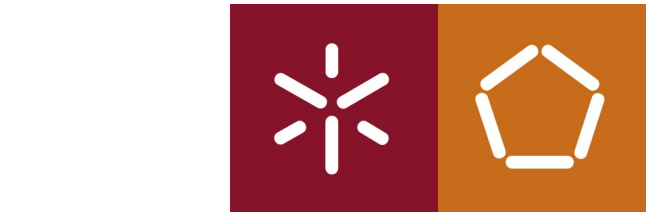

Universidade do Minho
Mestrado Integrado em Engenharia Informática
4ºano - 2º Semestre

Engenharia de Aplicações

Plataforma para Prática de Exercício Físico (*Fitness Stack*)

a84197 – João Pedro Araújo Parente
a84829 – José Nuno Martins da Costa
a84584 - Alexandra de Barros Reigada
pg44412 - Ana Margarida da Rocha Ferreira

16 de junho de 2021

Conteúdo

Lista de Figuras

1 Introdução

1.1 Motivação

Um componente essencial para um estilo de vida saudável é o exercício físico [1].

No último ano, o surgimento da pandemia de COVID-19, doença altamente contagiosa nos humanos, fez com que, a nível mundial, se tomassem medidas drásticas para o seu combate. As regras de distanciamento social, como a quarentena e o isolamento, assim como o encerramento de estabelecimentos públicos, levaram a grandes mudanças no estilo de vida das pessoas, pelo que estas passaram a estar fechadas em casa, promovendo um comportamento sedentário, diminuindo a atividade física e provocando consequências negativas, tanto na saúde física, como mental [2]. Além disso, estudos demonstram que a obesidade é um fator de risco para casos graves de COVID-19 e morte [3]. Deste modo, perante a situação pandémica que vivemos, é de extrema importância a prática regular de exercício físico, que para além de trazer inúmeros benefícios para a saúde do corpo humano, promove o bem-estar psicológico, que têm sido significativamente prejudicados.

Surge assim imperativamente a necessidade de uma Plataforma para a Prática de Exercício Físico, focada em construir um ambiente de entreajuda e de partilha de planos de treino, com a presença de treinadores pessoais qualificados, garantindo a qualidade e segurança de todos os envolvidos.

1.2 Contextualização

O **Fitness Stack** é uma plataforma *web* para a prática de exercício físico que vem disponibilizar aos utilizadores desta plataforma um meio de planear, criar e partilhar os seus treinos de uma forma simplificada, fácil e rápida. Esta aplicação contempla três tipo de intervenientes: o Utilizador, o Treinador e o Administrador. As principais características da aplicação são as seguintes:

1. Permitir que os utilizadores se registem no sistema;
2. Permitir aos utilizadores a criação, planeamento e realização de treinos com exercícios especializados criados por treinadores;
3. Permitir aos utilizadores acesso a informação acerca de como executar os exercícios corretamente através de descrições, imagens e vídeos;
4. Permitir aos utilizador partilhar e aceder a treinos criados por outros utilizadores ou treinadores e ainda selecionar os seus favoritos;

5. Permite aos utilizadores avaliar e comentar treinos assim como ser solicitar o acompanhamento de treinadores e avaliar a sua prestação;

6. Monitorizar o progresso ao longo do tempo através de um gráfico;

7. O administrador tem a responsabilidade de gerir toda a informação, inserir treinadores qualificados e remover informação que considerar imprópria.

1.3 Caracterização dos Utilizadores

De forma a perceber mais sobre os possíveis intervenientes do sistema, fez-se um questionário que foi respondido por 193 pessoas. A partir deste, foi possível concluir que os utilizadores teriam uma idade entre os 18-35 anos, uma formação académica a variar entre o Ensino Secundário e Ensino Superior e teriam uma experiência em termos de tecnologia de nível Básico a Intermédio, o que à partida não nos implicaria limitações tecnológicas. Além disso, com este estudo foi possível perceber que existem algumas funcionalidades do sistema que têm mais valor para os utilizadores, como é o caso de consultar o progresso e criar/aceder a treinos especializados, pelo que se procurou otimizar estas funcionalidades na aplicação. Os restantes utilizadores do sistema, tanto os treinadores como administradores, uma vez que são pessoas escolhidas especificamente, não existe tanta importância numa análise acerca das limitações a ponderar na interface. De qualquer modo, é boa prática mantê-la simples.

1.3.1 Utilizador

O utilizadores são o maior grupo de atores do sistema, que podem pesquisar treinos para realizar ou criar os seus próprios treinos, agendá-los e avaliá-los. Estes podem, ainda, aceder a explicações detalhadas relativamente à correta execução dos exercícios e, também, verificar treinadores disponíveis na plataforma e solicitar-lhes o acompanhamento. Durante este acompanhamento, é possível avaliar a prestação do seu treinador. Por fim, os utilizadores podem atualizar a sua informação física e monitorizar a sua evolução através de um gráfico.

1.3.2 Treinador

Os treinadores são um grupo de pessoas qualificadas, introduzidas no sistema pelos administradores, que podem criar treinos e partilhá-los com os restantes utilizadores da plataforma. Estes podem ainda ser responsáveis por

outros utilizadores, devendo disponibilizar e agendar treinos personalizados regularmente.

1.3.3 Administrador

Os administradores têm controlo sobre todas as entidades do sistema, podendo adicionar treinadores, remover treinos, exercícios, treinadores e utilizadores.

1.4 Fluxo Geral da Plataforma

Nesta secção é feita a descrição do fluxo geral da plataforma para as diferentes entidades da aplicação.

1.4.1 Utilizador

Qualquer membro pode registar-se na aplicação como utilizador. Para tal, é necessário fornecer informações pessoais. Após o registo o utilizador tem um perfil próprio no qual poderá ver as suas informações pessoais, atualizá-las e obter estatísticas relativas às últimas pesagens e o seu progresso. O utilizador pode consultar todos os treinos do sistema ou criar treinos utilizando os exercícios disponíveis. Após selecionar um treino, este pode iniciá-lo, sendo assim acompanhado pelo sistema que possui temporizadores e indica que exercícios executar e quando os executar. Além disso, um utilizador pode ainda solicitar um contrato com um treinador, desta forma se o treinador aceitar assume-se que o treinador seja responsável por este utilizador, podendo assim consultar o progresso deste seu aluno e agendar treinos. Durante esse contrato, o utilizador poderá avaliar o seu treinador.

1.4.2 Treinador

Um treinador autentica-se no sistema e de seguida poderá criar treinos e exercícios, aceitar ou recusar pedidos de alunos para ser responsável por eles e, por fim, agendar e consultar o progresso dos seus alunos.

1.4.3 Administrador

Após a autenticação, o administrador pode remover treinos, exercícios, treinadores e utilizadores, assim como registar os treinadores na plataforma, ficando assim responsável por posteriormente, fora da plataforma, fornecer as credencias aos treinadores.

2 Modelação

A fase de modelação consistiu na formalização das necessidades da plataforma. Nesta secção serão explicitados os requisitos funcionais e não funcionais, o diagrama de *use cases*, modelos de tarefas, diagramas de classes e o protótipo da interface gráfica.

2.1 Modelo de domínio

Na Figura 1 encontra-se a abordagem inicial ao problema, com a identificação das entidades e as relações entre as mesmas.

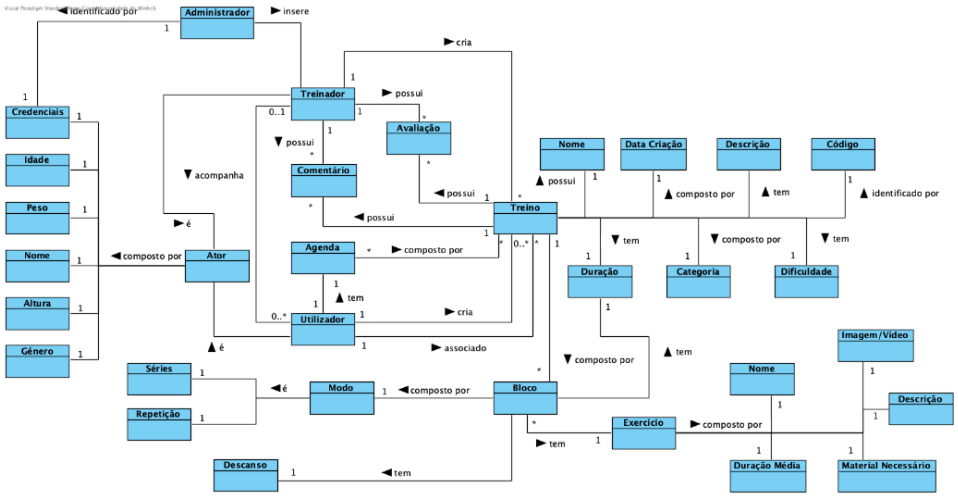

Figura 1: Modelo de domínio.

