FCTUC FACULDADE DE CIÊNCIAS
E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Computação Móvel

Relatório SelfGym

André Silva 2018277921 andresilva@student.dei.uc.pt
Diogo Cruz 2018306709 diogocruz@student.dei.uc.pt
Mariana Loreto 2018280762 loreto@student.dei.uc.pt

Mestrado em Engenharia Informática
2022/2023

7 de janeiro de 2023

Índice

1 Introdução

O projeto "Self Gym" surge como resposta ao desafio lançado na cadeira Computação Móvel do presente ano lectivo para a construção de uma aplicação móvel utilizando Android Studio.

Desta forma, os objetivos deste projeto passaram pela inclusão de várias técnicas e ferramentas abordadas nas aulas, como a utilização de fragmentos, atividades, bases de dados SQLite, concorrência e mecanismos de distribuição (serviços web, eventos e mensagens para notificações).

O "Self Gym" foi idealizado como uma aplicação móvel que possibilita ao utilizador gerir os seus treinos. A aplicação facilita a criação e edição treinos personalizados, através da seleção de exercícios presentes na base de dados. Permite ainda agendar os seus treinos e consultar um conjunto de estatísticas associadas aos mesmos.

Este relatório irá apresentar a estruturação deste projeto, iniciando-se pela especificação de requisitos e construção dos *mockups* iniciais da aplicação. Após esta fase foi criado o modelo físico da base de dados da aplicação e passou-se à implementação da mesma.

O *link* para o repositório do projeto em questão é o seguinte: `https://github.com/DreSilva/SelfGym`.

2 Especificação e Arquitetura

Nesta secção será apresentada a especificação dos requisitos funcionais para a aplicação "Self Gym", concretamente através da especificação de Casos de Uso. Serão também apresentados os *mockups* iniciais da aplicação.

Definidos os requisitos passou-se para especificação de Casos de Uso e estruturação da arquitetura, incluindo o modelo de base de dados como a organização a nível da aplicação.

2.1. Requisitos Funcionais

Primeiramente, foi recolhida uma lista de funcionalidades que a aplicação poderia vir a ter e classificadas seguindo o método de MoSCow de modo a atribuir-lhe uma prioridade e importância de implementação.

- Fazer login -> Could
- Fazer logout -> Could
- Criar um treino -> Must
- Apagar um treino -> Must
- Adicionar exercícios ao treino -> Must
- Apagar exercício do treino .-> Must
- Adicionar um circuito ao treino -> Must
- Apagar circuito -> Must
- Selecionar exercício por tempo ou repetições -> Should
- Filtros de procura de treino -> Should
- Filtros de procura de exercícios-> Should
- Adicionar treino ao calendário -> Should
- Notificar utilizador do horário do treino -> Could
- Exercícios devem ter imagens a demonstrar -> Could
- Definições -> Could
- O utilizador deve poder anotar observações do treino -> Must
- Histórico e estatísticas de planos de treinos feitos anteriormente -> Could
- PR notif - Won't
- Cronômetro durante o treino -> Could
- Partilha de treinos (share messenger ou semelhante) -> Could

É importante mencionar que nem todos estes requisitos chegaram ao projeto final como por exemplo a notificção de PR.

2.2. Casos de Uso

Depois da recolha de requisitos foram criados os seguintes Casos de Uso, de forma a detalhar os mais complexos e importantes. Estes Casos de Uso foram organizados por prioridades, de acordo com o método de MoSCow, de forma a orientar a ordem pela qual seriam implementados.

2.2.1. Caso de Uso 1.0

Caso de Uso 1.0	Criar Treino
Prioridade	Must have
Descrição	O utilizador deve conseguir criar um treino com vários exercícios.
Trigger	Clicar no botão "+"para adicionar um treino.
Garantas Mínimas	Notificar utilizador no caso de erros ao criar o treino.
Garantas de Sucesso	O treino é aguardado na base de dados e aparece na lista de treinos.
Pré-condições	Ter o ecrã de treinos aberto.

Guarda as alterações feitas ao treino na base de dados	
Ação Ator	**Ação Sistema**
	1. Cria um treino default vazio na BD
	2. Apresenta ecrã de treinos.
3. Insere título e tipo de treino.	
4. Insere os exercícios que pretende ao treino com os valores de sets, reps weight e time que pretende.	
5. Clica no botão para guardar	
	6. Guarda as alterações feitas ao treino na base de dados.

Pós-condições	O treino aparece na lista de treinos e pode ser adicionado ao calendário.

2.2.2. Caso de Uso 2.0

Caso de Uso 2.0	Adicionar exercício ao treino
Prioridade	Must have
Descrição	Introduzir novos exercícios a um treino.

Trigger	Utilizador carrega no botão "+"dentro da página do treino e selecionar exercício no *popup*.
Garantas Mínimas	Notificar utilizador no caso de erro ao adicionar o exercício.
Garantas de Sucesso	Exercícios novos são adicionados ao treino, e podem ser consultados posteriormente.
Pré-condições	Existir um treino criado.

Caso de Sucesso Principal	
Ação Ator	**Ação Sistema**
	1. Apresenta listagem de exercícios na categoria do treino selecionada (podendo ir buscar exercícios a outra categoria)
2. Seleciona um ou mais exercícios da lista.	
3. Carrega "Adicionar".	
	4. Atualiza exercícios e guarda na base de dados.

Pós-condições	Os exercícios selecionados são adicionados ao treino.

