WorkoutPal - Registo de Atividade e Alimentação

Henrique Fontes, 48295
Aureliu Iurcu, 48285
Miguel Agostinho, 48338

Orientador: Engenheiro Paulo Pereira

Relatório final para Projeto e Seminário
Licenciatura em Engenharia Informática e de Computadores

Maio 2024

Resumo

A *WorkoutPal* é uma aplicação de *fitness* e nutrição, onde os utilizadores podem pesquisar exercícios, criar planos de treino, registar a alimentação e acompanhar as suas metas nutricionais. O sistema integra uma aplicação cliente destinada a dispositivos móveis *Android* e *iOS*, além de um *backend* para gestão de dados e comunicação segura entre o cliente e o servidor.

Durante o desenvolvimento do projeto, enfrentámos diversos desafios, incluindo a gestão eficiente do tempo e a utilização de alguns *packages* não suportados pela *framework Expo*. Foram abordados conceitos técnicos, como a implementação de algoritmos para tratamento de dados, autenticação de utilizadores, e técnicas de segurança para garantir a integridade das informações armazenadas.

O projeto abrange o uso de tecnologias como *Express.js* para o *backend*, *MongoDB* para a base de dados e *React-Native-Expo* para o frontend.

Palavras-chave: *fitness*; nutrição; aplicação móvel; *Android*; servidor; *Express.js*; *MongoDB*; *React-Native*; autenticação.

Abstract

WorkoutPal is a fitness and nutrition app, where users can search for exercises, create training plans, record their nutrition and track their nutritional goals. The system integrates a customer application designed to Android and iOS mobile devices, in addition to a backend for data management and secure communication between the client and the server. During the development of the project, we faced several challenges, including efficiently managing the time and the use of some packages not supported by the Expo framework. Technical concepts were covered such as the implementation of algorithms for data processing, user authentication, and security techniques used to guarantee the integrity of the stored information. The project covers the use of technologies such as Express.js for the backend, Mon- goDB for the database and React-Native-Expo for the frontend.

Keywords: fitness; nutrition; mobile application; Android; server; Express.js; MongoDB; React-Native; authentication.

Índice

Índice de Figuras

1 Introdução

Neste capítulo é discutido o enquadramento sobre o qual se estabelece este projeto, os seus objetivos, tal como a forma como se encontra organizado este relatório.

1.1 Enquadramento

Embora a procura por um estilo de vida saudável sempre tenha sido uma prioridade para alguns, a sua importância tem aumentado diariamente como um contrapeso à tendência crescente para um estilo de vida cada vez mais sedentário. Para isso é reconhecido que a prática regular de exercício físico, aliada a uma alimentação equilibrada, é essencial para a manutenção da saúde e bem-estar.

É reconhecido também que a maior dificuldade encontrada por quem procura um estilo de vida mais saudável é a motivação e disciplina necessária para manter este compromisso de forma consistente.

É essencial que, de forma a auxiliar quem procura combater estas dificuldades, sejam criadas soluções que facilitem o tempo dedicado à adoção destes hábitos saudáveis. Para que estas soluções possam ser procuradas de forma regular é importante que sejam cativantes, de fácil acesso e adaptáveis às necessidades únicas de cada um. Além disso, é importante que estas soluções promovam uma sensação de progresso e realização, sustentando assim a motivação ao longo do tempo.

1.2 Objetivos

A aplicação *WorkoutPal* apresenta-se como uma solução para estas questões. Com o foco de cobrir ambas as áreas de *fitness* e alimentação, a ideia base deste projeto é criar, como o nome indica, um parceiro de treino para o utilizador, na forma de uma aplicação de telemóvel.

A componente de exercício físico da aplicação é abordada através da criação de planos de treino personalizados pelo utilizador, com toda a informação relevante sobre os exercícios em questão. Desta forma, para além de instruir o utilizador sobre como realizar os exercícios, a aplicação mantém um registo dos seus planos de treino, que podem ser consultados e modificados posteriormente.

Com o intuito de procurar facilitar a adoção de uma dieta saudável, a aplicação permite o registo de alimentos consumidos no dia a dia pelo utilizador, fornecendo também informação sobre os nutrientes que os constituem. Assim, pretende-se equipar o utilizador com um conhecimento mais profundo dos alimentos que consome, auxiliando assim o utilizador na tomada de decisões mais conscientes e fundamentadas.

A noção de progresso é um elemento crucial na gestão da motivação. A abordagem utilizada para a resolução desta questão passa pela apresentação de estatísticas do utilizador.

Para ser possível associar um utilizador aos seus registos, a aplicação dispõe de funcionalidades de autenticação, funcionalidades estas que se revelam necessárias apenas em determinadas situações.

O ângulo que é procurado é o de colmatar as questões de procura de uma vida saudável como um todo. No entanto, como os principais componentes da aplicação

funcionam de forma independente, é dada total liberdade ao utilizador para utilizar apenas a vertente que pretender da aplicação.