2.2 Requisitos

Tal como foi suprarreferido, efetuou-se um questionário através do qual foi possível inferir os potenciais utilizadores da aplicação, assim como as as funcionalidades mais valorizadas no sistema. Deste modo, foi feito o levantamento de requisitos com base nessas conclusões, encontrando-se descritos nesta secção.

2.2.1 Requisitos Funcionais

Os requisitos funcionais da aplicação em questão foram divididos pelos diferentes intervenientes na aplicação: utilizador, treinador e administrador.

Requisitos funcionais do utilizador:

- Criar treinos.
- Criar exercícios.
- Consultar treinos de outro utilizadores/treinadores.
- Visualizar o seu perfil.
- Pesquisar, visualizar, executar e avaliar treinos.
- Pesquisar e visualizar exercícios.
- Solicitar a um treinador para se tornar seu responsável.
- Visualizar a agenda dos seus treinos.
- Agendar os seus treinos.
- Visualizar as suas estatísticas.

Requisitos funcionais do treinador:

- Criar treinos.
- Criar exercícios.
- Consultar treinos.
- Visualizar o seu perfil.
- Pesquisar e visualizar treinos.
- Pesquisar e visualizar exercícios.
- Aceitar ou recusar o pedido de tornar-se responsável por um utilizador.
- Visualizar a agenda dos treinos de um utilizador, do qual é responsável.
- Agendar treinos a um utilizador, do qual é responsável.
- Visualizar estatísticas sobre o utilizador do qual é responsável.

Requisitos funcionais do administrador:

- Registar treinadores.
- Remover treinos.
- Remover exercícios.
- Remover utilizadores, treinadores e administradores.

2.2.2 Requisitos Não Funcionais

- O sistema deve ter disponibilidade total a partir da data de lançamento;
- A aplicação deve estar disponível em língua portuguesa;
- A aplicação não deve ser ofensiva em termos religiosos, étnicos e sexuais;
- A aplicação deve ser suportada pelos seguintes *browsers*: Safari, Google Chrome, Firefox e Microsoft Edge;
- Para incluir uma nova funcionalidade, esta deve ser lançada durante a noite no horário local;
- O tempo de resposta da aplicação deve ser o mais curto possível de forma a facilitar a experiência ao utilizador.
- O sistema não apresentará aos utilizadores quaisquer tipo de dados do tipo privado, isto é, um utilizador apenas deve ter acesso aos seus próprios dados e aos dados que são públicos a todo o sistema.

2.3 Diagramas de *Use Cases*

De acordo com as funcionalidades descritas acima, definiram-se os seguintes use cases.

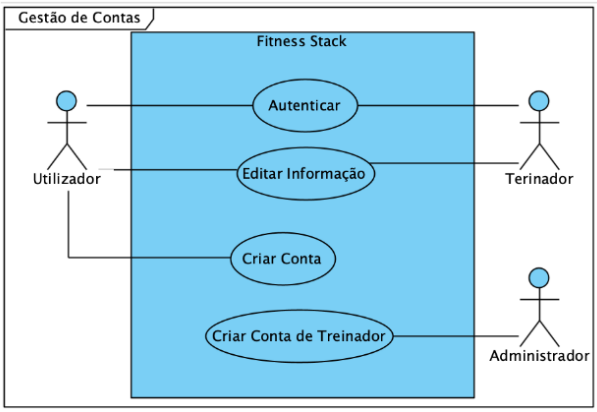

Figura 2: Diagrama de *Use Cases* Gestão de Contas.

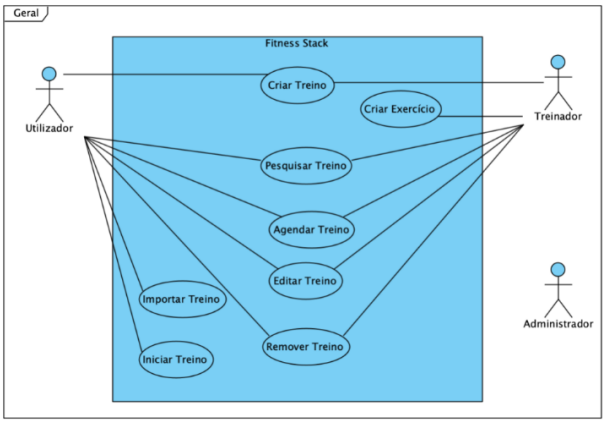

Figura 3: Diagrama de *Use Cases* Geral.

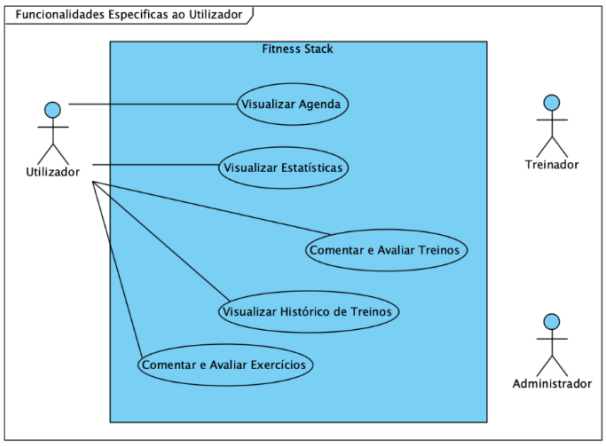

Figura 4: Diagrama de *Use Cases* Funcionalidades Específicas ao Utilizador.

2.4 Modelos de Tarefas

Decidiu-se ainda elaborar um modelo de tarefas para os *use cases* de mais importância na nossa aplicação, de forma a perceber-se melhor a interação entre os utilizadores e o sistema nestas situações.

2.4.1 Criar exercício

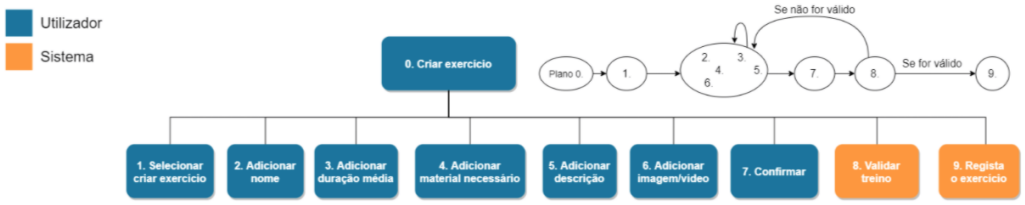

Figura 5: Modelo de tarefas: Criar exercício .

Para o treinador criar um exercício este terá de selecionar que o deseja fazer, após isso por qualquer ordem deve adicionar o nome, o material necessário, várias imagens e/ou vídeos e após todas essas operações confirmar.