2.2.3. Caso de Uso 3.0

Caso de Uso 3.0	Configurar o exercício no treino
Prioridade	Must have
Descrição	Definir características do exercício (*sets*, *reps*, duração, peso).
Trigger	Utilizador seleciona o campo que pretende alterar.
Garantas Mínimas	Notificar utilizador no caso de erro ao guardar a alteração.
Garantas de Sucesso	Características do exercício são atualizadas e ficam disponíveis para consulta futura.
Pré-condições	Existe um exercício criado num treino.

Caso de Sucesso Principal	
Ação Ator	**Ação Sistema**
1. Clica no botão de editar exercícios.	
	2. Sistema carrega um *pop up* que permite alterar o tipo de exercício (tempo ou reps, variados set ou não).

3. Seleciona as opções que pretende.	
4. Clica "Guardar".	
	5. Atualiza o UI do exercício.
6. Altera o valores nos campos do exercício.	
	7. Guarda o novo valor na base de dados.
Pós-condições	Volta à página do exercício com as alterações atualizadas.

2.2.4. Caso de Uso 4.0

Caso de Uso 4.0	Aplicar Filtro de Exercícios
Prioridade	Should have
Descrição	É possível filtrar os exercícios (Parte de corpo, com peso obrigatório ou não).
Trigger	Seleciona botão “filtros” no ecrã para adicionar exercícios.
Garantas Mínimas	O filtro não é aplicado.
Garantas de Sucesso	Aparecem apenas os exercícios que se encaixam nos filtros selecionados.
Pré-condições	Existe um treino.
Caso de Sucesso Principal	
Ação Ator	**Ação Sistema**
	1. Mostra um pop up com uma barra de procura e filtros(alguns já selecionados devido ao tipo de treino).
2. Seleciona os filtros que deseja.	
3. Clica “Filtrar”.	
	4. Carrega os exercícios que respeitam os filtros.
Pós-condições	Utilizador é redirecionado para página do exercício.

2.2.5. Caso de Uso 5.0

Caso de Uso 5.0	Adicionar circuito a um treino
Prioridade	Must have
Descrição	É possível que o utilizador adicione exercícios a serem repetidos em sequência (circuito).
Trigger	Utilizador carrega no botão "+"dentro da página do treino e seleciona Circuito no popup.
Garantas Mínimas	O utilizador é notificado que não foi adicionado nenhum circuito.
Garantas de Sucesso	Aparece um novo circuito..
Pré-condições	Existe um treino.

Caso de Sucesso Principal	
Ação Ator	**Ação Sistema**
	1. Insere o circuito no treino na BD.
	2. Adiciona circuito no ecrã de exercícios.

Pós-condições	O Utilizador é redirecionado para página do treino com o circuito adicionado.

2.2.6. Caso de Uso 6.0

Caso de Uso 6.0	Deixar observação num treino
Prioridade	Must have
Descrição	Deixar comentários num treino para saber em que progredir no treino a seguir.
Trigger	Utilizador carrega sobre campo de "observações".
Garantas Mínimas	O utilizador é notificado que não foi possível guardar o comentário.
Garantas de Sucesso	A observação é guardada na base de dados.
Pré-condições	Um treino existe e está associado a um dia da semana.

Caso de Sucesso Principal	
Ação Ator	**Ação Sistema**
1. Escreve um comentário no treino sobre a sua performance.	
	2. Clica "Guardar".
3. O sistema guarda o comentário nessa instância do treino para poder ser revisto.	

Pós-condições	Utilizador consegue ver o último comentário feito a um treino quando o tenta editar.

2.2.7. Caso de Uso 7.0

Caso de Uso 7.0	Adicionar um treino ao calendário
Prioridade	Should have
Descrição	Escolhendo de treinos já criados na aplicação o utilizador pode adicioná-los ao calendário.
Trigger	Utilizador carrega no botão "+"dentro da página do calendário.
Garantas Mínimas	O utilizador é notificado que não foi possível criar um adicionar um treino.
Garantas de Sucesso	Aparece um novo treino no dia.
Pré-condições	Ter alguns treinos previamente criados.

Caso de Sucesso Principal	
Ação Ator	**Ação Sistema**
1. Seleciona o dia ao qual deseja adicionar um treino.	
	2. Insere e cria um evento na BD.
	3. Lista os treinos que foram anteriormente criados.
	4. Seleciona o treino que quer fazer, a que horas e com que frequência o evento se repete.
	5. O sistema associa os treino no dia correto na base de dados.
	6. Atualiza o *layout* com o novo treino no dia.

Pós-condições	Utilizador consegue ver o novo treino criado na página.

2.2.8. Caso de Uso 8.0

Caso de Uso 8.0	Consultar estatísticas pessoais
Prioridade	Could have

Descrição	O utilizador pode consultar as suas estatísticas de treino.
Trigger	Utilizador carrega na aba de estatísticas.
Garantas Mínimas	O utilizador é notificado que não foi possível abrir as estatísticas ou que não existem dados para as realizar.
Garantas de Sucesso	Os gráficos são mostrados no ecrã.
Pré-condições	Existirem treinos realizados.
Caso de Sucesso Principal	
Ação Ator	**Ação Sistema**
	1. O sistema mostra os gráficos disponíveis para o utilizador ver.
Pós-condições	Utilizador consegue ver alterações nos gráficos sempre que realiza um treino.

