0%	33%	33%	0%	33%	100%
T4.1	0%	5%	5%	0%	90%	100%
T4.2	0%	5%	90%	0%	5%	100%
T4.3	0%	90%	5%	0%	5%	100%
A5	0%	33%	33%	0%	33%	100%

Tabela 5 - Participação em cada tarefa

2.4 Esforço previsto

A tabela 6 apresenta o esforço previsto, calculado através de métricas orientadas à função.

Este método de medição indireto tem como vantagens os seguintes pontos:

- o facto de ser independente da linguagem de programação;
- Usa atributos contáveis e que são fixados bastante cedo no processo de desenvolvimento do software;
- Não penaliza implementações "inventivas" (e curtas…) que usam menos linhas de código que outros possíveis alternativos mais complexos;
- Torna mais fácil a medida do impacto do uso de componentes reutilizáveis.

Componente/Complexidade	Baixa (1)	Média (3)	Alta (5)	Soma	Total
Nº de interfaces	1	1	3	5	19
Nº de ficheiros	5	5	10	20	70
Nº de queries	15	10	5	30	70
				Total	159

Tabela 6 - Cálculo do esforço através de métricas orientadas à função

2.5 Análise de riscos

A tabela 7 mostra os riscos a que este projeto está exposto bem como a categoria de cada risco, a probabilidade de ocorrer e o impacto que tem no projeto caso este aconteça.

Risco	Categoria	Probabilidade	Impacto
Incumprimento dos prazos de trabalho	Projecto	1,00%	Crítico
Incumprimento dos requisitos propostos pelo cliente	Projecto	5,00%	Crítico
Desistencia do único cliente	Negócio	10,00%	Crítico
Suborçamentação	Negócio	20,00%	Crítico
Alteração de requisistos	Projecto	20,00%	Marginal
Sub-estimativa do esforço	Projecto	40,00%	Marginal

Tabela 7 - Análise de riscos

2.6 Redução de riscos

A tabela 8 mostra os métodos utilizados para reduzir a probabilidade dos riscos da tabela 7 ocorrerem, bem como reduzir o impacto desses mesmos riscos caso eles ocorram.

Risco	Reduzir Probabilidade	Reduzir Impacto
Incumprimento dos prazos de trabalho	Destribuir adequadamente as tarefas do projecto pela equipa.	Colaboração entre a equipa.
Incumprimento dos requisitos propostos pelo cliente	Incluir o cliente na equipa de desenvolvimento.	
Desistencia do único cliente	Aumentar a comunicação com o cliente, informando-o do estado do	Encontrar outro potencial cliente.
Suborçamentação	Controlar o desenvolvimento da aplicação.	Reavaliar os custos do projecto.
Alteração de	Incluir o cliente na equipa de	
Sub-estimativa do esforço	Reavaliar frequentemente o esforço de todos os membros da equipa.	Controlar intensivamente o processo de desenvolvimento.

Tabela 8 - Redução de riscos

3 Modelo de requisitos

3.1 Requisitos funcionais

As tabelas 9 e 10 apresentam os requisitos funcionais e não funcionais identificados, assim como a sua prioridade.

Referência	Requisito funcional	Prioridade
RF1	O administrador do sistema deve adicionar e editar utilizadores.	Alta
RF2	O médico tem que receitar o tratamento de cada paciente.	Alta
RF3	O fisioterapeuta tem que criar o plano de treino de cada um dos seus pacientes.	Alta
RF4	O paciente deve ter acesso ao seu respetivo plano de treinos.	Alta
RF5	O médico e o fisioterapeuta devem ter acesso ao tratamento e aos planos de treinos.	Alta
RF6	O médico e o fisioterapeuta devem ter acesso aos dados pessoais dos seus pacientes.	Alta
RF7	O médico e o fisioterapeuta devem poder adicionar notas sobre cada paciente.	Média
RF8	O médico e o fisioterapeuta devem ter acesso a uma lista dos alertas que ocorreram num período de tempo.	Média
RF9	O sistema deve guardar o histórico de dados do paciente.	Média
RF10	O paciente deve ter acesso ao seu histórico de utilização.	Média
RF11	O médico e o fisioterapeuta devem ter a capacidade de verificar dados estatísticos sobre todos ou sobre cada um dos seus pacientes.	Baixa
RF12	O sistema deve permitir a autenticação dos utilizadores na aplicação.	Baixa
RF13	O sistema deve ter permissões específicas para cada grupo de utilizadores.	Baixa

Tabela 9 - Requisitos funcionais

3.2 Requisitos Não Funcionais

Referência	Requisito não funcional	Prioridade
RNF1	O telemóvel deve ter Bluetooth.	Alta
RNF2	A pulseira deve comunicar em tempo real com o telemóvel através de Bluetooth.	Alta
RNF3	O telemóvel deve ter acesso à internet.	Alta
RNF4	O software utilizado pelo médico deve ser compatível com Windows, macOS e Linux.	Média
RNF5	O software utilizado pelo fisioterapeuta deve ser compatível com Windows, macOS e Linux.	Média
RNF6	O software utilizado pelo paciente deve ser compatível com Android e IOS.	Média
RNF7	O software deve ter uma interface simples e intuitiva para os seus utilizadores.	Baixa
RNF8	Os dados registados pela pulseira devem ocupar pouco espaço de memória.	Baixa
RNF9	Os dados dos utilizadores devem estar encriptados.	Baixa

Tabela 10 - Requisitos não funcionais

4 Modelo de casos de Utilização

4.1 Visão geral

O diagrama de casos de uso abaixo (Figura 1) descreve a funcionalidade proposta para o sistema a implementar, facilitando o levantamento dos requisitos funcionais do sistema.

Um caso de uso é uma ação que um ator efetua e que interage com o sistema.

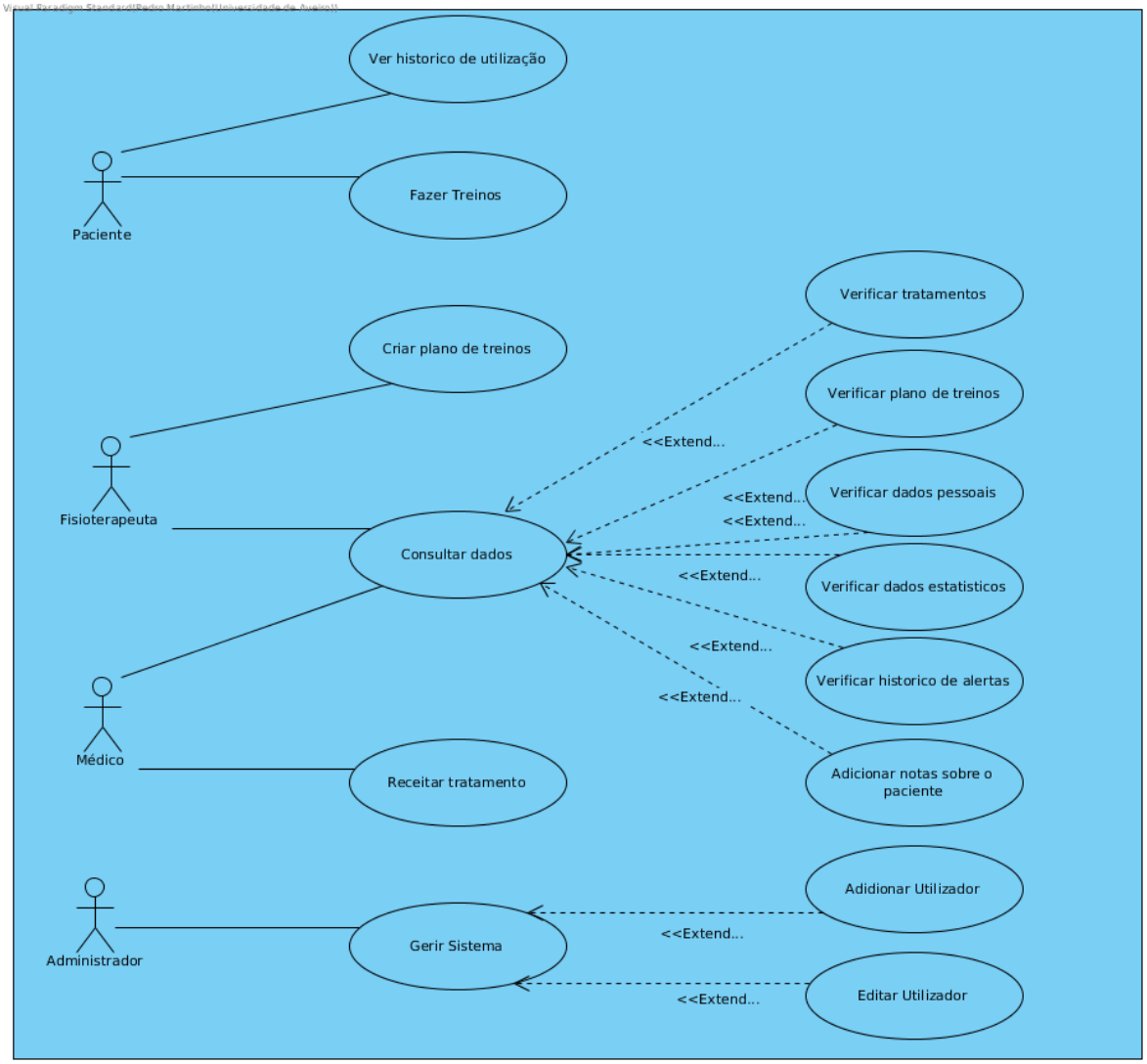

Figura 1 - Diagrama de casos de uso

4.2 Atores

Ator	Descrição
Paciente	Visualiza e realiza os treinos prescritos pelo fisioterapeuta e acede a históricos de utilização resumidos
Fisioterapeuta	Consulta dados relativos ao paciente e cria planos de treino em função do tratamento receitado pelo médico
Médico	Receita tratamentos ao paciente e consulta dados relativos ao mesmo
Administrador	Responsável pela manutenção do sistema

4.3 Descrição dos casos de utilização

Nome:	**Ver Histórico de utilização**
Atores:	Paciente
Finalidade:	Acesso ao seu histórico de uso
Requisitos funcionais:	RF9, RF10
Pré-condições:	Tem de estar autenticado no sistema e ter permissões para efetuar a operação
Sumário:	O ator pode aceder a um histórico dos seus treinos e a breves informações acerca do seu estado cardiovascular no seu dispositivo móvel