1.3 Estrutura do Documento

2 Análise do Problema

Neste capítulo, serão apresentados os conceitos essenciais para a compreensão da solução desenvolvida, bem como uma descrição dos elementos que a constituem. Serão delineados os intervenientes na plataforma, a informação que os descreve e identifica, e as ações disponíveis para cada interveniente. A análise destes elementos permite uma melhor visualização do problema que a *WorkoutPal* se propõe resolver, destacando a importância da personalização de planos de treino, da monitorização nutricional e da interação do utilizador para promover um estilo de vida saudável e equilibrado.

2.1 Conceitos Fundamentais

Os conceitos fundamentais da aplicação *WorkoutPal* envolvem a personalização de planos de treino, a monitorização nutricional e a interação com o utilizador.

Personalização de Planos de Treino: A *WorkoutPal* oferece aos utilizadores a capacidade de criar e personalizar planos de treino que se ajustam às suas necessidades e objetivos específicos. Esta personalização é crucial para auxiliar o utilizador no aumento do conhecimento dos exercícios que pratica nos treinos e garantir que os exercícios são apropriados para o nível de condicionamento físico do utilizador. A aplicação permite a seleção de exercícios variados, promovendo uma abordagem flexível e adaptável ao treino físico.

Monitorização Nutricional: A inclusão de funcionalidades de monitorização nutricional é um dos pilares da *WorkoutPal*. Os utilizadores podem registar alimentos através de um sistema de leitura de código de barras, pesquisar informações nutricionais detalhadas e acompanhar o consumo diário de calorias. Estas ferramentas permitem um controlo rigoroso da dieta, ajudando os utilizadores a fazer escolhas alimentares informadas e equilibradas, essenciais para o alcance de um estilo de vida mais saudável.

Participação e Interação do Utilizador: Os utilizadores da *WorkoutPal* são indivíduos que procuram melhorar a sua saúde e bem-estar através de uma abordagem equilibrada que combina exercício físico e nutrição. Estes podem criar os seus planos de treino, e, personalizá-los conforme as suas necessidades. A interação com a aplicação inclui a atualização de informações pessoais, a definição de metas e o acompanhamento do progresso, proporcionando um ambiente interativo e motivador.

Estes conceitos fundamentais estabelecem a base da *WorkoutPal*, promovendo a personalização, o controlo nutricional e a integração de funcionalidades de bem-estar. A abordagem holística visa apoiar os utilizadores na sua jornada para um estilo de vida mais saudável e equilibrado.

2.2 Funcionalidades

No que toca a autenticação o utilizador pode:

- **Criar conta**, para isso o mesmo deve aceder ao menu de opções e clicar no botão de *signup*, em consequência é redirecionado para o ecrã onde pode fazer o cadastro. O utilizador necessitará de preencher os campos com os seus dados e clicar no botão de *sign up*.

- **Entrar com uma conta já existente**. O utilizador tem acesso a esta opção através do menu de opções onde encontrará o botão de *log in*. Ao clicar no botão mencionado é redirecionado para o ecrã de *log in* onde é preciso preencher os campos com os seus dados e clicar no botão *Log In*.

- **Fechar sessão**. Para isso o utilizador precisará estar autenticado. Caso isso se verifique, o utilizador terá acesso ao botão *logout* no menu de opções. Aquando do pressionar desse botão, o mesmo será desconectado.

Para pesquisar exercícios o utilizador:

- O utilizador deve navegar até à página de pesquisa e inserir na barra de entrada apresentada, o nome do exercício que pretende pesquisar. Após escrever o nome, o utilizador carrega no botão "*Search*" e aparecem os resultados.

Nos planos de treino, o utilizador pode:

- **Criar planos de treino**. Para isso, o mesmo deve navegar para a página dos planos de treino e carregar no botão para criar planos de treino, que redirecionará o utilizador para o *modal*(em *iOS*), ou para a página de criação de planos de treino(em *Android*) onde serão pedidos o nome e descrição, opcional, do plano de treino. Após fornecer essses dados, é só carregar no botão de adicionar.

- Para **adicionar um exercício ao plano de treino**, o utilizador deve aceder à página do exercício e carregar no botão para adicionar ao plano de treino. Após carregar no botão o utilizador é redirecionado para um *modal*(em *iOS*), ou para a página de adição(em *Android*), onde aparecem todos os planos associados à sua conta. Basta o utilizador carregar num plano que o exercício é adicionado. É possível adicionar o mesmo exercício a vários planos de treino, por isso, quando o utilizador estiver satisfeito, carrega na seta para voltar para trás em *Android* ou fecha o *modal* em *iOS*.

- Para **remover um exercício do plano de treino**, o utilizador deve navegar para a página do plano de treino que deseja retirar o exercício, e carregar no botão abaixo do exercício.

- **Registar os planos de treino feitos num determinado dia**. O utilizador deve aceder à página do plano de treino realizado e carregar no botão do canto superior direito para adicionar o plano de treino ao registo diário.

Em relação á vertente da nutrição o utilizador pode:

- **Consulta do registo diário**. O utilizador, a partir do ecrã principal da alimentação acedido através do botão correspondente, consegue consultar o registo de alimentos, calorias e nutrientes.