2.2.9. Caso de Uso 9.0

Caso de Uso 9.0	Correr um treino calendarizado
Prioridade	Could have
Descrição	O utilizador consegue abrir o treino que pretende iniciar e ir progredindo nos exercícios com um cronómetro auxiliar.
Trigger	Utilizador carrega no treino que pretende iniciar.
Garantas Mínimas	O utilizador é notificado de que não foi possível carregar o treino..
Garantas de Sucesso	O treino é mostrado no ecrã e é possível clicar "seguinte" nos sets e exercícios.
Pré-condições	Existirem treinos calendarizados.
Caso de Sucesso Principal	
Ação Ator	**Ação Sistema**
	1. Sistema apresenta os exercícios do treino.
2. Realiza o *set* e marca como feito.	
	3. Avança no treino.
4. Iniciializa o cronómetro (quando necessário).	
5. Dá o treino como terminado.	

	6. Atualiza a BD com novos dados de estatísticas.
Pós-condições	O utilizador consegue voltar ao ecrã do calendário.

2.2.10. Caso de Uso 10.0

Caso de Uso 10.0	Notificação antes da hora do treino
Prioridade	Could have
Descrição	O utilizador tem de marcar uma hora de início para o treino. Será notificado quando essa hora chegar.
Trigger	Utilizador coloca um treino no calendário.
Garantas Mínimas	O utilizador não é notificado.
Garantas de Sucesso	O utilizador é notificado 15 minutos antes da hora do treino.
Pré-condições	Ter treinos marcados no calendário.
Caso de Sucesso Principal	
Ação Ator	**Ação Sistema**
	1. O sistema notifica o utilizador.
Pós-condições	Utilizador consegue abrir a aplicação no calendário ao clicar na notificação.

2.3. Mockups

Nesta secção serão apresentados os *mockups* finais da aplicação. Após alguns testes, identificou-se as três áreas principais da aplicação de forma a cumprir com os requisitos identificados. Assim, definiu-se uma área de **treinos**, uma área de **estatísticas** e uma área de **calendário**.

A aplicação inicia-se por defeito na página do calendário, onde se podem consultar e agendar treinos nos dias desejados, como se pode ver na figura 2.1. A partir desta página é possível adicionar (Fig. 2.2) treinos no calendario.

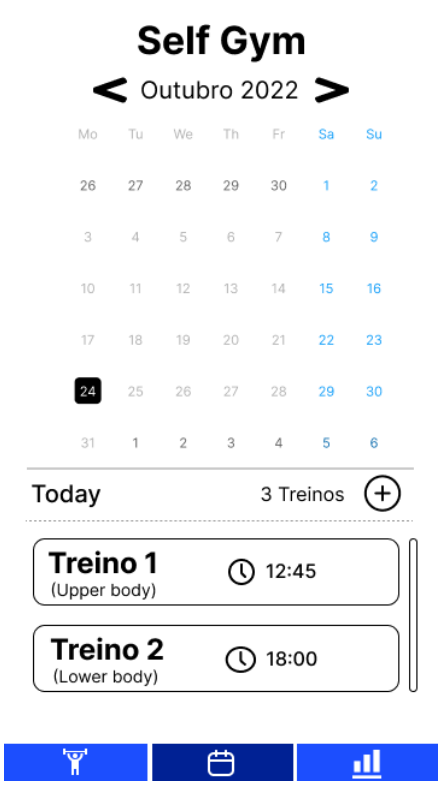

Figura 2.1: Página do Calendário

Figura 2.2: Página do Calendário com *Pop up* para Adicionar Treino

É também possivel apagar (Fig. 2.3) treinos no calendário. Na figura 2.4 é possível ver um exemplo de um lembrete de um treino calendarizado através de uma notificação.

Figura 2.3: Página do Calendário com *Pop up* para Apagar Treino

Figura 2.4: Exemplo Notificação

Na figura 2.5 pode-se ver o ecrã para concluir um treino. Será também neste ecrã que o utilizador tem acesso ao cronómetro.

Já na figura 2.6 é possível ver a mudança para a página dos treinos. Na parte superior da página é permitido filtrar os treinos tanto pelo nome, tanto pelo seu tipo.