2.4.2 Criar treino

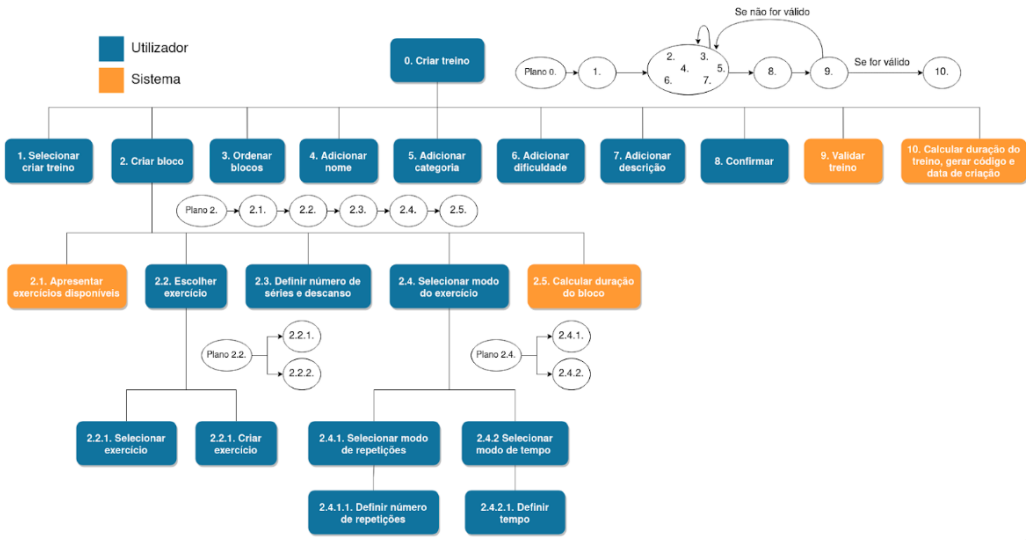

Figura 6: Modelo de tarefas: Criar exercício .

Para criar um treino, um utilizador ou treinador cria blocos, associando a cada bloco um exercício, o número de séries e descanso entre as séries e deve definir se será um exercício de repetições ou de duração. Além disso, é possível ordená-los. Por fim, associa-se a informação do treino e confirma-se a criação do treino.

2.4.3 Começar treino

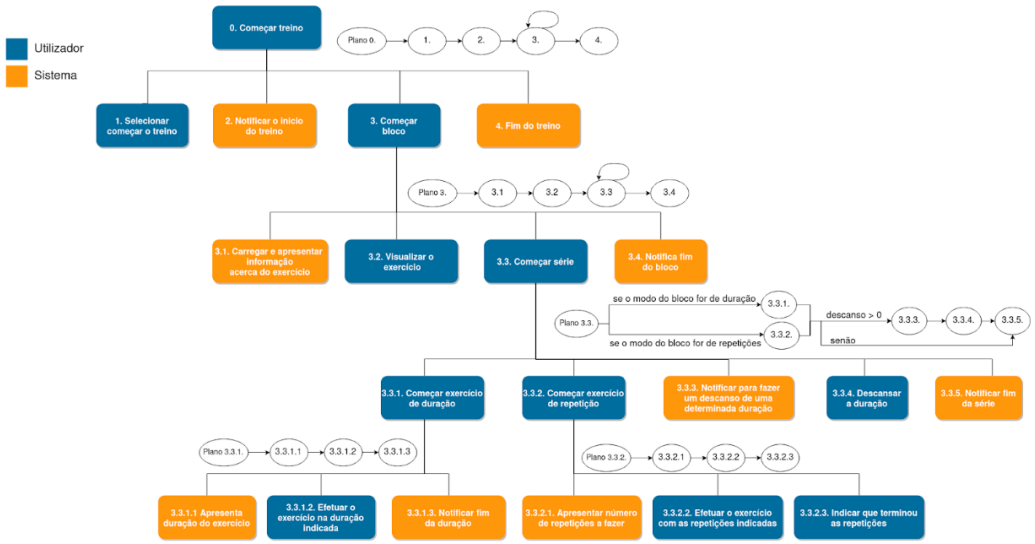

Figura 7: Modelo de tarefas: Criar exercício.

Para executar um treino, o utilizador seleciona a opção de começar o treino. Para cada bloco, o utilizador visualiza o exercício, a série em que se encontra, o número de repetições ou duração e a duração do período de descanso entre cada série.

2.5 Diagrama de Classes

A organização do código em classes tem dois objectivos fundamentais: facilitar a reutilização e facilitar a manutenção do código.

Desta forma, um diagrama de classes é uma representação da estrutura e das relações das classes que servem de modelo para objectos. É uma modelação muito útil para o desenvolvimento de sistemas, pois define todas as classes que o sistema necessita e é a base para a construção de outros diagramas.

O diagrama de classes começou como um diagrama conceptual que foi evoluindo ao longo do desenvolvimento do projeto, adaptando-se aos problemas e necessidades que foram surgindo.

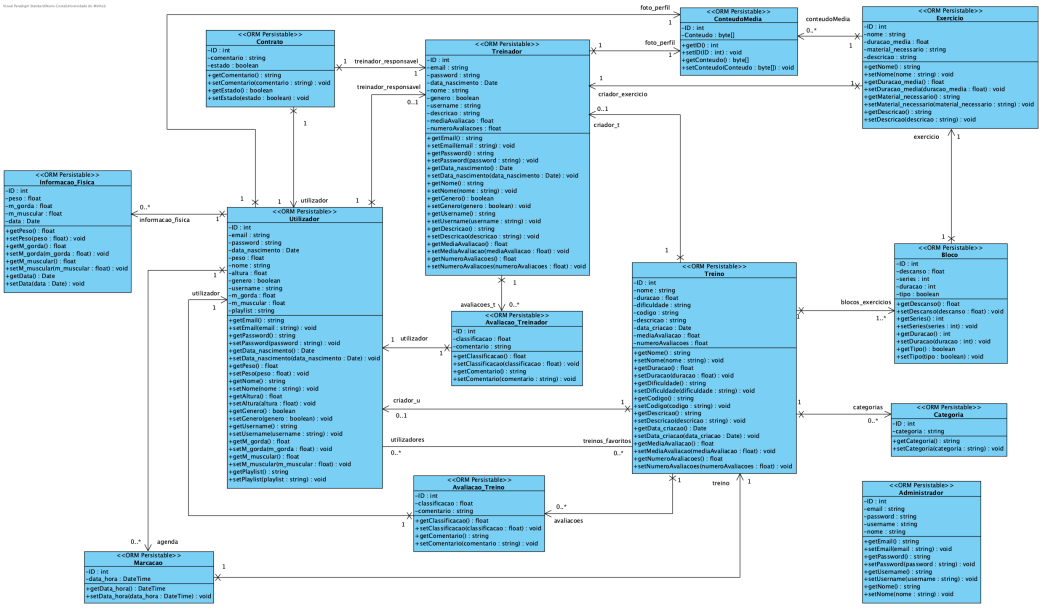

Figura 8: Diagrama de Classes.

2.6 Protótipo da Interface

Nesta fase de modelação, desenhamos uma interface de baixa fidelidade com recurso à ferramenta Figma, de forma a criar desde cedo um objetivo concreto e uma noção da informação que se pretende mostrar.

Figura 9: *Mockups*: Página inicial.

Figura 10: *Mockups*: Página sobre a aplicação.

Figura 11: *Mockups*: Página de *login*.

Figura 12: *Mockups*: Página de registo do utilizador.

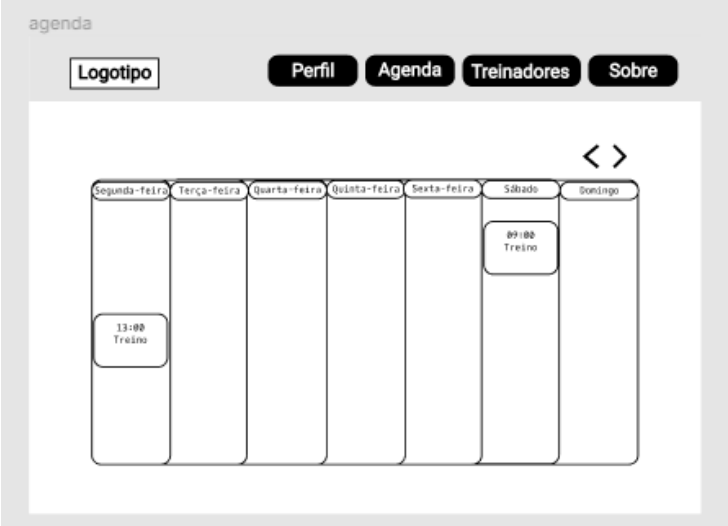

Figura 13: *Mockups*: Agenda do utilizador.

Figura 14: *Mockups*: Lista de treinos.

Figura 15: *Mockups*: Página de início de treino.

Figura 16: *Mockups*: Página inicial do utilizador.

Figura 17: *Mockups*: Informação de um treino.