- **Pesquisa de alimentos**. O utilizador, ao pressionar o botão de adicionar um alimento presente no ecrã do registo diário, é redirecionado para o ecrã de pesquisa de alimentos. Aqui pode fazer a pesquisa a partir da leitura de um código de barras ou da pesquisa pelo nome. A pesquisa feita através da leitura do código de barras funciona através de um botão, que ao ser pressionado solicita a permissão de uso da câmera do utilizador. Após dada a permissão, a visualização da câmera do utilizador é ativada. Após a leitura de um código de barras o utilizador é diretamente redirecionado para o ecrã do elemento correspondente. A pesquisa por um alimento pelo nome é feita através da inserção do mesmo na barra de pesquisa. Após a submissão são apresentados os resultados, que ao serem pressionados reencaminham o utilizador para o ecrã correspondente.

- **Registo de um alimento**.Para registar um alimento o utilizador navega para o ecrã referente ao mesmo, que pode ser acedido de duas formas: através do ecrã de pesquisa, tal como pressionando num dos alimentos pertencentes ao seu consumo diário. Este ecrã, para além de informações relevantes a um determinado alimento, apresenta também um botão de registo. Ao ser pressionado, para além de registar o alimento redireciona o utilizador para o ecrã do seu consumo diário.

As funcionalidades futuras planeadas incluem a integração com dispositivos *wearables* para uma monitorização mais precisa da atividade física e frequência cardíaca, recomendações personalizadas de exercícios e planos de treino com base no perfil e histórico do utilizador, funcionalidades sociais, como desafios, grupos de apoio e partilha de progresso, para promover a motivação e a interação entre utilizadores, e integração com aplicações de saúde e bem-estar populares para uma experiência mais abrangente e conveniente.

A monitorização nutricional inclui o registo de alimentos consumidos através da leitura de código de barras ou pesquisa manual, acesso a informações nutricionais detalhadas sobre os alimentos, incluindo calorias, informações nutricionais, e acompanhamento do consumo diário de calorias e nutrientes para ajudar os utilizadores a fazer escolhas alimentares informadas e equilibradas. Também é possível consultar um histórico de alimentos consumidos em dias passados para análise e acompanhamento do progresso.

O acompanhamento do progresso envolve o registo do peso atual do utilizador para monitorizar o progresso ao longo do tempo, visualização gráfica do progresso de peso para identificar tendências e padrões, e a opção de registar informações sobre os planos de treinos realizados, permitindo aos utilizadores acompanhar a sua evolução na força e resistência.

3 Descrição da Solução

Neste tópico irão ser abordadas de forma mais detalhada a interação entre os vários componentes do sistema, os elementos chave para a implementação dos mesmos, bem como todas as tecnologias utilizadas para o desenvolvimento e as razões para essas escolhas.

3.1 Arquitetura do Sistema

A arquitetura do sistema implementado baseia-se no modelo cliente/servidor onde a aplicação móvel, desempenhando o papel de cliente, comunica com o servidor através de pedidos *HTTP*.

O servidor é responsável por criar as conexões e comunicar tanto com a base de dados como com as duas *API's* utilizadas. O mesmo também detém a responsabilidade de disponibilizar tanto formas de autenticação ao cliente como autorizar o mesmo a utilizar os recursos disponíveis (Figura 1).

As *API*'s escolhidas foram a *OpenFoodFacts API* para alimentação/nutrição, uma vez que é das *API's* mais completas e fiáveis da área, e a *ExerciseDB API* visto que a mesma proporciona uma base de dados com mais de 1300 exercícios,informação simples de filtrar sobre cada exercício em específico e uma representação visual dos mesmos em formato de *GIF*. Ainda em relação a *ExerciseDB API*, foi tomada a decisão de fazer a clonagem diária de todos os exercícios para uma base de dados relacionada ao projeto devido a limitação de pedidos imposta.

No contexto do acesso do cliente aos recursos disponibilizados pelo servidor, é importante distinguir entre recursos públicos e privados. Os recursos públicos estão disponíveis sem a necessidade de qualquer autorização. No entanto, para os recursos que requerem autorização, esta é concedida através de um *token Bearer*, que é fornecido ao cliente no momento da autenticação. Desta forma garante-se segurança e a integridade dos dados trocados entre o cliente e o servidor.