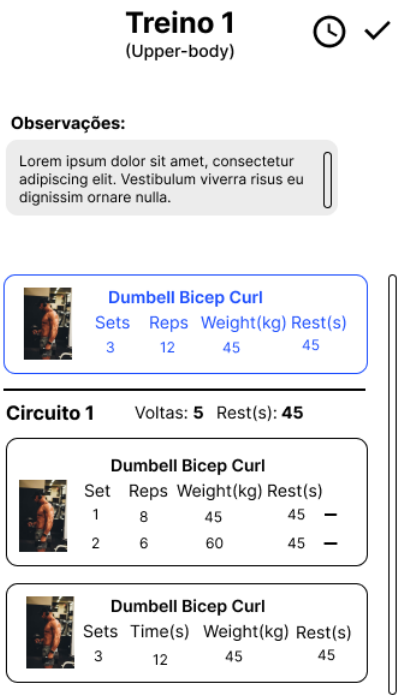

Figura 2.5: Treino com as funcionalidades de cronómetro, concluir e sair

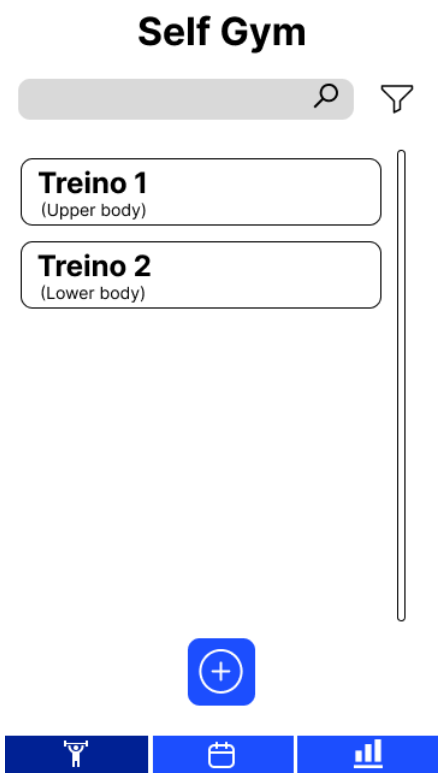

Figura 2.6: Página de Treinos

Para cada treino o utilizador pode adicionar um conjunto de exercícios ou até mesmo circuitos. Na figura 2.7 vê-se um treino com os vários exercício ou circuitos que o contistuem.

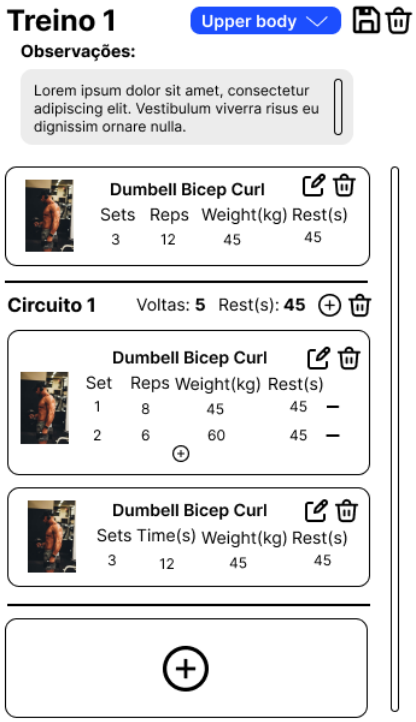

Figura 2.7: Treino Exemplo

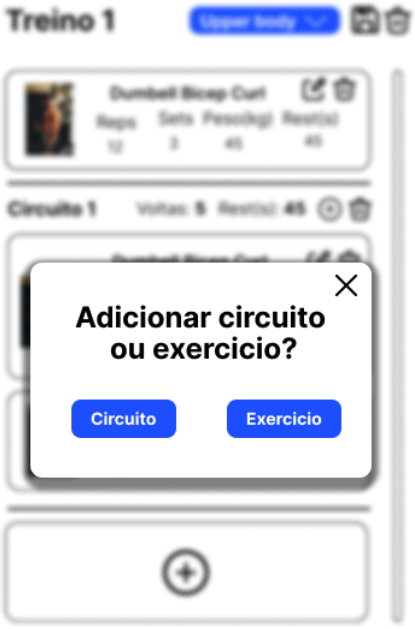

Figura 2.8: Adicionar Exercício ou Circuito

Na figura 2.8 é possível escolher o que adicionar ao treino (exercício ou circuito) e de seguida escolher o exercício desejado a partir de uma lista (Fig. 2.9). Esta lista pode ser filtrada consoante o tipo de exercício desejado (Fig. 2.10).

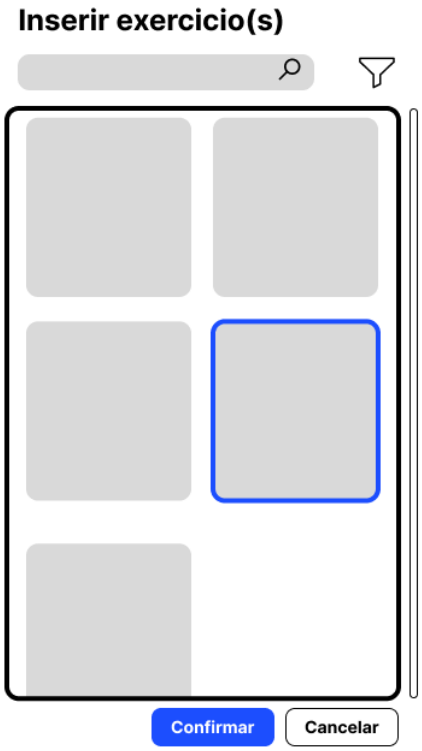

Figura 2.9: Criar Exercício

Figura 2.10: Filtrar Exercícios

Pode-se ainda editar atributos de exercícios em particular, como ilustrado na figura 2.11.

Finalmente, como se observa na figura 2.12, o utilizador tem acesso a uma secção de estatísticas associadas aos seus treinos.