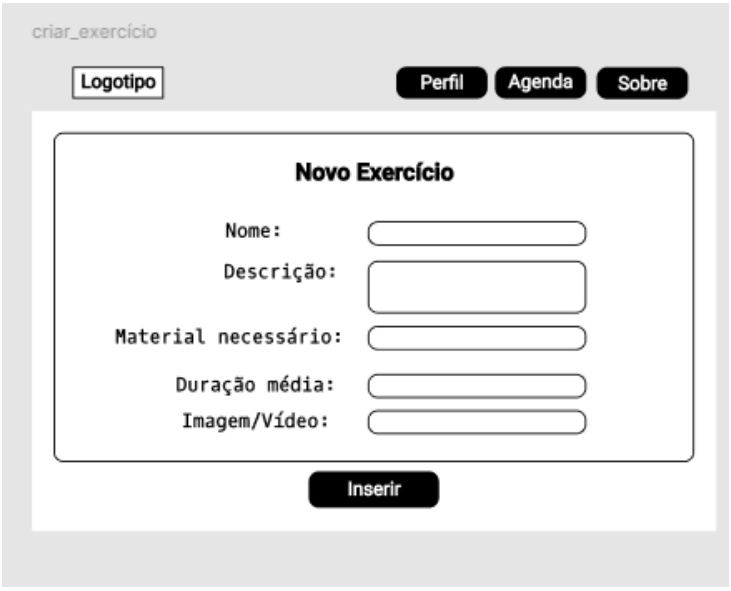

Figura 18: *Mockups*: Página para criar exercício.

Figura 19: *Mockups*: Página de execução de um treino.

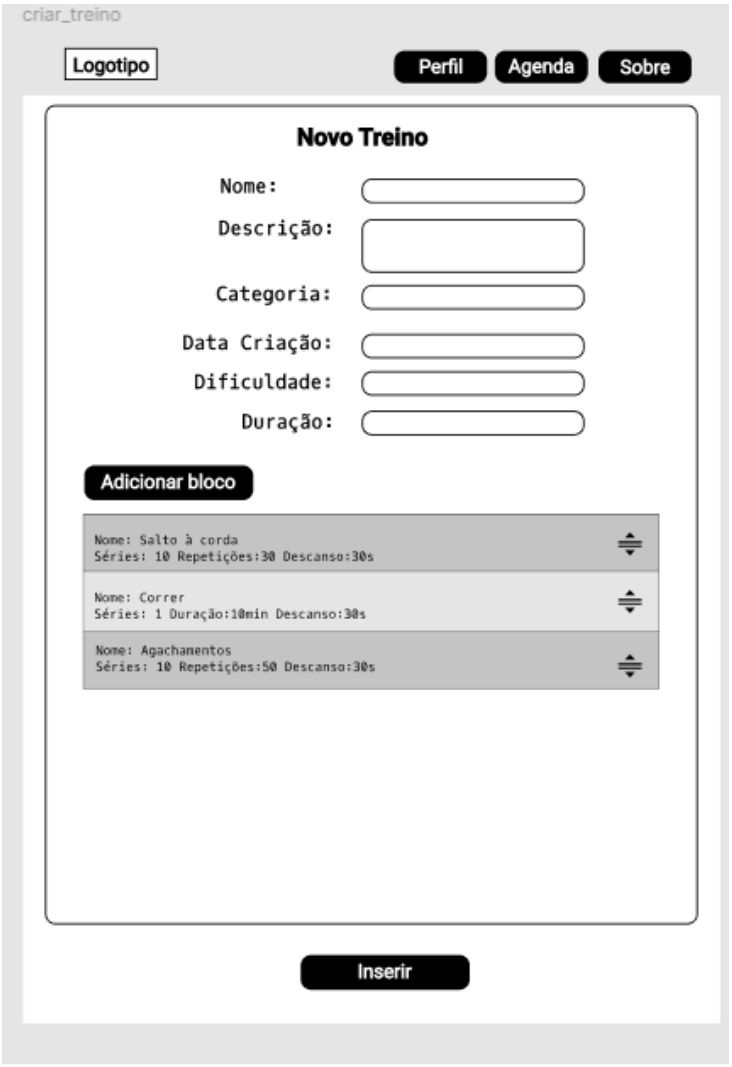

Figura 20: *Mockups*: Página para criar um treino.

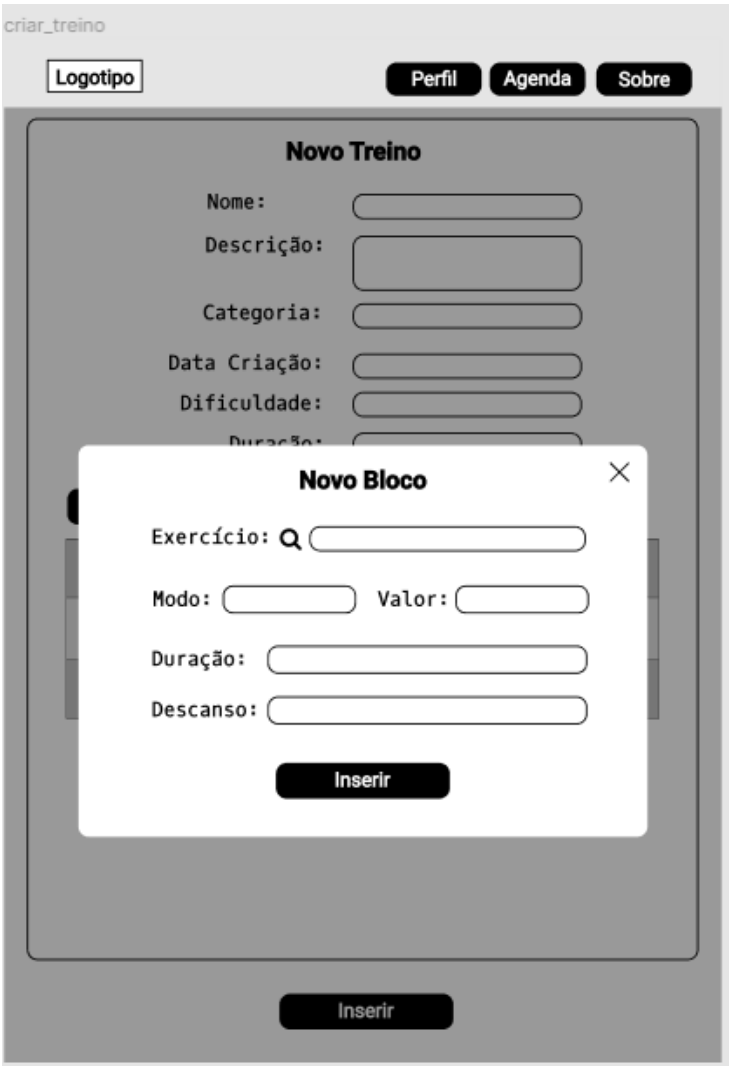

Figura 21: *Mockups*: Formulário de registo de um novo bloco no treino.

3 Implementação

Após a fase de modelação, discute-se a implementação, consideram-se várias *frameworks* e tecnologias, de modo a selecionar a mais adequada para desenvolver o projeto em questão.

3.1 Considerações Iniciais

Inicialmente, decidiu-se a linguagem de programação a ser utilizada, tendo sido escolhida a linguagem **Java** por ser uma linguagem robusta com inúmeros padrões e formas de estruturar uma arquitetura *web*. Com o intuito de facilitar o processo de desenvolvimento de uma aplicação, é útil usar *frameworks*, que são conjuntos de classes que formam um suporte de reutilização. Comparando aspetos como popularidade, suporte, simples utilização, adaptabilidade, destacou-se o **Spring** para a camada de negócio e **Hibernate** para a camada de dados. Para a camada de interface várias *frameworks* se destacaram como o Angular, React, Vue, mas devido a **Vue** ter sido uma *framework* abordada durante o período de aulas, optou-se por esta.

Resumindo, para a arquitetura da nossa aplicação foram utilizados os seguintes *frameworks*:

1. **Camada de Apresentação:** Vue
2. **Camada de Negócio:** Spring
3. **Camada de Dados:** Hibernate

3.2 *Frontend*

Como referido anteriormente, no *frontend* foi utilizado Vue. Vue, sendo uma *framework client-side*, torna a nossa aplicação mais iterativa e *responsive*, evitando a sobrecarga do servidor esperando, assim, uma melhor escalabilidade da nossa aplicação.