Nome:	**Fazer Treinos**
Atores:	Paciente
Finalidade:	Receber informação sobre o treino e efetuar o treino definido
Requisitos funcionais:	RF3, RF4
Pré-condições:	Tem de estar autenticado no sistema e ter permissões para efetuar a operação O plano de treinos tem de existir e estar associado ao ator
Sumário:	O ator visualiza o treino definido pelo fisioterapeuta no seu dispositivo móvel e efetua-o

Nome:	**Criar plano de treinos**
Atores:	Fisioterapeuta
Finalidade:	Criar um plano de treinos
Requisitos funcionais:	RF2, RF3
Pré-condições:	Tem de estar autenticado no sistema e permissões para efetuar a operação Tem de existir um tratamento prescrito pelo médico
Sumário:	O ator insere no sistema um plano de treinos associado a um paciente. Esse plano de treinos está dependente do tratamento prescrito pelo médico e pode sofrer alterações ao longo do tempo

Nome:	**Consultar dados**
Atores:	Fisioterapeuta, Médico
Finalidade:	Consultar vários tipos de dados referentes ao paciente e respetivo plano de treinos e tratamento associado
Requisitos funcionais:	RF4, RF5, RF6, RF7, RF8, RF11
Pré-condições:	Tem de estar autenticado no sistema e ter permissões para efetuar a operação
Sumário:	O ator consulta vários tipos de dados relacionados com o paciente tais como dados pessoais, o tratamento prescrito, o plano de treinos e seu histórico, vários dados estatísticos e acede aos alertas gerados pelo sistema de forma a poder otimizar o tratamento e subsequente plano de treinos Pode ainda criar notas relativas ao paciente

Nome:	**Receitar tratamento**
Atores:	Médico
Finalidade:	Prescrever um tratamento médico adequado ao paciente
Requisitos funcionais:	RF2
Pré-condições:	Tem de estar autenticado no sistema e ter permissões para efetuar a operação O paciente em questão (ao qual será prescrito o tratamento) tem de estar registado no sistema
Sumário:	O médico prescreve um tratamento médico adequado ao paciente com base no seu histórico clínico

Nome:	**Gerir Sistema**
Atores:	Administrador
Finalidade:	Criar e editar utilizadores e suas respetivas permissões
Requisitos funcionais:	RF1
Pré-condições:	Tem de estar autenticado no sistema para efetuar as operações ao próprio sistema
Sumário:	Gere os utilizadores com diferentes funções e permite efetuar a recuperação de dados de autenticação de sistema

5 Diagrama de classes

A figura 2 ilustra o diagrama de classes, mostrando os atributos e a sua relação.

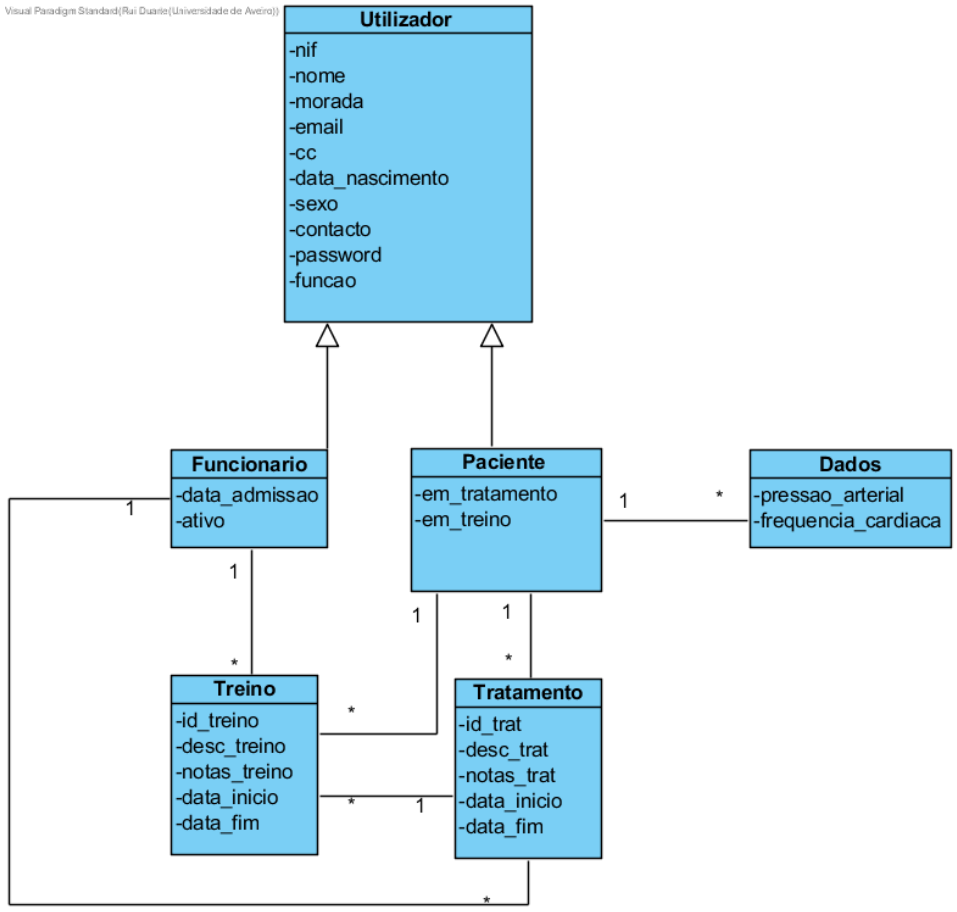

Figura 2 - Diagrama de Classes

6 Diagrama de atividades

A figura 3 apresenta o diagrama de atividades geral do sistema onde é descrito o comportamento do mesmo e respetivas exceções.

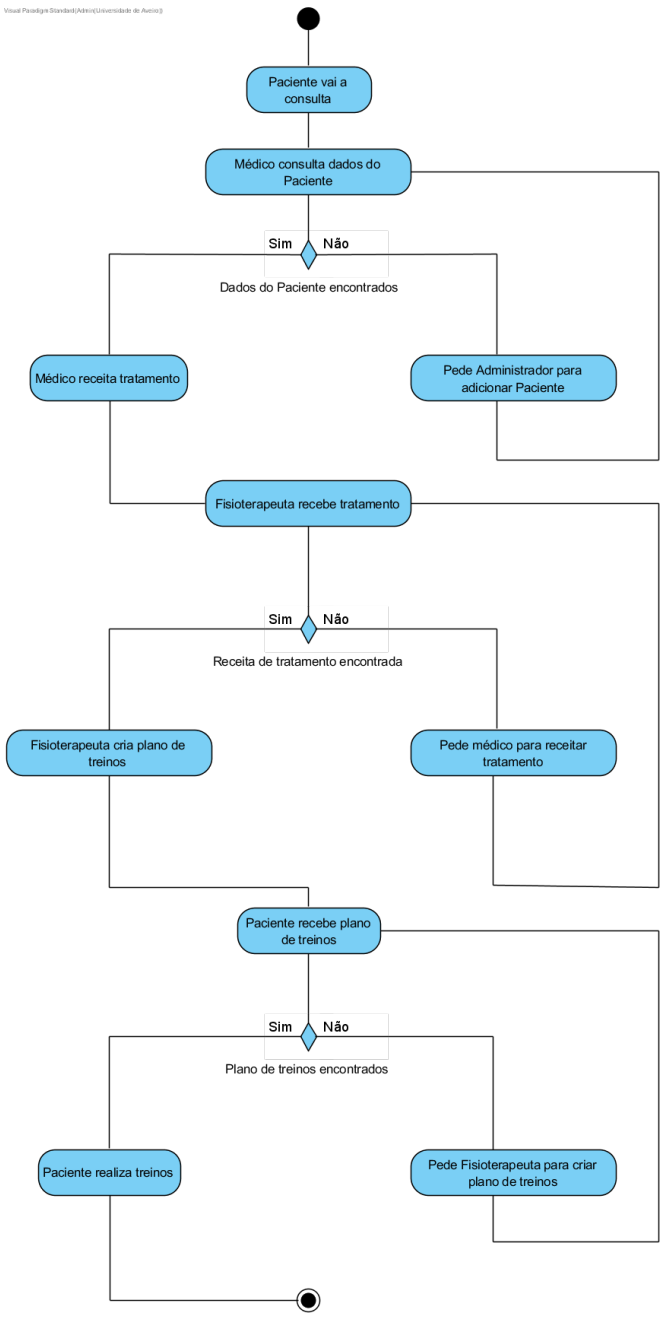

Figura 3 - Diagrama de atividades

7 Implementação

7.1 Modelo de dados persistente

Após a análise de requisitos e criação do diagrama de casos de uso e subsequente diagrama de classes, procedeu-se à criação do modelo lógico e físico, apresentado nas seguintes páginas.

7.1.1 Modelo Lógico

Segue o modelo lógico, representado pelo esquema relacional.

utilizador (id_utilizador (PK), password, nome, morada, cod_postal, localidade, nif, cc, sexo, data_nascimento, contacto, mail, função)

paciente (id_paciente (PK), id_utilizador (FK), em_tratamento, em_treino)

funcionário (id_funcionario (PK), id_utilizador(FK), data_admissao, activo)

tratamento (id_tratamento (PK), desc_tratamento, notas_tratamento, data_inicio, data_fim, paciente (FK), medico(FK))

treino (id_treino(PK), desc_treino, notas_treino, data_inicio, data_fim, fisioterapeuta(FK), tratamento(FK))

dados (paciente(PK), data(PK), hora(PK), pressao_arterial_min, pressao_arterial_min, freq_cardiaca)

alertas (id(PK), tipo, paciente(FK), data(FK), hora(FK))

paciente_alertas (id_paciente(PK), min_pressao_arterial_max, max_pressao_arterial_max, min_pressao_arterial_min, max_pressao_arterial_min, min_freq_cardiaca, max_freq_cardiaca)

logs_utilizador (id(PK), tipo, data_logs, utilizador, id_utilizador, password, nome, morada, cod_postal, localidade, nif, cc, sexo, data_nascimento, contacto, mail, função)

logs_paciente (id(PK), tipo, data_logs, utilizador, id_paciente, id_utilizador, em_tratamento, em_treino)

logs_funcionário (id(PK), tipo, data_logs, utilizador, id_funcionario, id_utilizador, data_admissao, activo)

logs_tratamento (id(PK), tipo, data_logs, utilizador, id_tratamento, desc_tratamento, notas_tratamento, data_inicio, data_fim, paciente, medico)

logs_treino (id(PK), tipo, data_logs, utilizador, id_treino, desc_treino, notas_treino, data_inicio, data_fim, fisioterapeuta, tratamento)

logs_paciente_alertas (id(PK), tipo, data_logs, utilizador, id_paciente, min_pressao_arterial_max, max_pressao_arterial_max, min_pressao_arterial_min, max_pressao_arterial_min, min_freq_cardiaca, max_freq_cardiaca)

Legenda: PK - Chave primária (Primary Key)

FK - Chave estrangeira (Foreign Key)

7.1.2 Normalização

A normalização é um processo de otimização utilizado como guia no desenho de base de dados relacionais de forma a garantir que as várias estruturas da base de dados são eficientes na representação da informação.