Figura 2.11: Editar Exercício

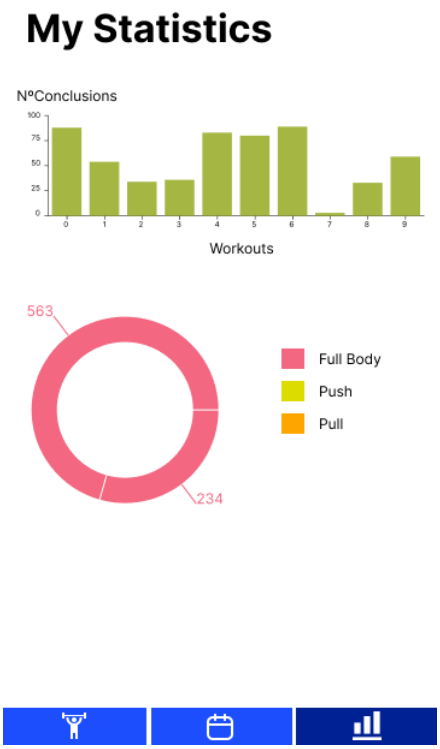

Figura 2.12: Estatísticas

2.4. Paletes de Cores e Logo

Para a escolha da palete de cores, foi utilizada a ferramenta canvas, que contém ua *color wheel* que facilita a associação de cores. Com uma análise do mercado, deparou-se também que as cores mais utilizadas para aplicações de desporto são as que contêm cores vivas e fortes porque promovem atividade física e motivação. Escolheu-se portanto, uma palete dentro desse espetro e que nos fosse apelativa visualmente.

Dentro da aplicação utilizou-se a regra do 60-30-10, **60%** para a cor principal *(West Side)*, **30%** para a cor secundária *(Aztec)* e **10%** para a cor terciária *(Catskill White)*. Esta regra segue o *standard* da indústria no desenvolvimento de aplicações.

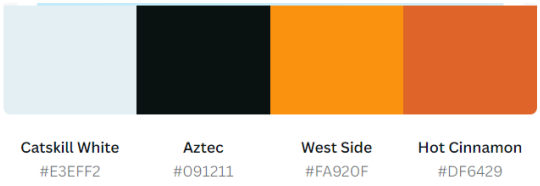

Figura 2.13: Palete de Cores.

Relativamente ao Logo também se utilizou o canvas para o criar. Considerando a palete de cores escolhida, e dentro do contexto de ser uma aplicação de ginásio, desenharam-se e experimentaram-se vários logos até se equacionar o logo representado na figura 2.14.

Figura 2.14: Evolução do logo da aplicação.

Como se pode ver na figura acima, foram incluidos elementos que aludicem à prática desportiva como pesos e foram estudadas várias combinações entre estes e o nome da aplicação, juntamente com outras elementos visuais como círculos e quadrados. A presença da palete de cores no logo final é bastante expressiva e concisa com a aplicação, tendo como base do fundo a cor principal (West Side).

2.5. Modelo de Base de Dados

De seguida, foi construído o diagrama conceptual da base de dados do "Self Gym", que se apresenta na figura 2.15. Foram concebidas seis entidades, sendo elas:

- **Workout** - um treino, identificado por um *id*, constituído por um nome e um tipo, observações e número de concluões. É uma das entidades centrais da aplicação.
- **Events** - um evento (treino agendado) no calendário.
- **Exercise** - um exercício genérico.
- **ExerciseWO** - um exercício inserido num treino (ou circuíto), ao qual estão associados um conjuntto de atributos a ser definidos pelo utilizador.
- **ExerciseSet** - um *set* (caraterísticas adicionais) de um exercício.
- **Circuit** - um circuíto com exercicíos e associado a um treino.