3.2.1 *Views*

Como páginas partilhadas por todos os intervenientes, existem a página inicial e a página *About* que contém informações acerca da aplicação. Para a autenticação e registo, existe uma página partilhada entre treinadores e utilizadores. No caso de administrador, considerou-se pertinente criar uma página "oculta", só para a autenticação deste.

Relativamente às páginas dos utilizadores, estes têm acesso à página de perfil que contém as suas informações pessoais, bem como um gráfico com o seu progresso relativamente ao seu peso, percentagem de massa gorda e percentagem de massa muscular. É nesta página que o utilizador pode alterar o *username*, email, palavra-passe, foto de perfil, *playlist* pré-definida e informação física. Tem ainda acesso a páginas com a listagem de exercícios, treinos e treinadores e informações adicionais relativas aos mesmos. Na página dos treinos é permitido criar treinos e na página dos treinadores é possível enviar mensagens aos treinadores, solicitando os seus serviços.

Já os treinadores, têm acesso ao seu perfil onde podem editar o *username*, email, palavra-passe e foto de perfil. Na página de exercícios, podem adicionar exercícios e na página de treinos podem visualizar os treinos, os exercícios que contém e as avaliações e comentários. Tem ainda uma página com os alunos pelos quais está responsável, onde pode responder, aceitando ou não, a solicitações de alunos, sendo informado das informações relevantes e ainda eventualmente de uma mensagem.

Quanto ao administrador, este pode aceder ao seu perfil, no qual pode editar informações como o *username*, *email* e palavra-passe. Além disso, este pode visualizar páginas com listas de toda a informação relativa aos utilizadores, treinadores, administradores, exercícios e treinos, que pode remover caso considere necessário. Este tem, também, a responsabilidade de registar treinadores no sistema, acedendo a um formulário para tal.

Caso seja introduzido um URL inválido, o *browser* redireciona para uma página de erro indicando que a página solicitada não foi encontrada.

3.2.2 API

A *frontend* para obter os dados comunica com a *backend* através de pedidos HTTP, sendo que para tal utilizou-se a biblioteca **axios**, com a qual o grupo já teve contacto em trabalhos passados.

3.3 Boas práticas

No desenvolvimento da aplicação para fornecer uma melhor experiência ao utilizador seguiu-se os seguintes Princípios de Usabilidade:

- **Learnability**: Facilidade com que novos utilizadores podem iniciar uma interação eficaz e alcançar máxima performance.
  - **Predictability**: Princípio, aplicado, como por exemplo, os formulários de inserção de dados, nos quais, por exemplo, no caso do email só é possível inserir *strings* com formato de email válido.

  - **Familiarity**: A agenda de marcação de treinos pode ter uma vista diária, semanal ou mensal e a forma de inserir a data do treino a agendar é familiar ao formato utilizado no dia-a-dia.

- **Flexibility**: A Flexibility é um princípio que diz respeito à multiplicidade de formas pelas quais o utilizador e o sistema trocam informação.
  - **Dialogue Initiative**: Para qualquer listagem de entidades existe sempre uma *searchbox* que concede outra forma de encontrar uma entidade.
  - **Customizability**: É possível selecionar o número de treinos que se pretende visualizar na tabela e ainda escolher por que ordem (por exemplo, os favoritos em primeiro lugar ou por ordem alfabética, etc.).

- **Robustness**: A Robustness refere-se ao nível de suporte fornecido ao utilizador na determinação do êxito e na avaliação do comportamento direcionado a objetivos.
  - **Responsiveness**: Após o sistema remover um treinador, este lança uma notificação com o resultado dessa operação. Por exemplo, após atualizar valores como o peso, o sistema informa o utilizador do sucesso da operação.
  - **Observability**: Ao executar um treino, o sistema notifica com um som quando este termina o seu período de descanso e pode iniciar o próximo bloco de exercícios; a página relativa a efetuar um treino contém cronómetros de forma a guiar o utilizador no tempo de cada exercício; o utilizador tem perceção visual dos treinos agendados na sua agenda.

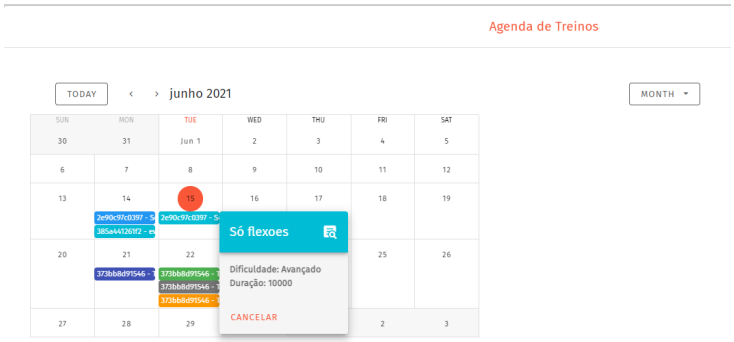

Figura 22: Boas práticas: Agenda relativa ao mês.

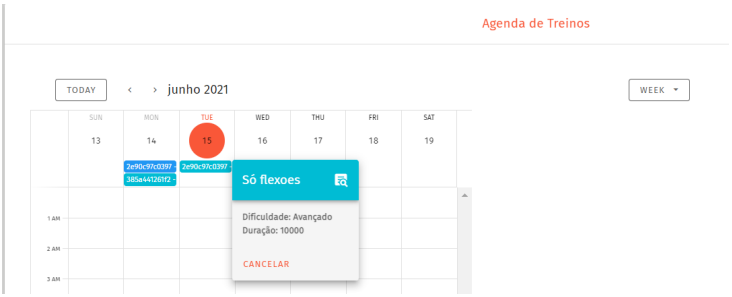

Figura 23: Boas práticas: Agenda relativa à semana.

Figura 24: Boas práticas: Cronómetros durante o treino.

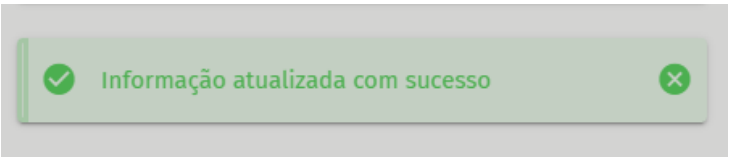

Figura 25: Boas práticas: Atualização de peso.

Figura 26: Boas práticas: Verificação de remoção.

Figura 27: Boas práticas: Notificação remoção.

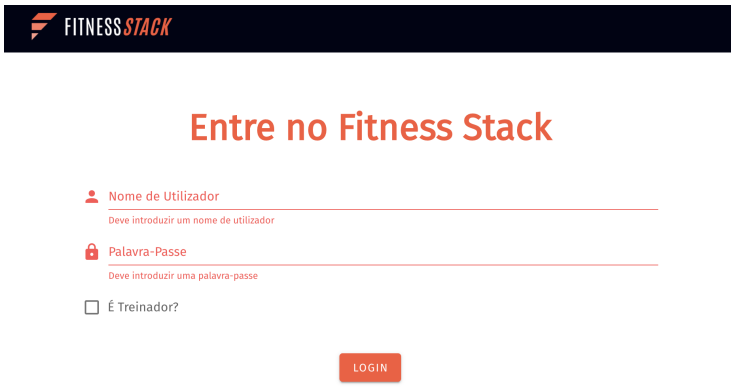

Figura 28: Boas práticas: *Login*.

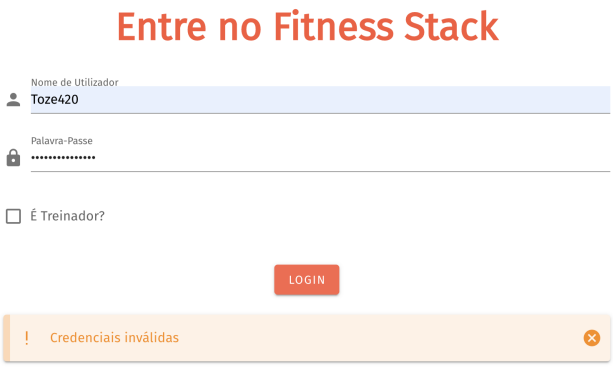

Figura 29: Boas práticas: Credenciais inválidas.