Primeira forma normal (1FN) – Requer que todos os atributos sejam atómicos. A normalização para 1FN elimina grupos repetitivos.
Segunda forma normal (2FN) – requer 1FN e refere que os atributos que não pertencem a uma chave candidata devem depender da mesma na totalidade e não parcialmente.
Terceira forma normal (3FN) – requer 2FN e requer que não haja nenhuma dependência funcional entre atributos não-chave.

Considerando os conhecimentos adquiridos ao longo do semestre na disciplina de Sistemas de Bases de Dados, onde a normalização foi um tema bastante em foco, a base de dados foi criada da forma mais eficiente possível, pelo que não foi necessário nenhum tipo de ajuste. Apenas de referir o cuidado a respeitar a 3FN aquando da criação da super-chave na tabela "dados" constituído pelos atributos paciente, data e hora que resolveu as possíveis dependências funcionais entre atributos não-chave que pudessem existir.

7.1.3 SGBD Alvo

O Sistema de gestão de base de dados (SGBD) fornece a interface entre os dados que são armazenados na base de dados (BD) e os seus utilizadores. Além disso permite definir, gerir e aceder aos dados existentes na BD.

O SGBD escolhido para este projeto é o PostgreSQL que tem como principais características o facto de ser open-source, baseado no modelo orientado a objetos, a compatibilidade com múltiplas plataformas, a eficiência e robustez.

7.1.4 Modelo físico

A figura 4 mostra o diagrama físico da base de dados onde os tipos de dados e constraints da SGBD escolhida se encontram bem definidas.

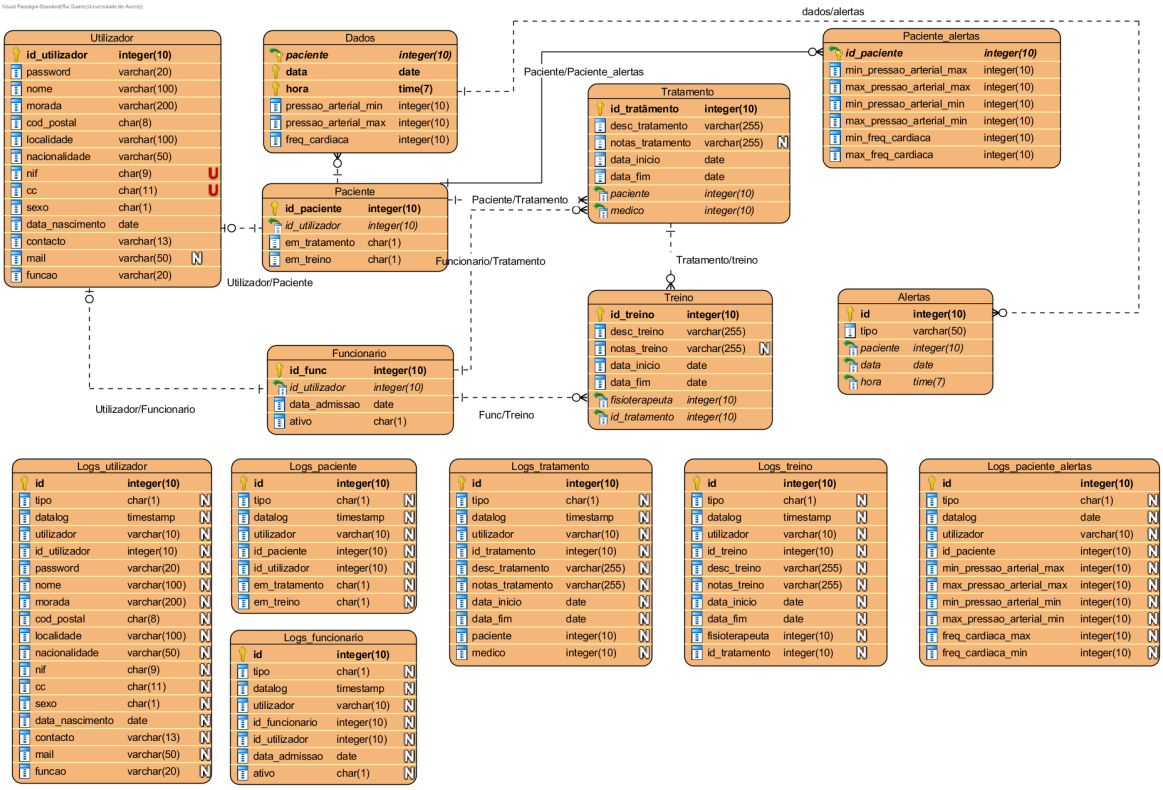

Figura 4 - Diagrama físico da base de dados

7.1.5 Restrições (Constraints)

Constraints CHECK no Domínio (globais):

```
CREATE DOMAIN SEXO AS CHAR(1) NOT NULL CHECK (VALUE IN('m','f'));
```

Este constraint permite que qualquer tabela que use este tipo de dados use esta regra (só poder ser introduzida a letra 'm' ou 'f').

```
CREATE DOMAIN SN AS CHAR (1) NOT NULL CHECK (VALUE IN('s','n'));
```

Com este constraint pretendemos substituir o tipo de dados BOOLEAN para tornar o programa mais "português" e o campo em causa poder ser preenchido com 's' ou 'n' ao invés do tradicional 't' (true) ou 'f' (false).

```
CREATE DOMAIN FUNCAO AS VARCHAR(20) NOT NULL CHECK (VALUE
IN('paciente','medico','fisioterapeuta','administrador'));
```

O atributo função permite definir o role do utilizador no sistema. O constraint com o mesmo nome permite limitar a introdução deste parâmetro para os valores definidos, de forma a manter a consistência.

Constraints NOT NULL

Os constraints NOT NULL permitem tornar um campo de preenchimento obrigatório. Foram usados em todos os atributos com a exceção dos atributos *mail*, *notas_treino* e *notas_tratamento* das relações *utilizador*, *treino* e *tratamento*, respetivamente.

Constraints UNIQUE

Os constraints UNIQUE permitem somente a introdução de valores únicos. Considerando que as chaves primárias e estrangeiras são sempre únicas, apenas foi aplicado este conceito aos atributos referentes ao cartão de cidadão ('cc') e número de contribuinte ('nif').

7.1.6 Procedimentos de execução automática (*Triggers*)

Um trigger é um tipo especial de Stored Procedure que é invocado automaticamente sempre que uma query de ação (insert, update ou delete) é executada sobre uma tabela ou view à qual o trigger esteja associado. Permitem implementar as regras definidas no desenho da BD, implementar regras subjacentes à lógica da aplicação e garantir a integridade e consistência dos dados.

Em termos de auditoria, foram criadas 6 tabelas de logs(registos) e 6 triggers correspondentes com o objetivo de registar todas as operações do tipo INSERT, UPDATE e DELETE efetuadas nas tabelas a que fazem referência. A essa informação é adicionado o utilizador que efetuou as referidas operações, juntamente com a data e hora da ocorrência.

Os trigger *tg_emtratamento* e *tg_emtreino* são responsáveis pela alteração dos atributos *em_tratamento* e *em_treino* para 's' (sim) sempre que um tratamento ou treino, respectivamente, é criado.

O trigger *tg_def_alertas* insere uma linha na relação *paciente_alertas* com os valores predefinidos de alertas de frequência cardíaca e pressão arterial, sempre que um paciente é adicionado à base de dados.

A criação de alertas é da responsabilidade do trigger *tg_alertas* que compara os valores da relação *dados* com os parametrizados na tabela *paciente_alertas*.

O trigger abaixo indicado (*tg_password*) permite encriptar o conteúdo do atributo 'password' referente à relação *utilizador* aquando da inserção ou alteração de um determinado utilizador, recorrendo à extensão "pgcrypto". Encriptação essa feita através de um algoritmo MD5 (128bits) com 48bits de *salt*. *Salt* é o valor obtido pela função *get_salt()*. Esta última função gera um valor aleatório que permite que utilizadores com passwords iguais tenham uma hash (password encriptada) completamente diferente.

```
CREATE EXTENSION pgcrypto;

CREATE OR REPLACE FUNCTION func_password() RETURNS TRIGGER AS $utilizador$
  BEGIN
            NEW.password := crypt(NEW.password, gen_salt('md5'));
            RETURN NEW;
  END;
$utilizador$ LANGUAGE plpgsql;

CREATE TRIGGER tg_password BEFORE INSERT OR UPDATE OF password ON utilizador
FOR EACH ROW EXECUTE PROCEDURE func_password();
```

7.1.7 Mecanismos de segurança

No âmbito da segurança foram criados 5 grupos de acesso com permissões distintas. Sempre que é criada um utilizador, o mesmo herda as permissões referentes ao grupo a que se encontra.

```
CREATE ROLE Tecnico ENCRYPTED PASSWORD 'ptda4' LOGIN SUPERUSER CREATEDB
CREATEROLE;
```

Este grupo é destinado a um técnico de base-de-dados. É um utilizador com poderes totais (superuser), encarregue da manutenção da base de dados.

```
CREATE ROLE Administrador NOSUPERUSER NOINHERIT NOCREATEDB CREATEROLE;
```

O administrador é, role responsável pela gestão administrativa do sistema. Cria novos utilizadores e edita dados de pacientes e funcionários e visualiza logs, quando necessário.

```
CREATE ROLE Medico NOSUPERUSER NOINHERIT NOCREATEDB NOCREATEROLE
```

O grupo dos médicos cria e edita tratamentos. Acede a dados e parametriza alertas dos pacientes.

```
CREATE ROLE Fisioterapeuta NOSUPERUSER NOINHERIT NOCREATEDB NOCREATEROLE;
```

Os fisioterapeutas criam e editam treinos. Acedem a alertas e visualizam os dados dos pacientes.

```
CREATE ROLE Paciente NOSUPERUSER NOINHERIT NOCREATEDB NOCREATEROLE;
```

Utilizadores do paciente apenas acedem aos seus treinos e tratamentos.