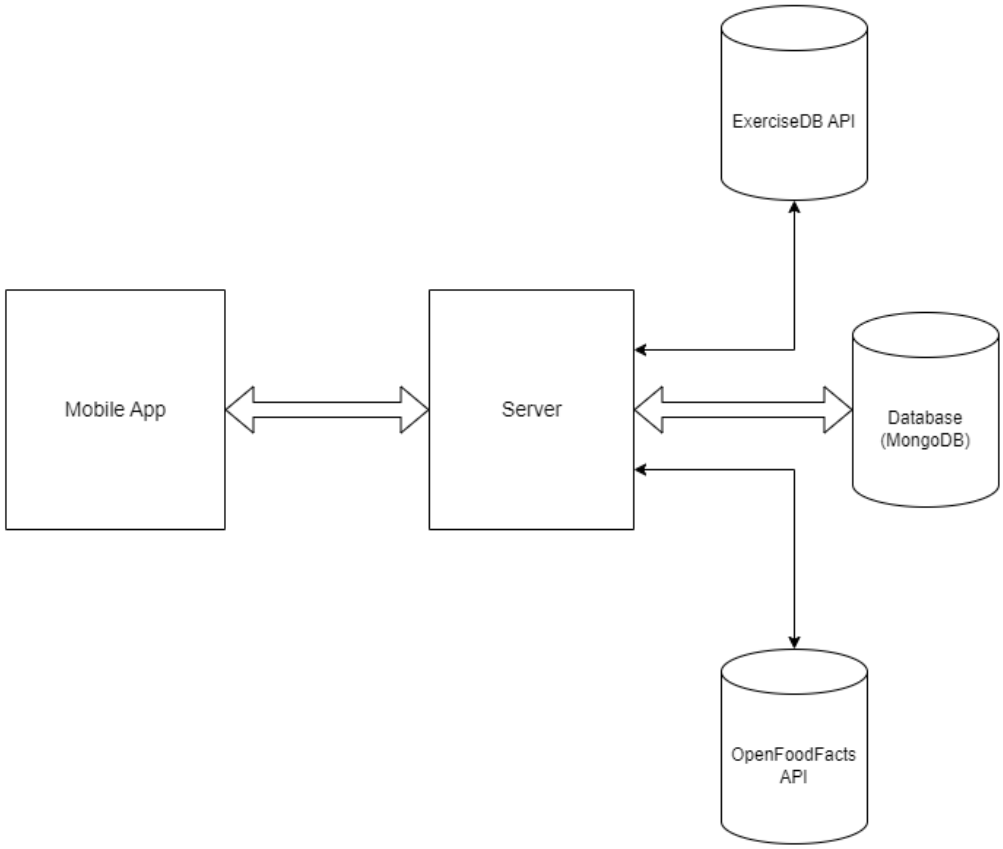

Figure 1: Arquitetura do Sistema

3.2 Tecnologias Utilizadas

Este subtópico irá abordar as várias tecnologias utilizadas para a implementação do cliente (aplicação móvel) como as que ajudaram a implementar o servidor. Também irão ser enunciados as várias razões que levaram a decisão para o uso de cada uma das tecnologias.

3.2.1 Servidor

Os motivos que levaram a decisão do uso de *NodeJS* em vez de qualquer outro ambiente de execução foram:

- Possibilitar o uso da linguagem *Typescript* que também é usada do lado do cliente facilitando dessa forma a gestão e manutenção do código
- Disponibilização do gestor de módulos *NPM* (*Node Package Manager*) que consegue proporcionar uma alta variedade de bibliotecas que usadas na implementação do servidor.
- Fácil interação com *Nginx*. Apesar de usar apenas uma *thread*, caso necessário, o servidor desenvolvido com *NodeJS* é facilmente escalável através do serviço disponibilizado por *Nginx*.

Typescript foi a linguagem preferida para o código visto que a mesma, no contexto *NodeJS*, é a linguagem que disponibiliza sistema de tipos ajudando assim na criação de código mais legível, maior facilidade em criar código *Object Oriented* e fácil manutenção e controlo.

Pelo facto de se ter conhecimento sólido sobre a *framework ExpressJS* e ser de uso simples, escolheu-se a mesma para a implementação das rotas.

Escolheu-se *MongoDB* em vez de uma base de dados relacional, como *PostgreSQL*, principalmente pelo facto do projeto não possuir muitas relações nos dados que são guardados.

Também o facto de ser uma base de dados com menos custos de manutenção e de maior facilidade de acesso influenciou a decisão. Para a comunicação com o *MongoDB* de uma forma mais segura foi tirado proveito dos tipos disponibilizados pela linguagem usada e da *framework Mongoose* para a representação dos documentos em causa através de esquemas.

3.2.2 Aplicação Móvel

Para a implementação da aplicação móvel foi usado *NodeJS* como ambiente de execução principalmente por 2 razões essenciais:

- Possibilitar o uso da linguagem *Typescript* usada pelo servidor também, facilitando dessa forma a gestão e manutenção do código.
- Disponibilizar o gestor de módulos *NPM (Node Package Manager)* que consegue proporcionar uma alta variedade de bibliotecas que são usadas pelo *WorkoutPal*.

React Native foi a *framework* preferida pela principal razão de possibilitar desenvolvimento de aplicações móveis tanto para *iOS* como *Android*.

A linguagem escolhida para o efeito foi *Typescript* pelos mesmos motivos enunciados do lado do servidor, porém também teve influencia o facto da aplicação ser criada em *React Native* sendo essa uma *framework* usada apenas nas linguegens *Javascript* e *Typescript*.

Uma das ferramentas mais importantes usadas para o desenvolvimento da aplicação foi *Expo* por várias razões:

- Facilidade em criar um projeto *React Native*
- Possibilidade de testar o código nativo tanto em dispositivos *Android* como *iOS*
- Suportada pela linguagem *Typescript*
- Diversas *API's* de acesso às funções de disponibilizadas por qualquer dispositivo móvel
- Facilidade na criação de compilações tanto para a *App Store* como para *Google Play Store*.

4 Implementação das Funcionalidades

Neste tópico irá ser descrita a forma como cada operação disponível foi implementada tanto do lado do servidor como da aplicação móvel.