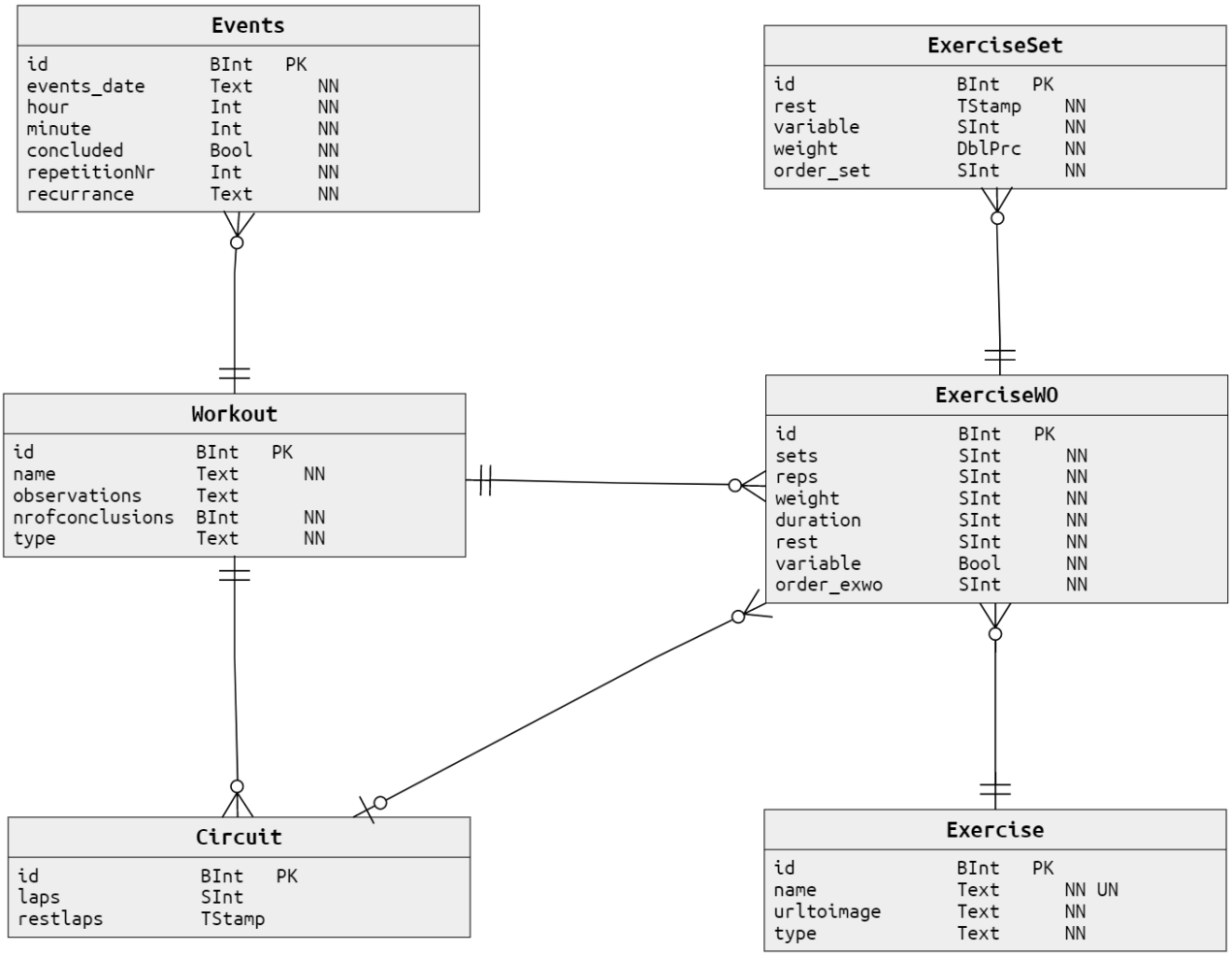

Figura 2.15: Diagrama Conceptual da Base de dados

2.6. Arquitetura da Aplicação

De seguida passou-se ao desenho da arquitetura da aplicação "Self Gym"utilizando o modelo C4, desenvolvido por Simon Brown, que consiste num conjunto hierárquico de diagramas de arquitetura de software para contexto, contentores, componentes e código da aplicação.

O diagrama de contexto da aplicação, mostra como esta se encaixa no mundo em termos das pessoas que o utilizam (o utilizador é um atleta) e dos outros sistemas de software com os quais ele interage (neste caso o MQTT).

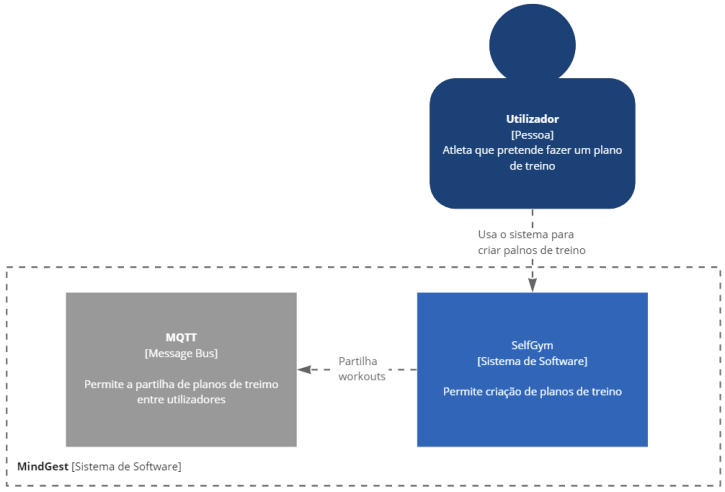

Figura 2.16: Diagrama de Contexto

O diagrama de contentores, amplia o sistema e mostra os contentores, sejam eles aplicações, base de dados, microserviços, que o compõem. No âmbito do "Self Gym", tem-se uma aplicação móvel construída em Java e é usada uma base de dados SQLite para guardar os dados do utilizador.

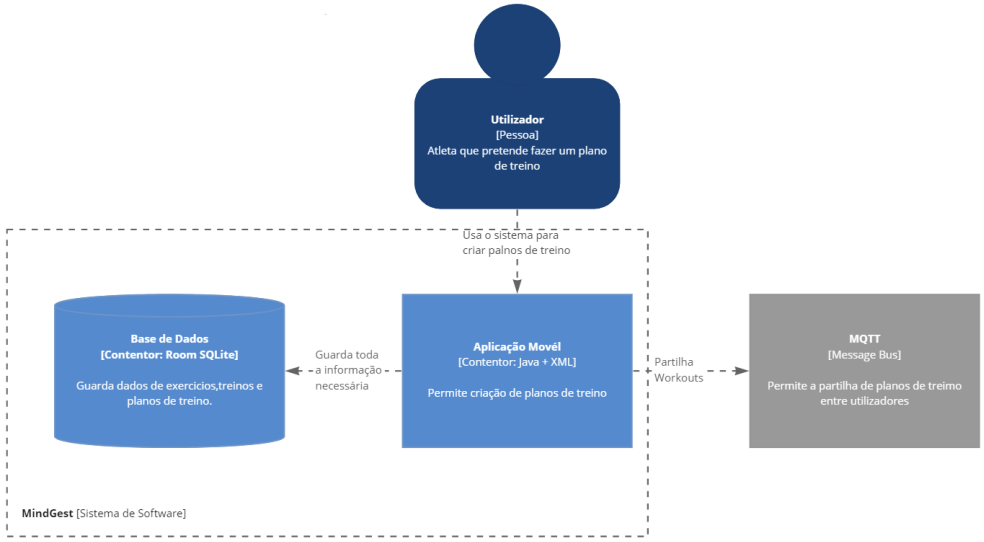

Figura 2.17: Diagrama de Contentores

O diagrama de componentes, expande o contentor da aplicação móvel para apresentar os seus componentes internos. Podemos então notar que a app se divide em 3 contentores principais: **Calendário,**

Treinos e Estatísticas. Cada um corresponde a um conjunto de fragmentos que permitem funcionalidades relacionas com esse componente. Têm também acesso a um *view model* que disponibiliza funções que devolvem dados necessários às funcionalidades do componente. Cada *view model* por sua vez acedem ao principal que tem acesso direto à base de dados da app.