Na página seguinte são apresentadas matrizes de autoridade com as permissões de cada grupo de utilizadores com relação às operações na DB.

De referir que algumas das permissões foram “forçadas” pelos triggers. Como exemplo, o utilizador paciente tem acesso à relação *paciente_alertas* porque o trigger *tg_alertas* (que é executado com um paciente) necessita de acesso a essa mesma relação. O mesmo se aplica à operação de INSERT nas relações *dados* e *alertas* do mesmo utilizador.

Matriz de autoridade para o grupo "Administrador"

	utilizador	paciente	funcionario	tratamento	treino	dados	alertas	paciente_alertas	vw_paciente	vw_func	logs_utilizador	logs_paciente	logs_funcionario	logs_tratamento	logs_treino	logs_paciente_alertas
SELECT	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
INSERT	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
UPDATE	X	X	X	X	X	X	X	X	X	X						
DELETE				X	X											

Matriz de autoridade para o grupo "Medico"

	utilizador	paciente	funcionario	tratamento	treino	dados	alertas	paciente_alertas	vw_paciente	vw_func	logs_utilizador	logs_paciente	logs_funcionario	logs_tratamento	logs_treino	logs_paciente_alertas
SELECT		X		X	X	X	X	X	X	X						
INSERT				X				X			X	X		X		X
UPDATE		X		X				X	X							
DELETE				X												

Matriz de autoridade para o grupo "Fisioterapeuta"

	utilizador	paciente	funcionario	tratamento	treino	dados	alertas	paciente_alertas	vw_paciente	vw_func	logs_utilizador	logs_paciente	logs_funcionario	logs_tratamento	logs_treino	logs_paciente_alertas
SELECT		X		X	X	X	X	X	X	X						
INSERT					X						X	X			X	
UPDATE		X			X				X							
DELETE					X											

Matriz de autoridade para o grupo "Paciente"

OP/Tabela	utilizador	paciente	funcionario	tratamento	treino	dados	alertas	paciente_alertas	vw_paciente	vw_func	logs_utilizador	logs_paciente	logs_funcionario	logs_tratamento	logs_treino	logs_paciente_alertas
SELECT				X	X			X	X							
INSERT						X	X									
UPDATE																
DELETE																

Tabela 11 - Matrizes de autoridade

7.1.8 Definição de vistas (Views)

Vistas (views) são representações virtuais de tabelas criadas a partir de comandos SELECT. Uma VIEW funciona como uma janela, dando diferentes perspetivas da BD para diferentes utilizadores. Geralmente utilizadas para devolver dados desnormalizados. Em vez de obrigar o utilizador a proceder constantemente a um conjunto de JOINS, as VIEWS permitem gerar facilmente um "ResultSet" com base numa consulta previamente gravada.

Foram criadas as views *vw_paciente* e *vw_func* para facilitar as queries no java, agrupando, respetivamente, as tabelas utilizador e paciente e utilizador e funcionário.

Só as views, simples, baseadas numa única tabela ou view são atualizáveis. O mesmo efeito pode ser obtido em views mais complexas através da criação de rules (regras) na view, o que converte as operações na view em operações nas tabelas corretas. Tal é demonstrado na rule rl_*vw_paciente* abaixo indicada.

```
CREATE OR REPLACE RULE rl_vw_paciente_INSERT AS ON INSERT TO vw_paciente DO
INSTEAD (

        INSERT INTO utilizador
VALUES(DEFAULT,NEW.password,NEW.nome,NEW.morada,NEW.cod_postal,NEW.loc
alidade,NEW.nacionalidade,NEW.nif,NEW.cc,NEW.sexo,NEW.data_nascimento,NEW.
contacto,NEW.mail,NEW.funcao);

        INSERT INTO paciente
VALUES(DEFAULT,currval('utilizador_id_utilizador_seq'),NEW.em_tratamento,NEW.e
m_treino); );

CREATE OR REPLACE RULE rl_vw_paciente_UPDATE AS ON UPDATE TO vw_paciente
DO INSTEAD (

        UPDATE utilizador SET
password=NEW.password,nome=NEW.nome,morada=NEW.morada,cod_postal=NEW.
cod_postal,localidade=NEW.localidade,nacionalidade=NEW.nacionalidade,nif=NEW.
nif,cc=NEW.cc,sexo=NEW.sexo,data_nascimento=NEW.data_nascimento,contacto=N
EW.contacto,mail=NEW.mail,funcao=NEW.funcao WHERE
id_utilizador=OLD.id_utilizador;

        UPDATE paciente SET em_tratamento=NEW.em_tratamento,
em_treino=NEW.em_treino WHERE id_paciente=OLD.id_paciente; );

CREATE OR REPLACE RULE rl_vw_paciente_DELETE AS ON DELETE TO vw_paciente DO
INSTEAD (

        DELETE FROM paciente WHERE id_paciente=OLD.id_paciente;

        DELETE FROM utilizador WHERE id_utilizador=OLD.id_utilizador; );
```

7.2 Comunicação com a base de dados

No nosso projeto criámos uma classe com o nome de "Driver" que é responsável por todas as comunicações que ocorrem entre a base de dados (BD) e as interfaces. Para estabelecer a ligação entre esta classe e a base de dados foi utilizado o Java Database Connectivity (JDBC) que é uma interface de programação de aplicações (API) que permite o envio de instruções SQL através do java.

Nesta classe temos todas as queries que são feitas à base de dados bem como todos os métodos de inserção ou alteração de dados na BD. Ao criar um objeto desta classe, o construtor faz automaticamente o login com os dados fornecidos na criação do mesmo. Para verificar se o login foi bem sucedido pode-se utilizar o método isConnected(), que está explicado mais à frente.

Os métodos existentes na classe são os que se seguem:

Métodos públicos:

- addFunc() - Método responsável pela a adição de funcionários à base de dados.
- addPac() - Método responsável pela a adição de pacientes à base de dados.
- addTratamento() - Método que adiciona um tratamento à base de dados.
- addTreino() - Método que adiciona um treino à base de dados.
- dadosFuncionario() - Método que devolve todos os dados de um funcionário.
- dadosPaciente() - Método que devolve todos os dados de um paciente.
- editFunc() - Método responsável pela alteração dos dados dos funcionários.
- editPaciente() - Método responsável pela alteração dos dados dos pacientes.
- editTratamento() - Método que edita um tratamento existente na base de dados.
- editTreino() - Método que edita um treino existente na base de dados.
- enviarDados() - Método que envia os dados registados pela pulseira para a base de dados.
- finalizarTratamento() - Método que finaliza o tratamento atual de um paciente.
- finalizarTreino() - Método que finaliza o treino atual de um paciente.
- getAlertas() - Método que devolve todos os alertas de um paciente.
- getAllFuncionarios() - Método que devolve a view vw_func em formato de ResultSet.
- getAllPacientes() - Método que devolve a view vw_paciente em formato de ResultSet.
- getDados() - Método que devolve todos os dados registados pela pulseira de um paciente.
- getDadosTratamento() - Método que devolve todas as informações de um tratamento.
- getFuncID() - Método que através do NIF devolve o ID de funcionário de um utilizador.
- getLimites() - Método que devolve os limites de cada paciente.
- getPacienteID() - Método que através do NIF devolve o ID de paciente de um utilizador.
- getTratamentoEstado() - Método que verifica se um paciente tem um tratamento ativo associado.

- getTratamentos() - Método que devolve todos os tratamentos de um paciente.
- getTreinoEstado() - Método que verifica se um paciente tem um treino activo associado.
- getTreinos() - Método que devolve todos os treinos associados a um tratamento.
- getUserID() - Método que através do NIF devolve o ID de utilizador.
- isConnected() - Método que verifica se o user está ligado à base de dados.
- logout() - Método responsável pelo logout dos users na base de dados.
- pacientesComTratamento() - Método que devolve todos os pacientes com um tratamento activo.
- pacientesSemTratamento() - Método que devolve todos os pacientes sem um tratamento activo.
- setLimites() - Método que define os limites máximos e mínimos de cada paciente. Limites utilizados para a determinação dos alertas.
- updateEstadoFuncionario() - Método que actualiza um estado de um funcionário (caso este fique activo/desactivo).
- userType() - Método que devolve o tipo de utilizador logado (paciente, fisioterapeuta, médico, administrador).

Métodos privados:

- login() - Método responsável pelo login na base de dados.
- queryDB() - Método responsável pela a execução de queries à base de dados. Criado para poupar código e facilitar as queries.

7.3 *Interface*

A interface gráfica foi criada com base no modelo de dados persistente e nos requisitos funcionais e não funcionais do projeto. Foi utilizada a plataforma *JavaFX* para o desenho e criação da interface gráfica do utilizador (GUI) recorrendo ao Java IDE (integrated development environment) denominado *IntelliJ IDEA*.

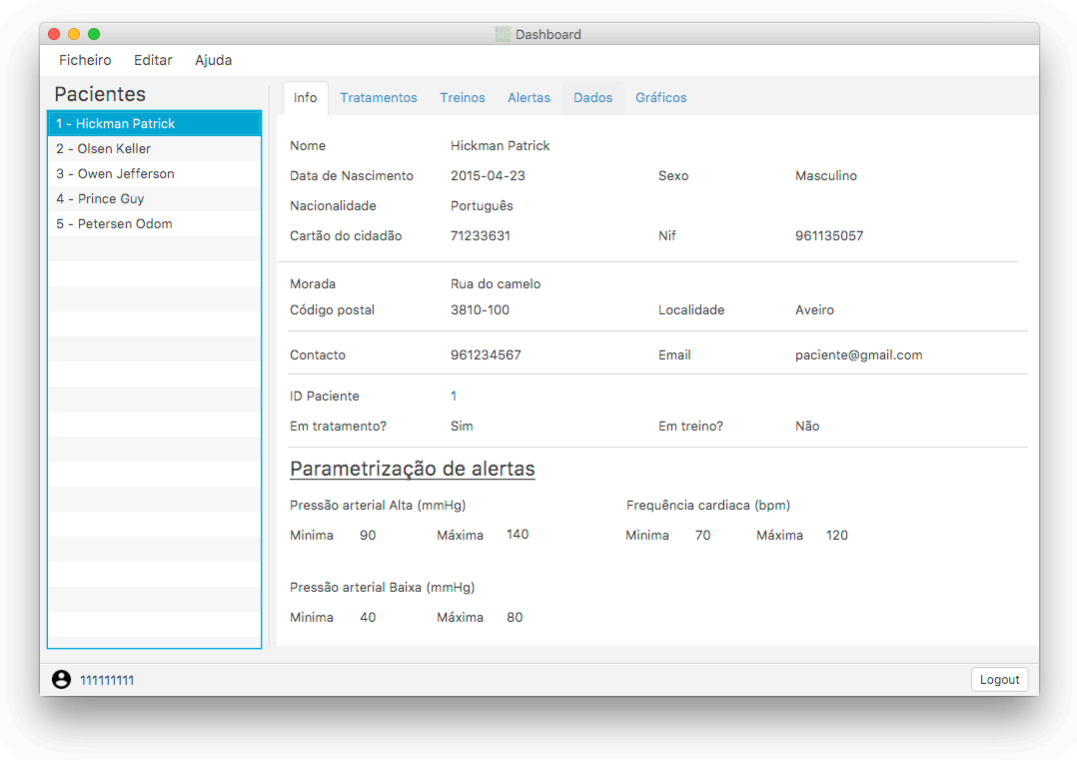

Figura 5 - Interface médico

Todas as interfaces foram criadas com base em diversos conceitos e conhecimentos adquiridos na disciplina de IHC (Interação Humano-Computador), pondo em prática conceitos como a teoria de Nielsen, profundidade e abrangência, entre outros. De forma a tornar a experiência de utilização mais agradável e moderna, foi adicionada a toda interface um estilo de componentes gráficos baseados em *Bootstrap 3* (a partir de um ficheiro CSS).