4.1 Autenticação

Em relação à autenticação irão ser enunciadas as operações disponibilizadas pelo servidor tal com a implementação de cada uma em particular.

É importante mencionar que a autenticação do lado do servidor é dividida em 3 partes que irão analisar e processar um pedido, sendo estas:

- ***Data***, representado pela classe *AuthData*, responsável pela comunicação com a base de dados para procura, atualização e criação de utilizadores.
- ***Services***, representado pela classe *AuthServices*, que recebe a classe *AuthData* como dependência, e responsável por implementar a lógica das operações realizando as diferentes verificações e filtragens necessárias.
- ***API***, representado no projeto pela classe *AuthAPI*, que recebe as classes *AuthData* e *AuthServices* como dependências, e responsável por receber os pedidos, chamar os métodos disponibilizados pelas outras classes para processar esses mesmos pedidos e por fim criar uma resposta adequada ao cliente consoante o resultado obtido do processamento realizado.

A (Figura 2) representa a interação entre os componentes enunciados.

A aplicação cliente por sua vez implementa 2 ecrãs, um para o *signup* e outra para *login* e interagem com o domínio e serviços. O domínio contendo as representações de utilizador *User*, de uma reposta de sucesso após um pedido realizado ao servidor *AuthResponse* e de uma resposta em caso de insucesso *ResponseError*. Os serviços por sua vez incluem todos os métodos que representam as ações realizadas nos ecrãs e a lógica associada a cada ação.

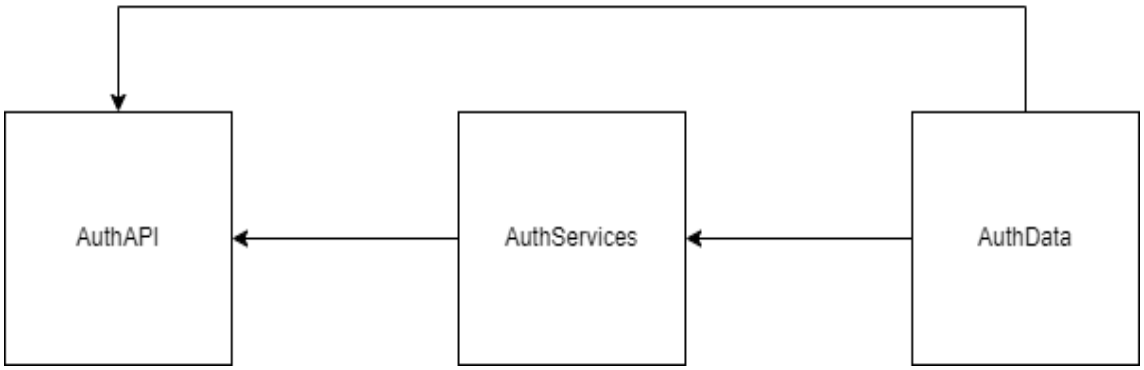

Figure 2: Interação entre os componentes de autenticação

4.1.1 *Signup*

A aplicação cliente ao usar a operação de *signup*, disponibilizada pelo servidor, precisará de fazer um pedido que inclui 3 informações importantes (nome, email, password) por cima das quais irão ser feitas verificações, como por exemplo existência de algum utilizador com o email presente no pedido, e em caso de não ser encontrado qualquer erro ou conflito, é adicionado um novo utilizador na base de dados com as informações providenciadas e enviado uma resposta de sucesso contendo um *token* de autenticação/autorização que a aplicação cliente irá usar para acesso a recursos privados posteriormente.

Tal operação acontece no ecrã criado para o efeito *SignUp* após o utilizador inserir as informações pedidas e clicar no botão de Sign Up. Após o *signup* ser concluído com sucesso o *token* retornado na reposta é guardado no armazenamento local do dispositivo que posteriormente será utilizado noutras operações implementados por outros ecrãs.

Autenticação com *Google* que internamente chama operação de *signup* do servidor

4.1.2 Login

A aplicação móvel invocando a operação de *login* necessita enviar 2 informações no pedido (email, *password*), tal como na operação anterior o pedido é verificado pelos serviços e em caso de não existir qualquer conflito acontece uma atualização do *token* de autenticação/autorização do utilizador identificado pelas informações passadas no pedido e por fim criada uma resposta tendo como conteúdo este mesmo *token*. Tal como na operação de *signup* o *token* retornado é guardado no armazenamento local para futuras operações.

No ecrã implementado para o efeito esta operção é invocada após o utilizador inserir os dados pedidos e clicar no botão de *Log In*.

Login com *Google* que internamente chama operação de *login* do servidor

4.1.3 *Logout*

WorkoutPal no menu de opções disponibiliza a opção de *logout*. Aquando do utilizador clicar na opção mencionada, é feito um pedido de *logout* ao servidor passando o *token*, presente no armazenamento local do dispositivo, como *token* de autenticação *Bearer*. Após esta ação, o servidor recebe o pedido e caso o *token* presente no mesmo pertence a algum utilizador, atualiza o campo *token* na base de dados deste mesmo utilizador para *null*, representando dessa forma um utilizador não autenticado.