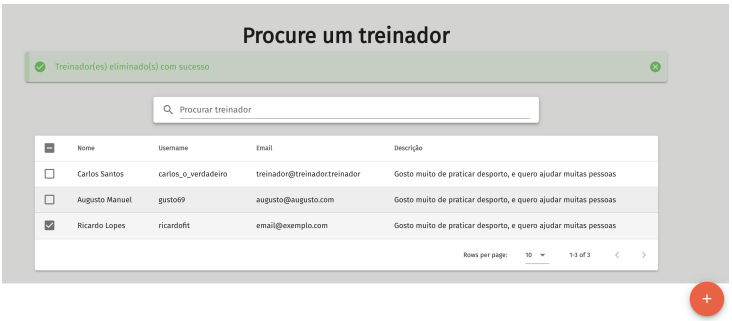

Figura 30: Boas práticas: Notificação remoção.

3.4 *Backend*

Como referido anteriormente, no *backend* utilizou-se Spring e Hibernate, duas *frameworks* que simplificaram e automatizaram o desenvolvimento da aplicação.

3.4.1 *Controllers*

Os *controllers* sãos componentes do *backend* responsáveis por fornecer a API. Recebendo os pedidos do cliente executam as operações correspondentes para calcular o que foi pedido e respondem com o resultado. De forma a manter a aplicação simples e de fácil compreensão, optou-se por fazer a seguinte divisão de *controllers*:

- **AdministradorController**: Rotas para o administrador.

- **TreinadorController**: Rotas para o treinador.
- **UtilizadorController**: Rotas para o utilizador.
- **LoginController**: Toda a lógica de *login*.
- **RegisterController**: Toda a lógica de registo.
- **TreinoController**: Rotas para acesso aos treinos.
- **ExercicioController**: Tudo relacionado com a criação e remoção de exercícios.
- **MarcacaoController**: Rotas para agendamento de treinos.
- **MediaController**: Rotas para o *frontend* pedir imagens/vídeos/áudios.

3.4.2 *Beans*

Os Java Beans são pequenas componentes do sistema, cuja função é a execução de tarefas dos clientes, sendo que cada método dos *beans* representa um *use case*. De forma a separar a lógica da aplicação que faça sentido no contexto da nossa aplicação, dividiu-se a aplicação nos seguintes *beans*:

1. **gestao_categorias**: Classe das categorias de um treino.
2. **gestao_conteudomedia**: Para tratar dos vídeos, imagens e áudios.
3. **gestao_contratos**: Classe chamada sempre que se tem de criar, aceitar ou recusar um contrato.
4. **gestao_exercicios**: Gestão dos exercícios da aplicação.
5. **gestao_treinos**: Gestão dos treinos da aplicação.
6. **gestao_marcacoes**: Responsável por fazer a gestão do agendamento de treinos aos utilizadores.
7. **gestao_treinadores**: Todas as operações relevantes à entidade treinador.
8. **gestao_utilizadores**: Todas as operações relevantes à entidade utilizador.
9. **gestao_administradores**: Todas as operações relevantes à entidade administrador.
10. **gestao_verificacoes**: Responsável pela criação e verificação dos *tokens*.

3.4.3 Base de dados

Para a base de dados optou-se por usar o Hibernate, que é uma ferramenta bastante popular, com bastante suporte de *ORM mapping* e que facilita muito os acessos à base de dados. Como sistema de base de dados, escolheu-se MySQL por ser um um sistema com o qual o grupo já desenvolveu alguns projetos passados e sente-se confortável, para além de ajudar a manter uma estrutura bem definida das entidades e as relações entre elas.

4 *Deployment*

Nesta secção aborda-se todos os assuntos relacionados com a colocação da aplicação a funcionar com acesso para todos, ou seja, colocar a aplicação na *internet*.

4.1 Considerações Iniciais

Para fazer o *deployment* utilizou-se a Google Cloud, que permitiu criar várias máquinas, separar servidores aplicacionais, servidores de *web* e bases de dados.

Na Google Cloud optou-se por criar máquinas para cada serviço, ao invés de *containers*, uma vez que o grupo valoriza a *performance* das máquinas. Além disso, deste modo, a preparação de criar uma nova instância de um dos componentes torna-se bastante simples, o que desvaloriza um pouco a elasticidade dos *containers*, no sentido de aumentar ou reduzir máquinas de uma dada componente, neste projeto.

4.2 Arquitetura

Tendo em conta as várias componentes da aplicação, implementou-se a seguinte arquitetura:

A) 1 máquina com o Nginx para ser a entrada da aplicação e a qual os utilizadores se vão conectar;

B) 2 máquinas com *frontend*;

C) 1 máquina a balancear os pedidos do *frontend* para o *backend*;

D) 1 balanceador a intersetar os pedidos do *frontend* ao *backend* e a distribuir a carga pelos servidores do *backend*;

E) 2 máquinas com *backend*;

F) 2 máquinas a correr *master* e outra *slave*, tendo o *slave* a fazer leituras e escritas e o *slave* só leituras.

Desta forma, foi possível ter alguma resiliência ao nível dos servidores *web*, servidores aplicacionais e base de dados (se a *master* for abaixo a *slave* pode assumir o seu papel). Os balanceadores são pontos únicos de falha, pois caso eles falhem, a aplicação pára de funcionar corretamente. O que deveria ser feito seria ter uma réplica que assumisse esse papel, mas montar um sistema desses na Google Cloud é algo complicado. Outra alternativa seria utilizar o próprio balanceador da Google, mas como se trata de um ambiente de testes e dado que não é possível desligar esse balanceador, estando sempre a gastar dinheiro por hora, não se considerou relevante utilizá-lo.

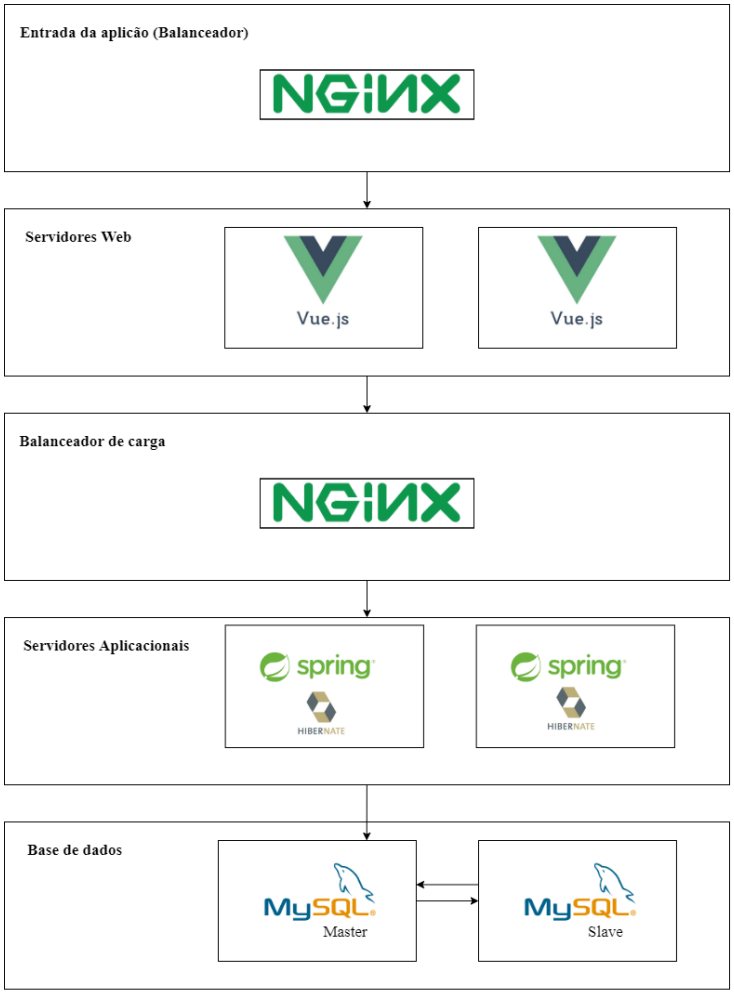

Figura 31: *Deployment*: Arquitetura.