De forma ajudar o utilizador a identificar e corrigir erros, foram também adicionadas várias mensagens e notificações por todo o sistema (tanto de alerta como de erro). Para prevenir erros, foram adicionados limites mínimos e máximos para aquando da introdução de dados por parte do utilizador em formulários (nif com 9 dígitos numéricos obrigatórios, não poder ser selecionada uma data de finalização de tratamento/treino inferior à de inicio). Em baixo encontram-se alguns destes exemplos.

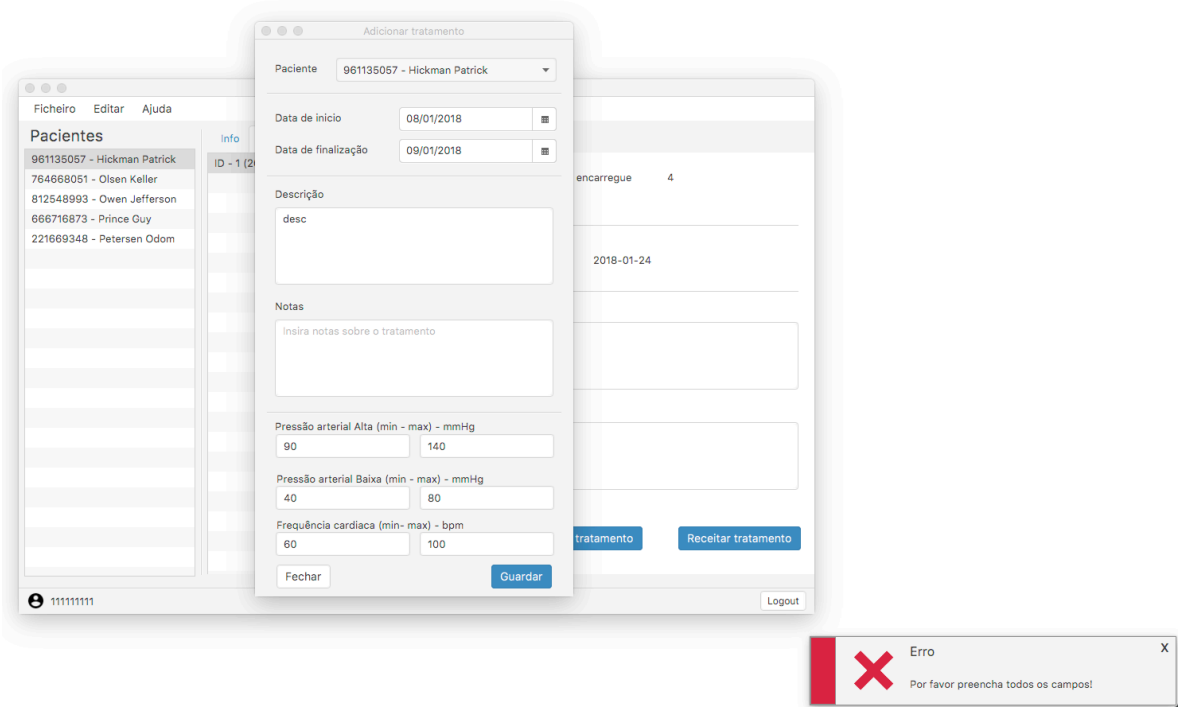

Figura 6 - Notificação de erro ao adicionar um tratamento

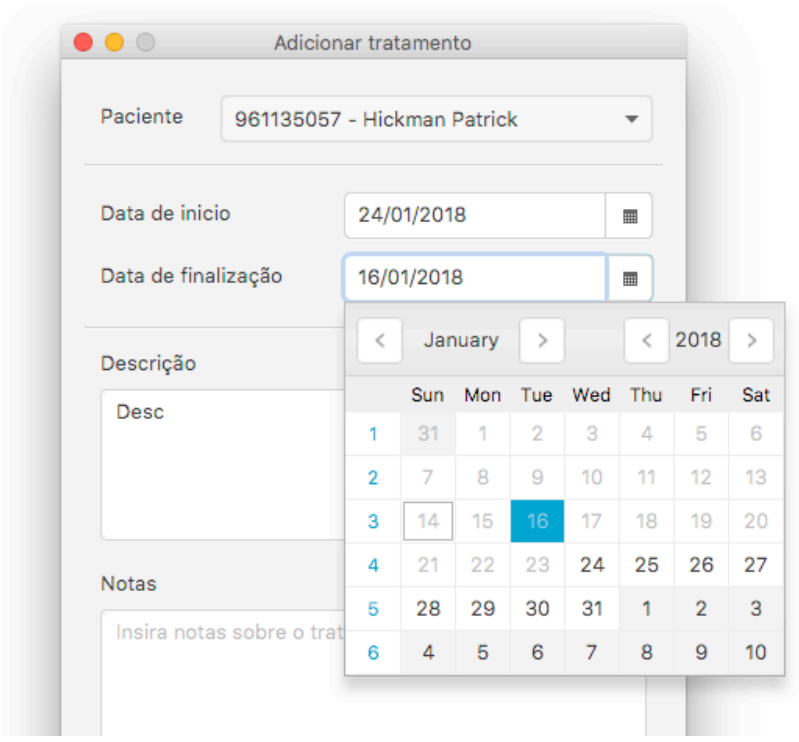

Figura 7 - Datas inferiores à de inicio não são selecionáveis

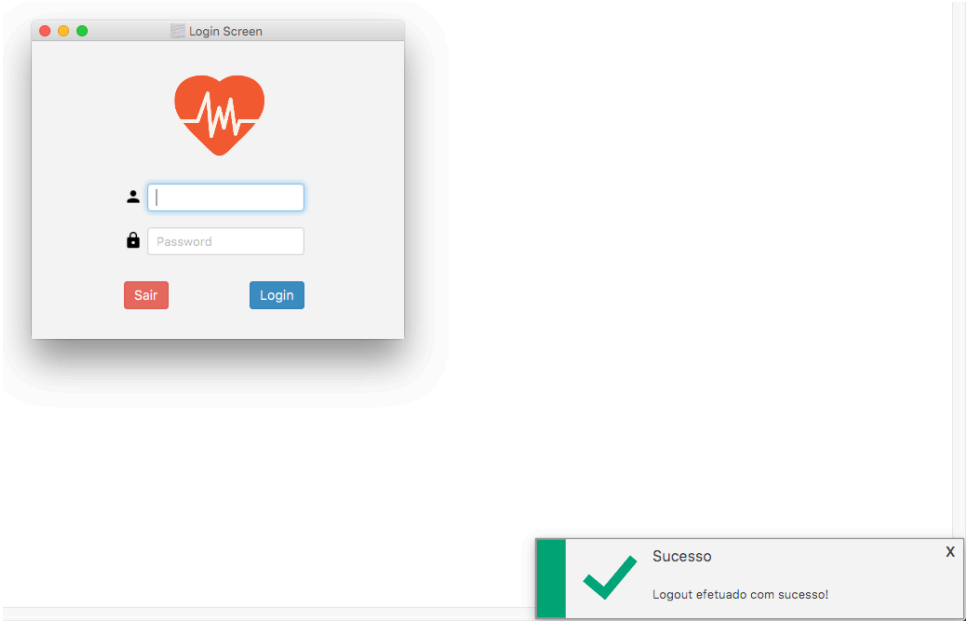

Figura 8 - Notificação de sucesso ao fazer logout

Foram criadas interfaces para os três tipos de utilizadores diferentes:

- Médico
- Fisioterapeuta
- Administrador

Interface médico/fisioterapeuta

As interfaces para o médico e fisioterapeuta são similares, com exceção das funções que cada um pode executar. O médico possui controlos para criar, editar e concluir tratamentos, enquanto que o fisioterapeuta possui controlos para criar, editar e concluir treinos.

Esta interface tem 4 componentes principais:

- Menu bar – Ações rápidas em formato de menu (opções para fazer logout, sair do programa, receitar tratamento/treino dependendo funcionário logado e menu de ajuda);
- Secção paciente – lista com todos os pacientes no sistema, sempre visível em toda a interface;
- Seção informações – secção com várias tabs para cada tipo de informação sobre os pacientes (informação do paciente, tratamentos, treinos, alertas, dados dos treinos e gráficos com os dados dos treinos);
- Barra de estado – pequena seção no fundo da página que mostra o NIF do funcionário que efetuou login e permite visualizar as suas informações pessoais (assim como fazer logout) clicando no icon à esquerda do NIF.