4.2 *Fitness*

4.2.1 Pesquisa de exercícios

A aplicação *WorkoutPal* permite que os utilizadores pesquisem os exercícios que desejam incluir nos seus planos de treino. Ao selecionar a pesquisa de exercícios, o utilizador é redirecionado para um ecrã de pesquisa que apresenta uma barra de entrada onde o utilizador pode digitar o nome do exercício desejado. Quando o utilizador carrega no botão "*Search*" do teclado, a aplicação envia um pedido ao

servidor para obter uma lista de exercícios que contenham o termo digitado. A lista de exercícios apresentada inclui apenas o nome do exercício e o equipamento necessário. Para obter mais informações detalhadas sobre cada exercício, como a parte do corpo visada, instruções passo a passo e uma representação visual do exercício através de um *GIF*, entre outros, o utilizador precisa de carregar num dos exercícios apresentados, o que redireciona para a página do exercícios, onde se encontram informações adicionais para ajudar os utilizadores a entender melhor como realizar o exercício de maneira correta.

4.2.2 Planos de treino

Na aplicação são, também fornecidas, aos utilizadores, ferramentas para criar e personalizar os seus próprios planos de treino. Ao selecionar a opção de planos de treino, o utilizador é redirecionado para a página correspondente, onde é feito um pedido ao servidor para obter a lista de planos de treino associados à sua conta. Após receber os dados, são apresentados o nome e descrição de cada plano. Para obter mais detalhes sobre um plano específico, o utilizador pode carregar no mesmo, o que redireciona para a página do plano de treino. Nesta página, são feitos vários pedidos ao servidor para obter as informações dos exercícios incluídos no plano, identificados pelos seus *ID's*. Uma vez obtidos os dados dos exercícios, são exibidos detalhes como o *GIF* demonstrando a execução correta do movimento, o nome do exercício e o equipamento necessário. Adicionalmente, é possível carregar em cada exercício para visualizar informações mais detalhadas.

A criação de novos planos de treino é feita através de um botão na página de planos de treino, que redireciona o utilizador para um *modal* (no caso de *iOS*) ou para uma página dedicada (no caso de *Android*). Nesta interface, o utilizador pode inserir o nome do plano e uma descrição, opcional, e submeter o formulário. Ao fazer o pedido de criação, o servidor verifica se já existe um plano com o mesmo nome associado àquela conta. Caso positivo, é retornado um erro indicando que o plano já existe.

A adição de exercícios a um plano de treino é feita através de um botão na página do exercício, que redireciona para um *modal* (*iOS*) ou página (*Android*) de adição. Neste ecrã, durante a inicialização, é feito um pedido ao servidor para obter a lista de planos de treino do utilizador. Após exibir os planos disponíveis, o utilizador precisa apenas de carregar no plano ao qual deseja adicionar o exercício para que o processo seja concluído. Ao carregar num plano, é enviado um pedido ao servidor para adicionar o exercício à lista de exercícios daquele plano. O servidor verifica se o exercício já consta da lista, e em caso positivo, retorna um erro indicando que o mesmo já está da lista. Para simplificar o processo de adição a múltiplos planos, não é feito um redirecionamento automático após a adição. Isto permite que o utilizador carregue em vários planos para adicionar o mesmo exercício a cada um deles. Quando terminar de adicionar os exercícios desejados, o utilizador precisa carregar na seta de retorno para voltar à página do exercício ou, no caso de *iOS*, fechar o *modal*.

A remoção de exercícios de um plano de treino é feita através de um botão abaixo de cada exercício na página do plano. O processo envolve um pedido ao servidor para remover o exercício da lista de exercícios daquele plano. O servidor verifica se o exercício realmente consta da lista do plano. Caso contrário, retorna um erro indicando que o recurso não existe na lista. Em caso de erro, é apresentado *feedback*

ao utilizador. Em caso de sucesso, para reduzir a quantidade de pedidos ao servidor, o exercício é removido diretamente da lista armazenada na memória do dispositivo, em vez de pedir a lista toda de novo e efetuar os pedidos para obter os detalhes dos exercícios.

4.2.3 Registo de um plano de treino

A funcionalidade de registo diário dos planos de treino executados na aplicação *WorkoutPal* é uma ferramenta essencial para os utilizadores acompanharem o seu progresso de forma organizada. Ao aceder à página do plano de treino, os utilizadores encontram um botão no canto superior direito designado para o registo diário. Ao carregar neste botão, é enviado um pedido ao servidor para processar a informação. O servidor realiza uma pesquisa na lista que contém os registos diários do utilizador, incluindo os planos de treino realizados e a nutrição registada.

Caso não seja encontrado um registo para o dia em questão, o servidor cria um novo objeto "*Day*". Este objeto contém a data, um *array* vazio para a comida consumida e um *array* com o plano de treino registado. Por outro lado, se já existir um registo para o dia, o servidor apenas atualiza o *array* com os exercícios registados, mantendo assim um histórico preciso e organizado do progresso diário do utilizador na aplicação *WorkoutPal*.

4.3 Nutrição

Nesta secção é exposta a forma como as operações de pesquisa, registo, e a informação do consumo diário de alimentos são realizadas na aplicação.