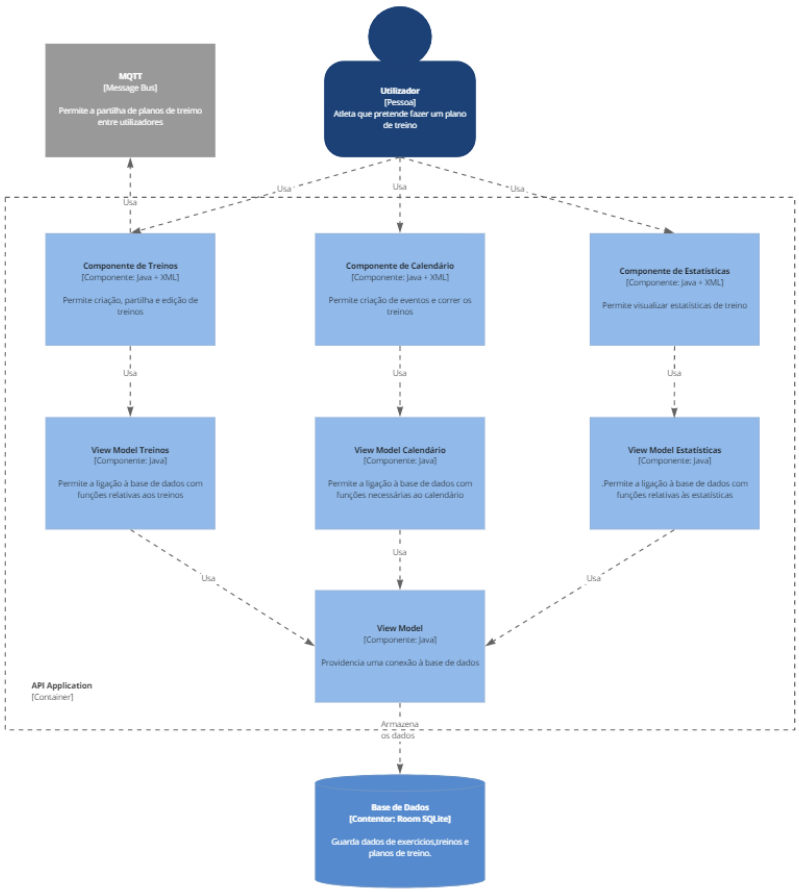

Figura 2.18: Diagrama de Compenentes

Finalmente, o diagrama de código apenas expande no diagrama dos contentores mostrando a organização das partes dos sistema nas respetivas tecnologias.

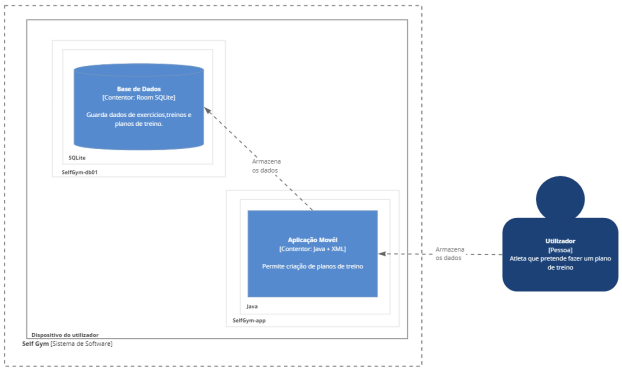

Figura 2.19: Diagrama de Código

3 Distribuição do Trabalho

Nesta secção será descrita a metodologia de trabalho, bem como a distribuição de tarefas por cada membro da equipa.

3.1. Metodologia de Trabalho

Sendo a equipa do "Self Gym"o seu próprio cliente, permitiu estabelecer desde cedo requistos claros para o projeto e um ambiente de desenvolvimento pequeno e estável. Deste modo, aplicou-se neste projeto uma metodologia de desenvolvimento do estilo *Waterfall*. Assim, adotou-se uma abordagem linear e sequencial ao desenvolvimento do projeto, implicando que cada fase fosse concluida antes de passar à fase seguinte.

3.2. Divisão do Trabalho

De seguida, apresenta-se o diagrama de Gantt com a distribuição de trabalho por cada membro do grupo.