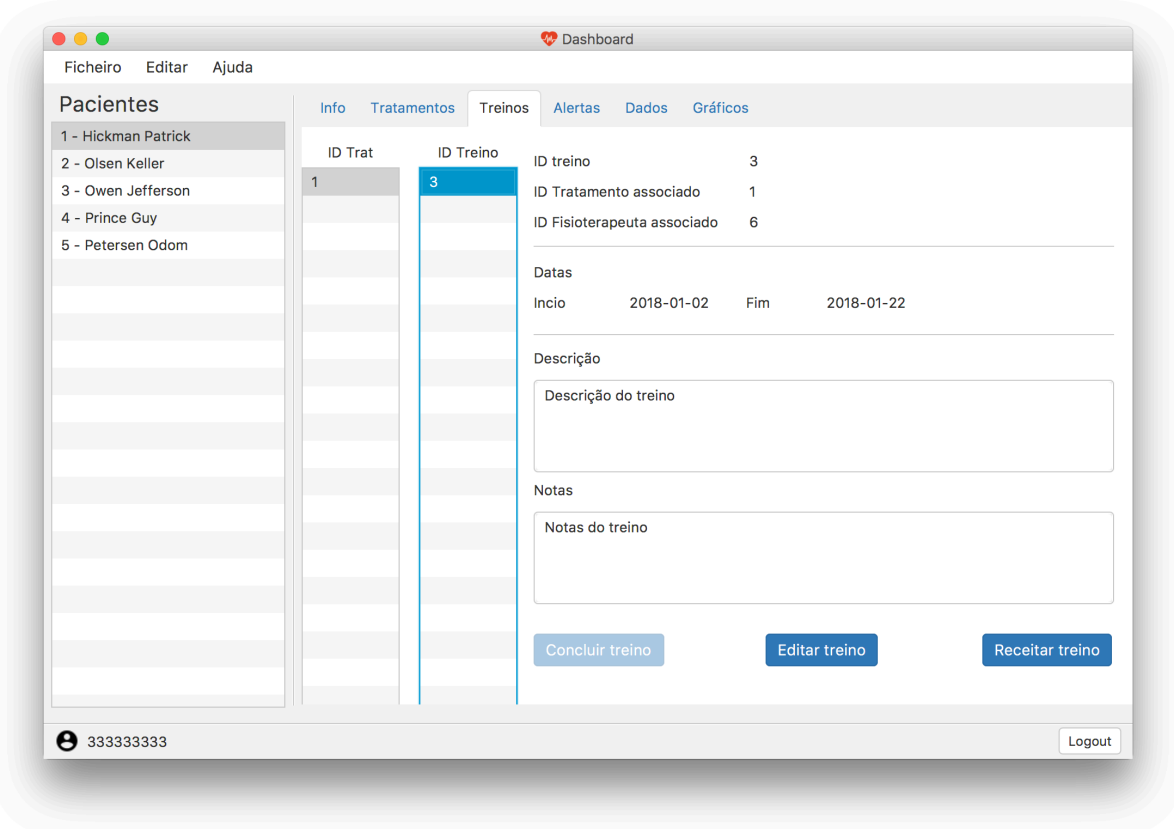

Figura 9 – Tab treinos (do fisioterapeuta)

Descrição de cada tab na secção "informações"

Tab info – mostra todas as informações pessoais relativas ao paciente selecionado. Mostra também os limites definidos (para a pressão arterial e frequência cardíaca) para a criação de alertas enquanto executa treinos.

Tab tratamentos – mostra a listagem de todos os tratamentos (passados e presentes) associados ao paciente selecionado, assim como os detalhes de cada tratamento ao lado. Se o funcionário com sessão iniciada for um médico, este terá 3 botões para executar ações: criar tratamentos, editar tratamentos (já existentes, listados na lista) e concluir tratamentos em curso.

Tab treinos – mostra a listagem de todos os treinos (passados e presentes) associados ao paciente selecionado, assim como os detalhes de cada treino ao lado. Se o funcionário com sessão iniciada for um fisioterapeuta, este terá 3 botões para executar ações: criar treinos, editar treinos (já existentes, listados na lista) e concluir treinos em curso.

Tab alertas – apresenta uma tabela com todos os alertas gerados pelo sistema de monitorização (relativos ao paciente selecionado). Esta tabela inclui colunas para o ID do alerta, tipo de alerta (pressão arterial ou frequência cardíaca), data e hora.

Tab dados – apresenta uma tabela com todos os dados registados durante todos os treinos do paciente selecionado na lista (por ordem cronológica).

Tab gráficos – esta tab contém duas sub tabs, cada uma com um gráfico: gráfico para os dados de pressão arterial (mínima e alta) e gráfico para os dados de frequência cárdica. Estes gráficos permitem ao funcionário visualizar estes dados de forma mais fácil de analisar, tornando assim mais fácil e rápida a tomada de decisões a quando a criação de tratamentos ou treinos para o paciente em questão.

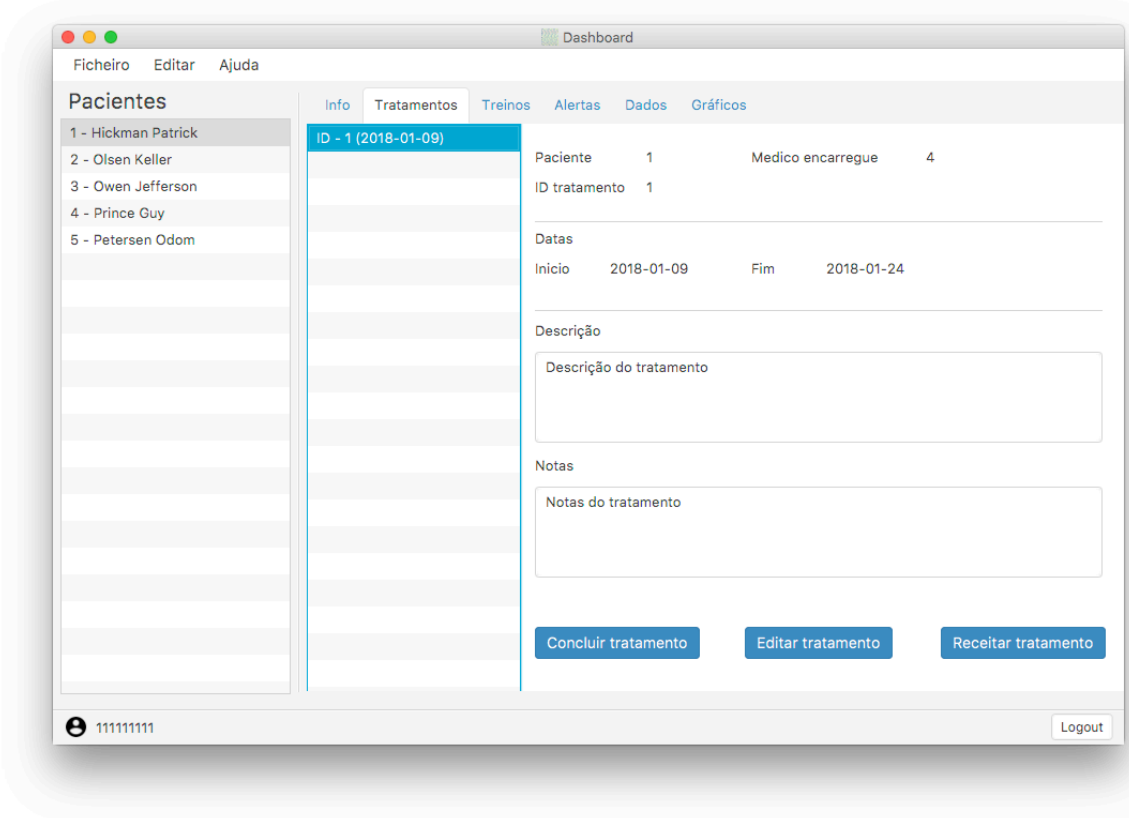

Figura 10 - Tab tratamentos (do médico)