4.3.1 Pesquisa de alimentos

O processo de pesquisa por um alimento começa após o utilizador pressionar o botão correspondente. O utilizador é assim redirecionado para o ecrã correspondente que apresenta um botão e uma barra de pesquisa, referentes ás opções de pesquisa: pelo nome, e pela leitura de um código de barras.

Para pesquisar um alimento pelo nome o utilizador digita o nome do alimento que procura. Ao submeter, é enviado um pedido ao servidor de forma a obter os resultados pretendidos. O servidor envia um pedido á *API* de alimentos e mapeia a informação que recebe de forma a obter o formato definido no domínio da aplicação, retornando depois os resultados da procura ao cliente. Estes resultados, que incluem o nome, os nutrientes e a imagem de cada alimento, são apresentados ao utilizador. Desta forma, o utilizador pode encontrar de maneira eficiente o alimento que procura. Após pressionar um dos resultados o utilizador é redirecionado para o ecrã do alimento correspondente.

Ao pressionar no botão referente á pesquisa pela leitura pelo código de barras pela primeira vez, é solicitada uma autorização para aceder à câmara do dispositivo. Uma vez concedida a permissão, a visualização da câmara do utilizador é ativada. Quando um código de barras é lido com sucesso, o utilizador é automaticamente redirecionado para o ecrã do alimento correspondente.

4.3.2 Registo de um alimento

O processo de registo de um alimento é efetuado no ecrã específico dedicado do mesmo. Nesta interface são fornecidos detalhes que caracterizam o alimento, incluindo o seu nome, os nutrientes mais relevantes, o conteúdo calórico e uma representação visual através de uma imagem. Além disso, caso o utilizador deseje obter mais informações, pode pressionar o botão designado para isso e mais detalhes serão exibidos.

Nesta interface está presente também um botão de registo, que ao ser pressionado envia um pedido ao servidor. É feito um pedido á base de dados para ir buscar o utilizador que fez o pedido. Caso o utilizador não seja encontrado, é retornado um erro ao cliente. Caso contrário, adiciona-se um novo alimento consumido ao registo diário do utilizador, ou se não existir registo prévio, adiciona-se um novo registo com o alimento em questão. Após isto os dados atualizados são guardados na base de dados.

Após a operação estar concluída o utilizador é redirecionado para o ecrã principal da alimentação, onde se encontra o registo diário de alimentos consumidos, com o novo elemento adicionado.

4.3.3 Informação sobre o consumo diário

A informação sobre o consumo diário pode ser encontrada no ecrã principal da alimentação, que pode ser acedido através do botão correspondente que se encontra presente no *layout* da aplicação.

Quando este ecrã é acedido um pedido é enviado ao servidor, que por sua vez acede á base de dados de forma a encontrar o utilizador em questão. Caso este não seja encontrado, um erro é retornado ao cliente, caso contrário, é feita uma procura pelo dia atual do utilizador e este é retornado ao cliente de forma a ser apresentado.

4.4 Estatística

4.4.1 Peso corporal

4.4.2 Exercícios

4.4.3 Nutricional

5 Implementação do Servidor

5.1 Arquitetura

A implementação do servidor é baseada numa estrutura *API*->Serviços->*Data*, que permite uma separação clara entre as responsabilidades de cada camada.

***API*:**

A *API* é responsável por lidar com o *request* recebido do cliente e formular a resposta. Esta é a primeira camada de abstração entre o cliente e o servidor e é responsável por:

- **Receber o *request*:** A *API* recebe o *request* do cliente e processa-o.

- **Formular a resposta:** A *API* formula a resposta HTTP baseada nos resultados da chamada aos serviços feita com os dados recebidos do cliente.

Serviços:
Os Serviços são responsáveis por chamar a *Data* e avaliar os resultados. Estes são a segunda camada de abstração entre a *API* e a *Data* e são responsáveis por:

- **Chamar a *Data*:** Os Serviços chamam a *Data* para obter os dados necessários.
- **Avaliar os resultados:** Os Serviços avaliam os resultados da chamada à *Data* e retornam o resultado ou erro, se houver.

***Data*:**
A *Data* é responsável por tratar da comunicação com a base de dados ou, no caso da comida, com a *API OpenFoodFacts*. Esta é a terceira camada de abstração é responsável por:

- **Comunicação com a base de dados:** A *Data* trata da comunicação com a base de dados, incluindo a criação, leitura, atualização e exclusão de dados.
- **Comunicação com a *API OpenFoodFacts*:** A *Data* também trata da comunicação com a *API OpenFoodFacts*, incluindo a pesquisa de informações sobre alimentos e a obtenção de dados nutricionais.

Benefícios da utilização desta arquitetura:
A estrutura *API*->Serviços->*Data* oferece vários benefícios, incluindo:

- **Separação de responsabilidades:** Cada camada tem uma responsabilidade clara e bem definida, o que facilita a manutenção e o desenvolvimento do servidor.
- **Flexibilidade:** A estrutura permite substituir facilmente qualquer das peças(*API*, Serviços e *Data*) sem afetar as restantes.
- **Reutilização de código:** A estrutura permite que o código seja reutilizado em diferentes partes do servidor, reduzindo a duplicação de código e a complexidade do sistema.