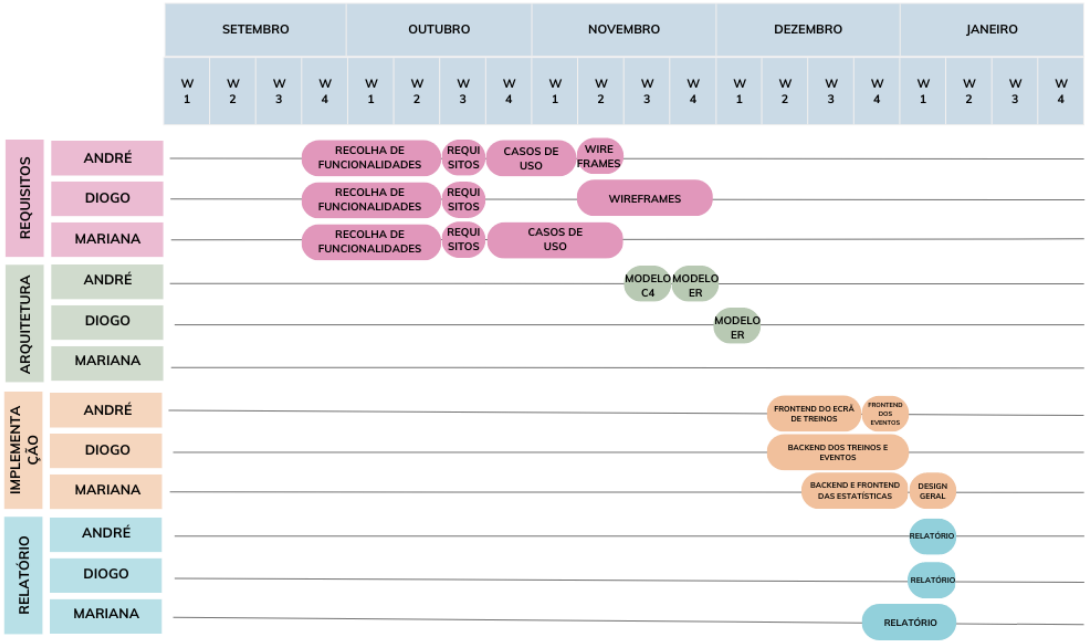

Figura 3.1: Diagrama de Gantt

Começamos o semestre por escolher as funcionalidades que iriamos desenvolver e verificamos se era um grau de complexidade aceitável de acordo com o Professor João Fernandes. Após escolhermos as funcionalidades, recolhemos os requisitos e fizemos casos de uso. Com os casos de uso fizemos os mockups, onde nos baseamos para a implementação. Com os modelo C4 definimos a arquitetura. Posteriormente passamos ao desenvolvimento da aplicação e testamos a robustez desta. Por fim escrevemos o relatorio, com todo o trabalho desenvolvido para este projeto.

4 Implementação

A implementação deste projeto usou várias dependências, sendo a principal a *Navigation Component*. Esta dependência permite a navegação entre fragmentos oferecendo um botão de ir para trás configurável. Utilizou-se a framework *ROOM*, que oferece um interface fácil para aceder à base de dados. Usou-se a dependência *gson* para transformar dados de string para json e vice-versa. Usou-se a dependência *modelmapper*, que permite a conversão de uma classe para outra. Utilizou-se ainda dependências para os charts, o *time*, as imagens vindas de links, o serviço, as notificações e para o *MQTT*.

No decorrer deste projeto, houve necessidade de adaptar o modelo descrito na arquitetura. É utilizado um *viewmodel* personalizado apenas para um workout. De forma a garantir persistência de dados entre fragmentos aquando da adição de um exercício foi preciso guardar os dados alterados previamente. Desta forma é possível recuperar os dados que foram guardados nesse viewmodel sem que estes sejam atualizados na base de dados com mudanças que o utilizador pode nem querer guardar.

Por outro lado, utiliza se um *viewmodel* geral, que contêm todos os workouts, estatisticas e eventos. Quando a aplicação é iniciada, a primeira coisa que acontece é preencher o *viewmodel* geral com todos os dados da base de dados. Após isso, este vai sendo atualizado mediante as operações realizadas. Dentro do contexto dos *viewmodels*, este projeto é ainda composto por vários observadores que garantem atualizações nos ecrã mediante a atualização dos dados.

Utilizaram-se dois serviços na aplicação, um para o *MQTT* que continuará a correr em background caso a aplicação fique em segundo plano, para permitir que a conexão ao broker continue ativa. O outro para notificar as datas dos workouts. Notifica 15 minutos antes de cada um desses ocorrer. Este ultimo serviço encontra-se sempre em execução, quer a aplicação esteja encerrada, ou não.

No que concerne à base de dados, as operações lógicas feitas, quer aos workouts, quer aos eventos necessitaram da utilização de transações para manter a consistência e integridade dos dados. É preciso usar threads para garantir que o projeto não encrava com cada operação necessária. Por fim, é preciso uma thread para correr o temporizador do treino em *background* de forma a notificar o utilizador quando este terminar.

5 Conclusão

Neste projeto foram aplicadas as várias técnicas utilizadas nos *challenges*, incluindo também outras que foram dadas apenas em aula. Entre as mais simples estão a utilização de *layouts*, atividades e fragmentos, bem como a utilização de *viewmodels* para armazenar objetos *LiveData*. Relativamente ao armazenamento de dados deu-se recurso a processos concurrentes para fazer operações em outras threads aquando o armazenamento, estudou se a framework *ROOM* para persistir dados em *SQLite*. Estudou-se ainda formas de comunicar entre serviços através de brokers *MQTT* e como implementar serviços numa aplicação android.

Ao nível da engenharia de *software*, utilizamos os modelos *C4* para arquitetura e adotamos uma metedologia *waterfall*. A primeira fase do trabalho foi a criação de requisitos, arquitetura e mockups para ter uma ideia do que era necessario desenvolver. A segunda fase foi a implementação com recurso a ferramenta *github* para gerir e organizar o trabalho entre os varios membros. A ultima fase consistiu na escrita da documentação produzida ao longo do trabalho, com a criação de um diagrama de *Gantt* para a melhor clarificação da distribuição de tarefas.