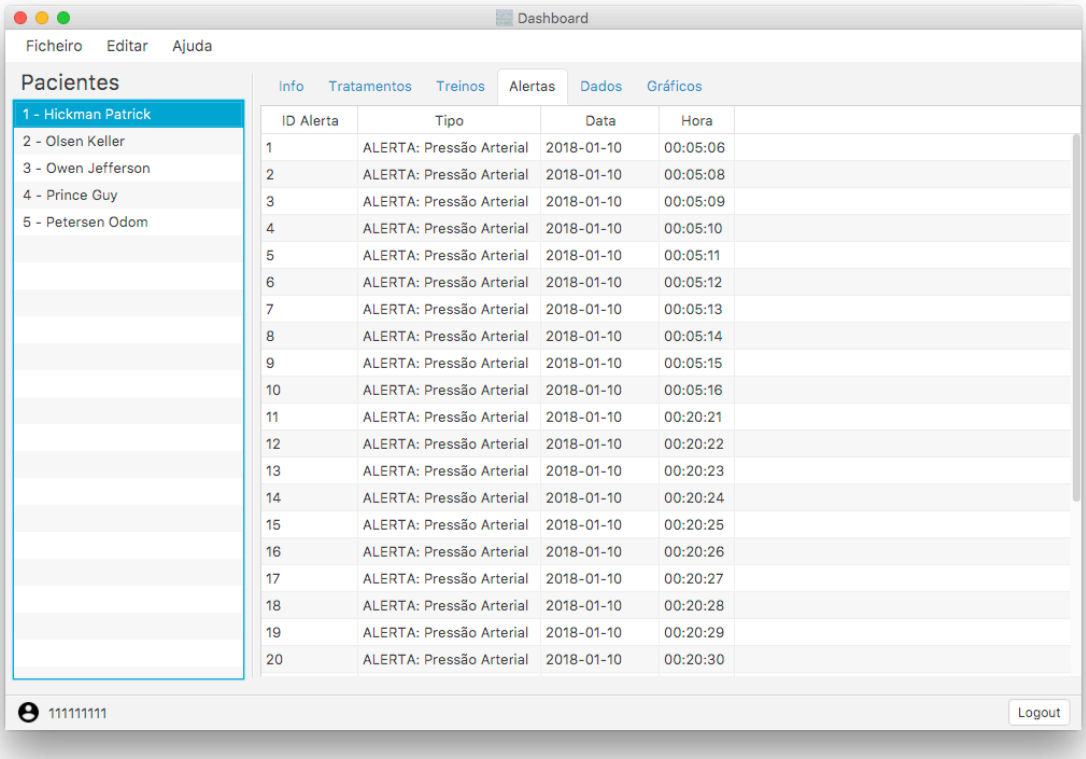

Figura 11 - Tab alertas

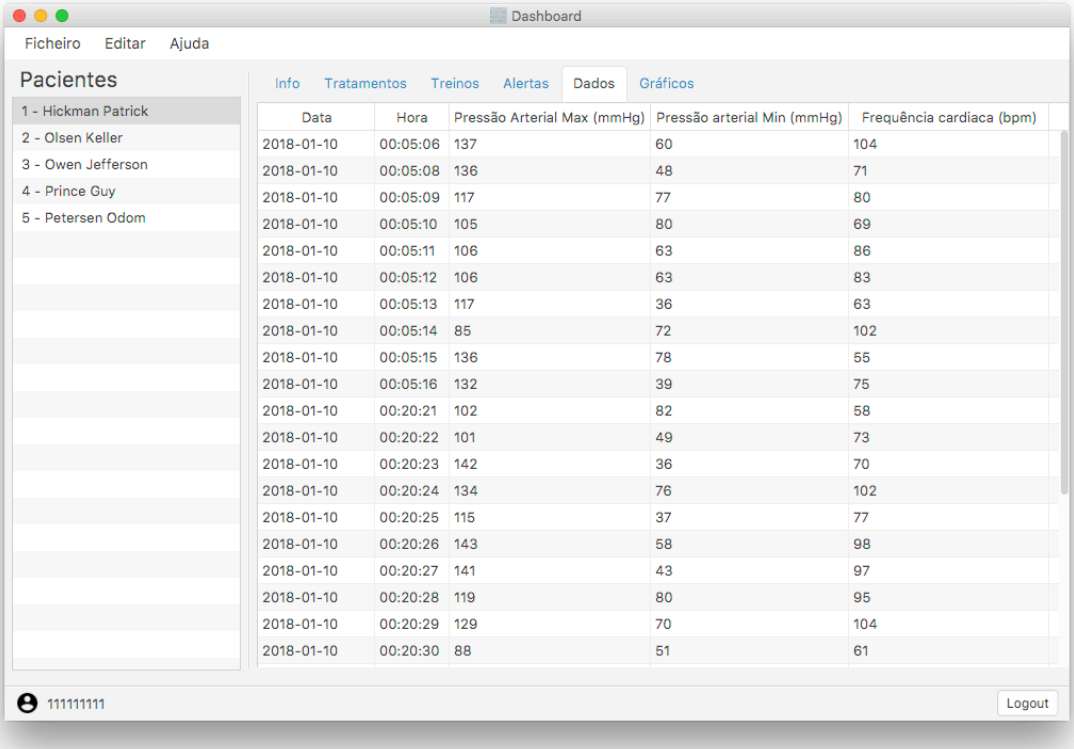

Figura 12 - Tab dados

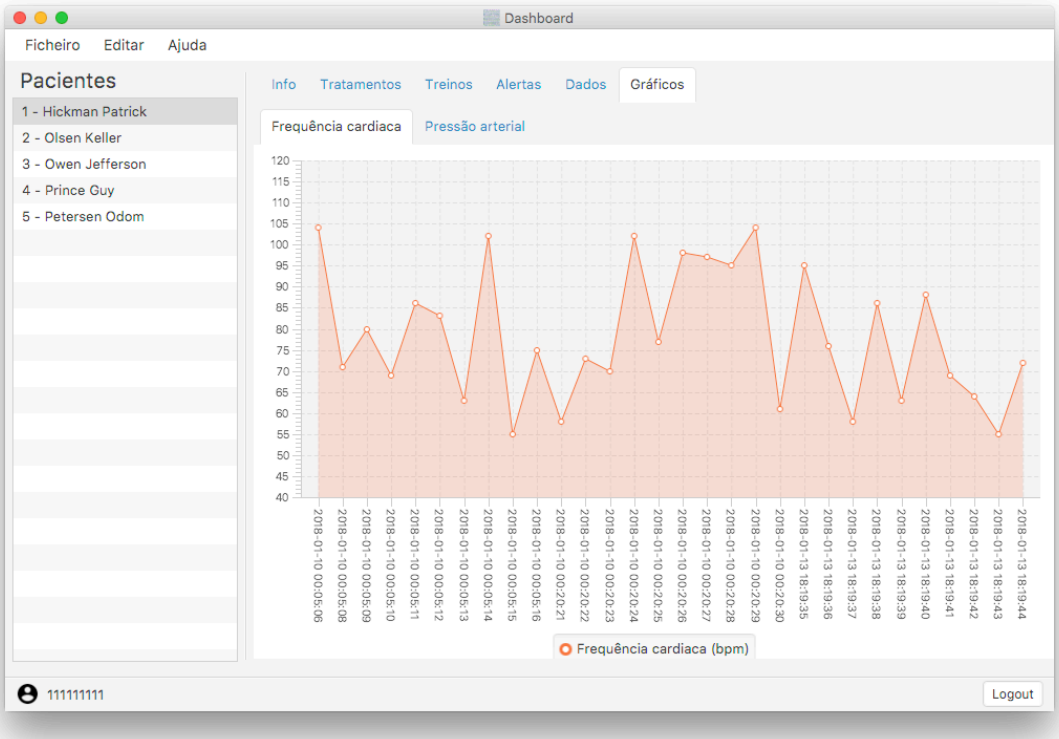

Figura 13 - Tab gráficos (frequência cardíaca)

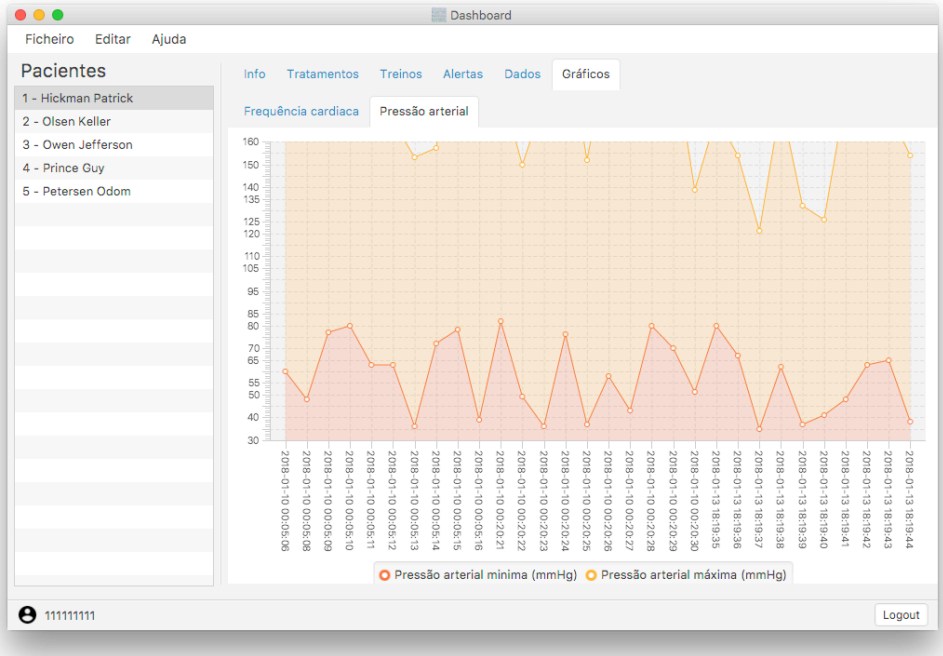

Figura 14 - Tab gráficos (pressão arterial)

Interface administrador

A interface dedicada a administradores do sistema fornece ao administrador com sessão iniciada uma lista com todos os utilizadores do sistema (pacientes e funcionários, listados por NIF e nome) juntamente com 4 botões (adicionar paciente, adicionar funcionário, editar utilizador selecionado e visualizar logs do sistema). Cada um destes botões irá abrir uma nova interface adequada ao tipo de ação que será efetuada.

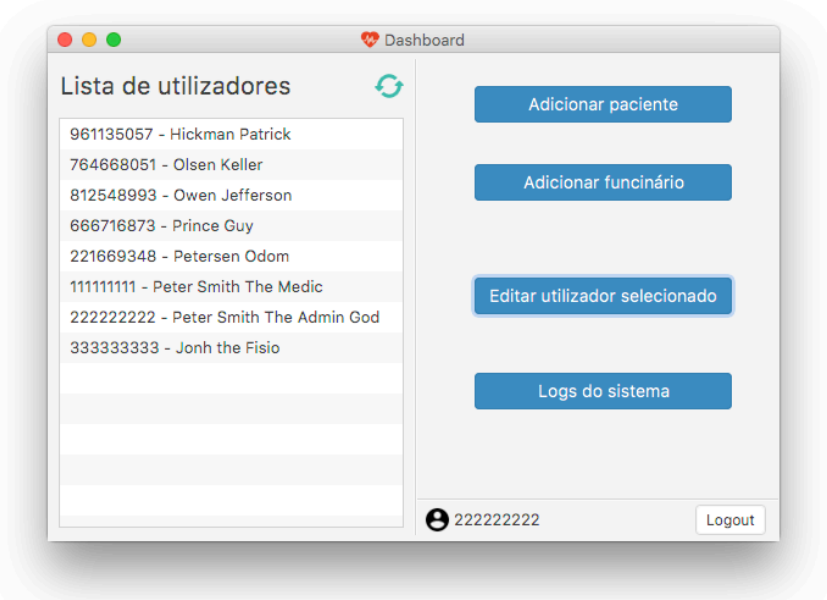

Figura 15 - Interface do administrador

8 Instalação

Para a instalação deste sistema no servidor é necessário seguir os seguintes passos:

1º) Instalar o software PostgreSQL (recomendado: instalar também o pgAdmin).

2º) Criar uma base de dados com o nome "ptda4".

3º) Executar o script “scriptDB.sql” disponibilizado para a criação da estrutura da base de dados.

Para os utilizadores deste sistema executarem o software devem seguir os seguintes passos:

1º) Verificar se o dispositivo em que pretendem correr o programa tem o Java Development Kit (JDK) instalado. Para tal podem escrever na linha de comandos "java" ou "javac". Se o sistema não reconhecer nenhum dos comandos pode-se concluir que o Java Development Kit não está instalado e deve ser instalado.

2º) Após a instalação do JDK basta correr o ficheiro ptda4.jar para iniciar o programa.

9 Análise Crítica e Conclusões

Ao longo da realização deste projeto ocorreram várias mudanças nas nossas ideias originais e houveram muitas alterações que foram feitas ao longo do tempo para melhorar a funcionalidade e a qualidade geral do produto final.

No fim de tudo podemos concluir que conseguimos ultrapassar as dificuldades que apareceram na realização deste trabalho e que o conseguimos acabar com sucesso. Todos os requisitos estipulados inicialmente foram cumpridos e como tal, acreditamos que o sistema se apresenta inteiramente funcional nesta fase. Apesar disto, acreditamos também que ainda assim existe espaço para melhoramento e aperfeiçoamento deste sistema.