5 Análise de carga

5.1 Considerações iniciais

A análise de carga à aplicação é uma fase do projeto de alta importância, visto que é possível verificar e analisar a quantidade de clientes que o sistema suporta ou tentar encontrar funcionalidades do sistema que poderiam estar melhor implementadas e que poderão ser melhoradas.

5.2 Testes de carga

Para a realização dos testes usou-se a ferramenta JMeter, que permite simular vários clientes a fazer pedidos HTTP à aplicação desenvolvida.

Decidiu-se, então, realizar três tipos de testes com 100 e 500 clientes:

1. GET agenda do utilizador.
2. GET lista todos os treinos.
3. GET perfil de um utilizador.

5.2.1 GET Agenda do utilizador

Para simular vários clientes a pedirem a sua agenda, como o utilizador em concreto não afeta em nada os resultados fizeram-se todos os testes à mesma agenda.

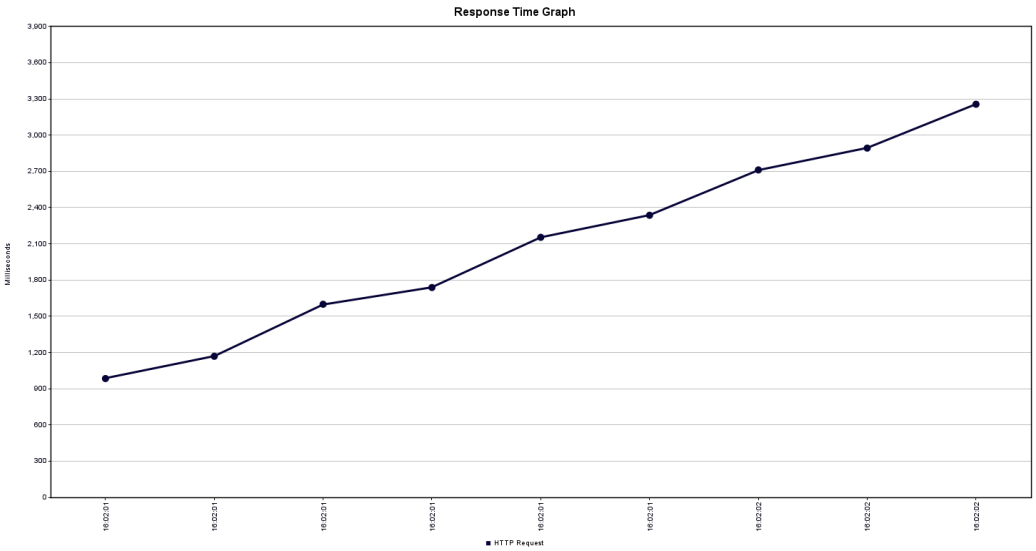

Figura 32: GET Agenda do utilizador: 100 clientes.

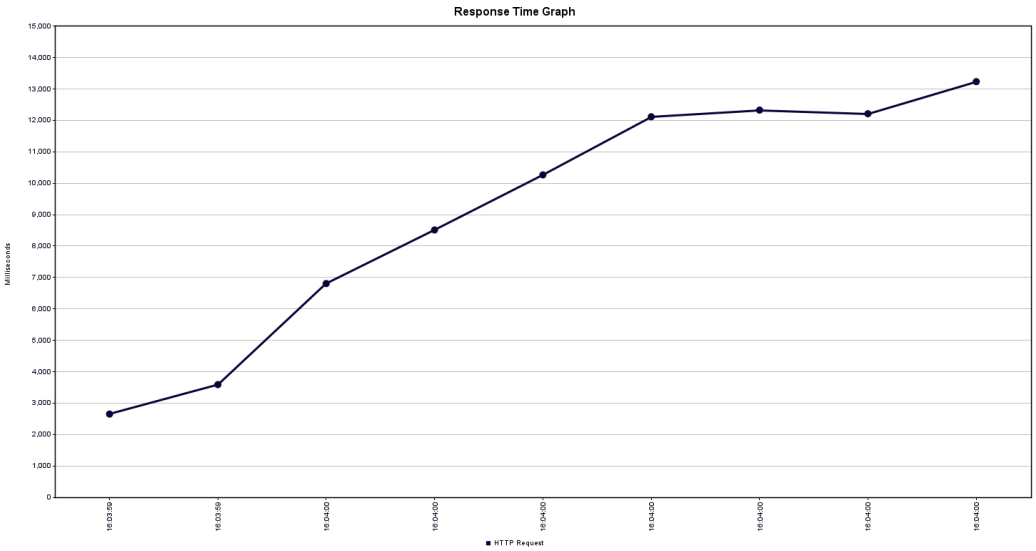

Figura 33: GET Agenda do utilizador: 500 clientes.

Olhando para os dois gráficos, é possível perceber que com 500 clientes já existe um tempo de resposta bastante elevado e que este pode ser o limite da nossa aplicação.

5.2.2 GET Listar todos os treinos

Este treino foi realizado com 100 treinos na base de dados. O objetivo é simular clientes a pedirem esta lista de treinos.

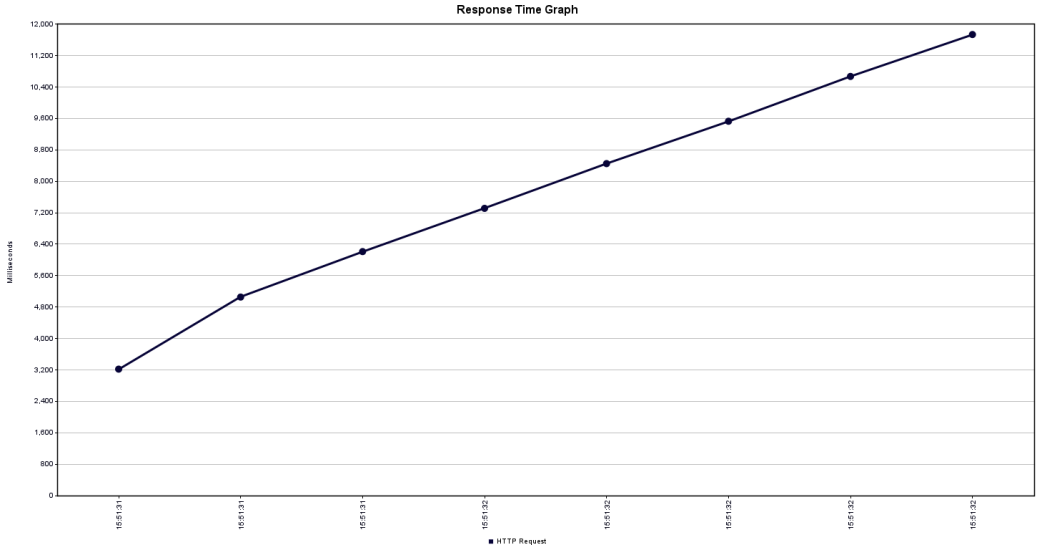

Figura 34: GET Listar todos os treinos: 100 clientes.

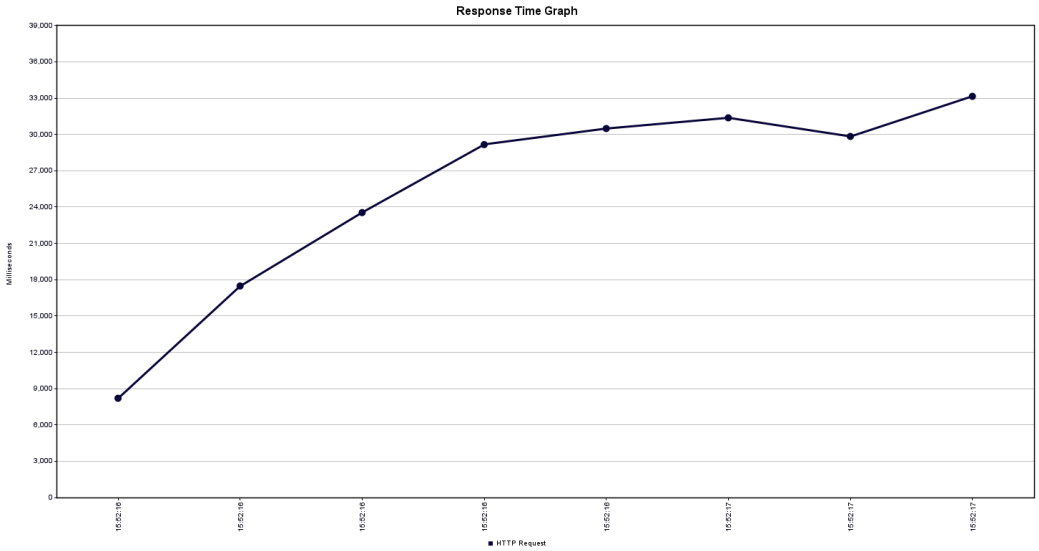

Figura 35: GET Listar todos os treinos: 500 clientes.