10 Fontes e material de referência

NUNES, Mauro; O'NEILL, Henrique. *Fundamental de UML*. 6ª ed. FCA

SILVA, Alberto; VIDEIRA, Carlos. *UML, Metodologias e Ferramentas CASE*. 2ª ed. Vol. I. Centroatlantico.pt. Portugal, 2005

Fundação Portuguesa de Cardiologia: http://www.fpcardiologia.pt/reabilitacao-cardiaca-2/

PostgreSQL 9.6.6 Documentation: https://www.postgresql.org/docs/9.6/static/

PostgreSQL - Best way for Password Encryption using pgcrypto's Cryptographic functions: https://www.dbrnd.com/2016/03/postgresql-best-way-for-password-encryption-using-pgcryptos-cryptographic-functions/

11 Anexos

Lista de *commits* efetuados durante a realização deste projeto

Commit	Autor	Data	Descrição
0f95256	Pedro Martinho	15-01-2018 17:06:21	Adição do ficheiro de cálculo de esforço.
e703793	luispinho	14-01-2018 22:49:33	Adição do script para popular a BD. Versão beta recebe inputs de login e instruções a executar.
a0571de	luispinho	14-01-2018 19:20:51	Pequena alteração na classe Driver. O fisio consegue agora concluir treinos.
e36ffd2	RuiDuarte	14-01-2018 18:44:29	totais nas percentagens_tarefas
c9f3cba	Pedro Martinho	14-01-2018 18:28:57	Alteração dos valores no ficheiro Percentagens_Tarefas.ods
66e2c92	Pedro Martinho	14-01-2018 18:20:00	Adição das tabelas de participação nas tarefas. Atualização do gráfico de Gantt.
5d1411a	Pedro Martinho	14-01-2018 17:52:52	Adição da função getTreinoEstado().
34aa0e8	RuiDuarte	14-01-2018 04:13:15	Merge branch 'master' of https://bitbucket.org/Pinho27/ptda4
89295d9	RuiDuarte	14-01-2018 04:04:39	Alteração de privilegios no grupo fisioterapeuta; eliminição do caso de uso "enviar plano"
0f0d31a	luispinho	14-01-2018 03:59:13	Adição de interface logs no Admin. Criação da interface Fisio. Fisio consegue editar e criar treinos na interface. Alterações na interface admin. Alteração dos gráficos (2 em vez de 4). Criação e implementação da tab Alertas. Ao editar um tratamento e possível editar os limites definidos para o user. Criação de ficheiros para estruturas de dados (para os logs). Otimização do código. Adição de botão logout ao footer das interfaces.
4700b1c	RuiDuarte-PC\Rui Duarte	13-01-2018 21:31:05	-reformulacao diagrama classes - update diagrama db
92aa828	RuiDuarte	13-01-2018 21:20:48	formatacao da tabela authoring
ef68f09	luispinho	10-01-2018 00:40:34	Adicionada parte da interface no paciente para mostrar os limites definidos. Os gráficos estão operacionais. Ao adicionar um tratamento adiciona também os limites introduzidos. Pequena

			correção na Driver (função getTratamentos()).
fdf1421	Pedro Martinho	09-01-2018 21:02:40	Correcção de erro na função enviarDados().
ba6de87	Pedro Martinho	09-01-2018 19:21:11	Correcção dos erros do ultimo commit.
1c880c3	Pedro Martinho	08-01-2018 22:59:48	Adição das funções enviarDados();getDados();getLimites();setLimites() e simplificação do código.
11995b9	luispinho	05-01-2018 13:48:54	Treinos a aparecer na interface gráfica. Alteração da interface de treinos. Bug fixes.
c9198d0	Pedro Martinho	05-01-2018 13:03:42	Adição e alteração de algumas funções.
c925176	luispinho	05-01-2018 12:32:49	Adicionados gráficos á interface Doc. Restrição nos dados que podem ser editados pelo admin. Fix do médoto "finalizarTratamento" na driver.
1dc62f8	RuiDuarte	04-01-2018 18:06:30	-fix roles logs_utilizador
59c4e4d	RuiDuarte	04-01-2018 02:44:26	edit editpaciente create editfunc; editTratamento; editTreino; getTratamentoEstado fix trigger treino
1ffd7c6	RuiDuarte	04-01-2018 00:50:22	-fix nas permissoes de admin -fix nas permissoes das views -criacao logs_paciente_alertas; trigger e update roles -fix nos triggers de estado -fix no trigger alertas update authoring.xls a reflectir as alteracoes -criacao de trigger def_alertas
41db758	luispinho	03-01-2018 23:27:49	Adicionada função gráfica para adicionar e editar tratamentos. O admin já consegue adicionar pacientes. Ficheiro "Driver": bug fixes e adicionada função para ir buscar os dados de um tratamento. Edição do método de criação de funcionário.
0c16191	luispinho	03-01-2018 20:54:37	Bug fixes.
875f0c3	RuiDuarte-PC\Rui Duarte	03-01-2018 19:51:07	-fix triggers -fix roles -update authoring -criacao backup DB
dd12a56	Pedro Martinho	03-01-2018 18:01:52	Adição de funções.
b17791d	RuiDuarte-PC\Rui Duarte	03-01-2018 16:04:44	fix triggers
b48ac2a	RuiDuarte-PC\Rui Duarte	02-01-2018 21:12:12	update roles update authoring update diagrama DB
2c82c8a	Pedro Martinho	02-01-2018 18:49:40	Adição de funções. Correcção de erros.

746f2f1	luispinho	02-01-2018 03:29:55	Correção de bugs no logout. Criação e implementação das janelas de edit de utilizadores. Adicionado botão de refresh na interface do admin. Ajuste nas interfaces gráficas. Verificação de erros na introdução de dados ao criar e editar utilizadores e tratamentos. Adição de documentação JavaDoc. Refactoring de código. Bug fixes. Métodos para normalizar formato de datas. Remoção das notas do paciente. Criação do método getDriver() para poder utilizar a BD a partir de outras classes para além da Main.
c3c1162	luispinho	29-12-2017 22:12:07	Alteração da tabela de Authoring. Adição de classes e UIs para edição de pacientes funcionários e tratamentos. Funcionamento com dados reais a partir da BD. Correção de bugs e otimização do código.
d3075ab	Pedro Martinho	29-12-2017 18:29:51	Correcção de bug no authoring.
5b9f0a9	Pedro Martinho	29-12-2017 18:18:50	Alteração do authoring.
51ccfed	Pedro Martinho	29-12-2017 17:20:24	Adição de funções de query. Alteração dos add().
d2f6508	Pedro Martinho	29-12-2017 16:48:51	Atualização da Base de Dados.
57bb85f	Pedro Martinho	29-12-2017 16:30:58	Atualização da BD.
bf379eb	Pedro Martinho	29-12-2017 15:47:05	Atualização da Base de Dados.
6827993	Pedro Martinho	28-12-2017 12:12:24	Correção de um erro do ultimo commit.
7b0193e	Pedro Martinho	28-12-2017 12:10:18	Adição de getters.
1d1fadc	Pedro Martinho	27-12-2017 12:15:53	Adição de duas funções nas classes de conectividade.
43030eb	Pedro Martinho	27-12-2017 00:23:59	Modificação dos ficheiros das funções.
491ee7b	RuiDuarte-PC\Rui Duarte	26-12-2017 23:33:10	update diagrama
3045d2e	RuiDuarte-PC\Rui Duarte	26-12-2017 23:10:45	-xls c/ alertas -update sql c/alertas personalizados -update authoring
f681c3e	luispinho	26-12-2017 20:56:54	Organização da estrutra de pastas. Novo pop-up com info sobre o funcionário logado. Libraria JBDC adicionada ao projeto. Remoção de ficheiros obsoletos.
38161f0	Pedro Martinho	26-12-2017 17:14:57	Adição dos ficheiros de ligação e de manipulação da base de dados.

81e1149	luispinho	26-12-2017 17:06:58	Criação da pasta "Software". Primeiro commit e adição do programa principal em Java.
8bc351f	RuiDuarte-PC\Rui Duarte	26-12-2017 16:19:22	fix tables; update trigger alertas
bd93566	RuiDuarte-PC\Rui Duarte	21-12-2017 18:14:17	atualizacao diagrama base de dados eliminacao tabela logs_dados e respectivo trigger
7e8e242	RuiDuarte-PC\Rui Duarte	21-12-2017 17:41:31	-tabela de alertas -trigger alertas
78502d9	luispinho	21-12-2017 16:13:38	Alteração de nomes de ficheiros obsoletos.
635ccbf	RuiDuarte-PC\Rui Duarte	21-12-2017 05:05:36	db: criacao de roles; trigger para a alterar o estado da coluna em_tratamento; trigger para encriptacao;
dd5eee9	RuiDuarte-PC\Rui Duarte	21-12-2017 04:17:17	numeracao dos scripts ajustes à DB
7654fd5	Pedro Martinho	19-12-2017 12:43:35	Actualização da base de dados.
a125ffa	RuiDuarte-PC\Rui Duarte	19-12-2017 03:17:47	fix last commit
1f6439e	RuiDuarte-PC\Rui Duarte	19-12-2017 03:13:48	Alteracao do diagrama da DB DB: -views actualizaveis -encriptacao de passwords -authoring
3e5da46	RuiDuarte-PC\Rui Duarte	14-12-2017 18:58:52	Screenshot da DB Scripts iniciais da DB
4d59865	Pedro Martinho	30-10-2017 15:29:47	Adição da ata nº6.
47be55d	Daniel Martins	30-10-2017 14:36:00	update actividades screenshot
e22258c	Pedro Martinho	30-10-2017 11:14:45	Conclusão do diagrama de classes.
698f590	Rui Duarte	29-10-2017 19:42:06	update classes update usecases screenshots
c9f982f	Daniel Martins	29-10-2017 14:16:11	Diagramas de Atividades geral e individuais
6bbfaa1	Pedro Martinho	23-10-2017 15:48:28	Adição da 5ª Ata. Alteração do diagrama de classes.
b4eb5ea	Rui Duarte	22-10-2017 18:45:45	Versão inicial do diagrama de classes; Cleanup
2376ec2	Rui Duarte	21-10-2017 20:07:27	Reformulação do Use-cases Upload do screenshot
7e08dd9	Rui Duarte	21-10-2017 17:12:10	Revisão do diagrama de casos de uso Alteração da localização do mesmo
4037e44	Pedro Martinho	19-10-2017 15:48:30	Adição da 4a ata. Adição do diagrama dos casos de uso.
de4fd2a	Pedro Martinho	11-10-2017 12:20:59	Alteração no ficheiro dos requisitos
78b6d24	Pedro Martinho	11-10-2017 12:01:47	Adição da Lista de Requisitos

b845823	Pedro Martinho	09-10-2017 16:21:23	Adição das atas
dc97438	luispinho	09-10-2017 16:05:02	First commit.