Com 500 clientes a fazer o pedido, um cliente pode esperar até 43s, o que simplesmente não é razoável. Isto pode ser devido ao listar os treinos, devolverem-se todos os treinos. Se com apenas 100 treinos se obtém este valor elevado, o que seria mais sensato seria implementar paginação do lado do *frontend*, e pedir ao *backend* apenas uma página de cada vez. Para verificar esta solução, ainda se realizou este pedido GET com apenas 10 treinos na

base de dados. Como é possível observar no gráfico abaixo, a paginação seria uma melhoria bastante relevante na nossa aplicação.

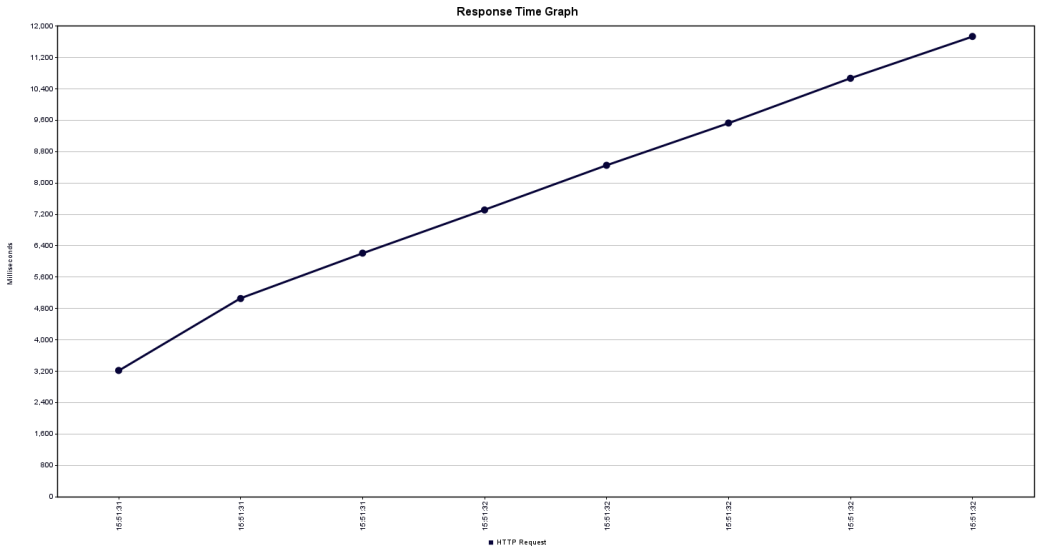

Figura 36: GET Listar todos os treinos: 500 clientes, mas com apenas 10 treinos.

5.2.3 GET Perfil de um utilizador

O pedido do próprio perfil é uma operação que todos os utilizadores fazem, pois esta é a página que recebem após efetuarem *login*, tornando-se assim um teste bastante relevante.

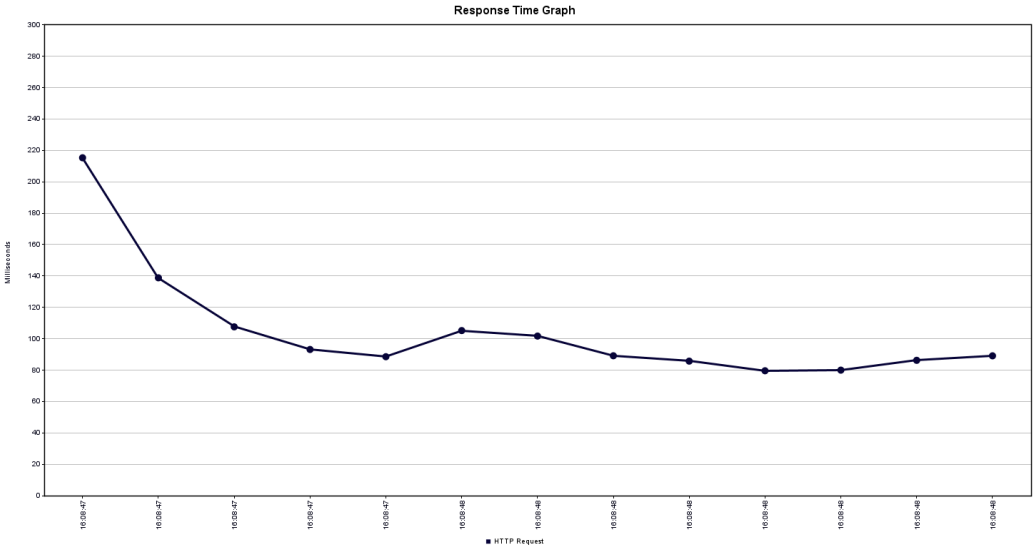

Figura 37: GET Perfil de um utilizador: 100 clientes.

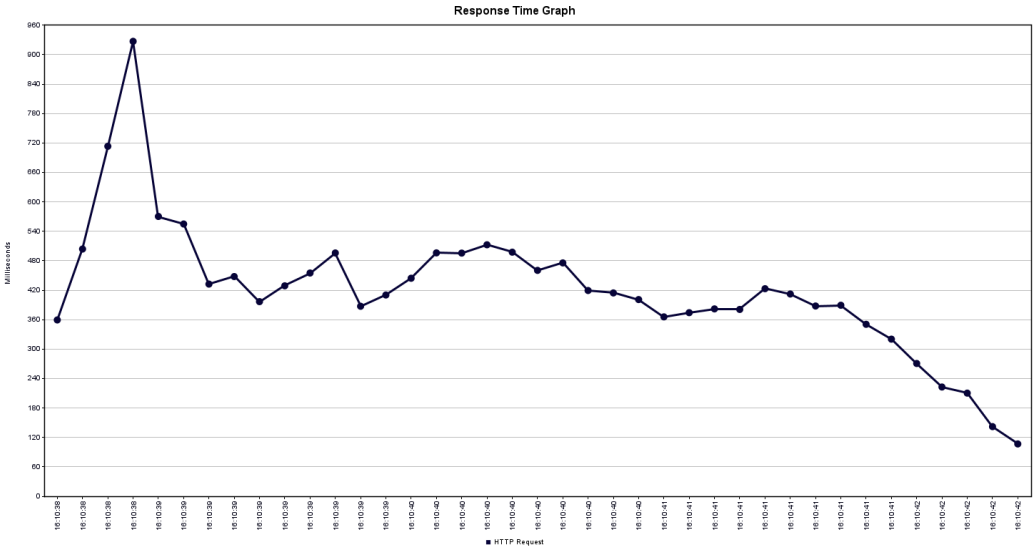

Figura 38: GET Perfil de um utilizador: 500 clientes.

Com os gráficos obtidos foi possível concluir que o grupo obteve tempos bastantes bons, os melhores comparando com os outros testes e que não é sobre esta funcionalidade que se deve tentar melhor a performance.

6 Conclusão

O presente trabalho prático consistiu no desenvolvimento de uma plataforma para prática de exercício físico, tendo-se conseguido alcançar todos os objetivos propostos.

Algumas melhorias para trabalhos futuros seriam a implementação das operações CRUD para todas as entidades do sistema, dado que nesta versão da aplicação não se implementou a operação de editar para todas as entidades constituintes do sistema e ainda a implementação de paginação em todas as listas para aumentar significativamente a performance da aplicação. Outro aspeto a melhorar seria a permitir a edição de treinos e exercícios, remoção de exercícios mesmo quando estes estão associados a treinos e também permitir a integração com calendários populares, como por exemplo, o Google Calendar, o que iria aumentar a qualidade de gestão dos treinos dos utilizadores.

Em suma, conclui-se que os conteúdos lecionados nas unidades curriculares de Arquiteturas Aplicacionais e de Sistemas Interativos foram consolidados.