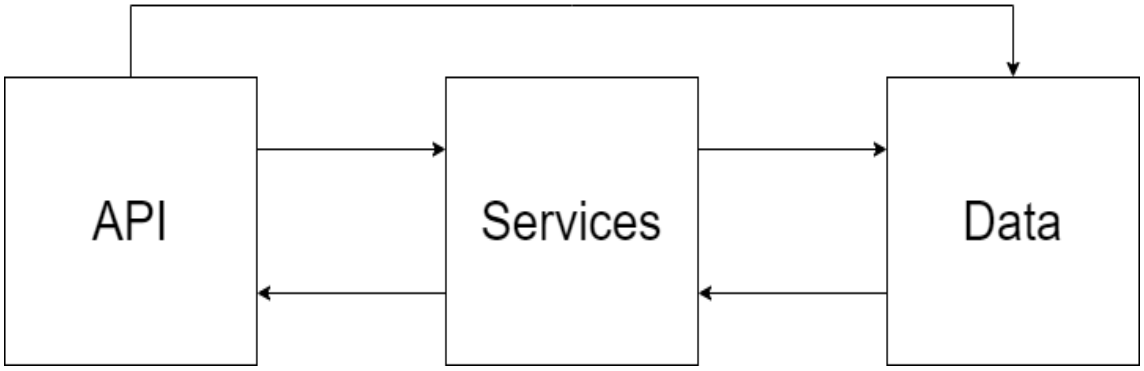

Figure 3: Arquitetura do Servidor

Em resumo, a implementação do servidor é baseada numa estrutura *API*->Serviços->*Data*, que permite uma separação clara entre as responsabilidades de cada camada e oferece benefícios como a separação de responsabilidades, flexibilidade e reutilização de código.

5.2 Comunicação com a Base de Dados

Na aplicação *WorkoutPal* a comunicação com a Base de Dados é feita através da biblioteca *Mongoose*, utilizando *Schemas* para definir a estrutura dos dados e garantir a estruturação e validação dos dados.

5.2.1 Conexão

A conexão com a Base de Dados é efetuada durante a inicialização do servidor, garantindo que a comunicação seja estabelecida antes de qualquer operação ser realizada. Além disso, a conexão é fechada automaticamente quando o servidor é desligado, garantindo que a Base de Dados seja libertada quando não está a ser utilizada. É utilizada a biblioteca *Mongoose*, que fornece uma interface simples de usar para trabalhar com a Base de Dados *MongoDB*. Para isso, criámos dois *Schemas*, "*exerciseSchema*" e "*userSchema*", que definem a estrutura dos dados que vamos armazenar.

5.2.2 Representação dos Dados

***UserSchema*:**

- username: Nome do utilizador,
- email: Email do utilizador,
- password: *Hash* com *salt* para segurança,
- token: *Token* de autenticação,
- `workout_plans`: *Array* de objetos *WorktouPlan*,
- days: *Array* de objetos *Day*

O *userSchema* define a estrutura dos dados dos utilizadores, incluindo o nome de utilizador, email, *hash* com *salt* da senha, *token* de autenticação e a lista de planos de treino e dias registados.

***ExerciseSchema*:**

- id: Identificador único do exercício.
- name: Nome do exercício.
- bodyPart: Parte do corpo trabalhada.
- equipment: Equipamento necessário.
- gifURL: *URL* do *GIF* demonstrativo do exercício.
- target: Objetivo do exercício.
- secondaryMuscles: *Array* com os músculos secundários envolvidos.
- instructions: *Array* com as instruções passo a passo do exercício.

O *exerciseSchema* define a estrutura dos dados dos exercícios, incluindo o nome, parte do corpo trabalhada, equipamento necessário, *URL* do *GIF* demonstrativo e as instruções passo a passo do exercício.

5.3 Modelo de Dados

Neste capítulo, vamos explorar o modelo de dados utilizado na aplicação *WorkoutPal*. Optámos pelo *MongoDB* devido à sua performance e por não necessitarmos de relações complexas entre os dados. Em baixo estão as estruturas de dados utilizadas e uma breve explicação dos seus campos:

***Users*:**

- id: Identificador único do utilizador.
- username: Nome de utilizador.
- email: Endereço de email do utilizador.
- password: *Hash* da password com *salt* para segurança.
- token: *Token* de autenticação.
- `workout_plans`: *Array* de objetos *WorkoutPlan*, contendo nome único, descrição e lista de exercícios.
- days: *Array* de objetos *Day*, representando o registo diário. Cada *Day* possui data, lista de alimentos consumidos (objetos *Food*) e lista de exercícios realizados.

***WorkoutPlan*:**

- name: Nome único do plano de treino.
- description: Descrição opcional do plano de treino.

- exercises: *Array* de *id's* dos exercícios no plano.

***Day*:**

- date: Data no formato "dd-mm-yyyy".
- consumedFood: *Array* de objetos *Food* com *id*, nome, calorias, proteínas, gorduras e carboidratos.
- `workout_plansDone`: *Array* de *id's* dos exercícios realizados no dia.

***Food*:**

- id: Identificador único do alimento.
- name: Nome do alimento.
- calories: Calorias do alimento.
- protein: Proteínas do alimento.
- fat: Gorduras do alimento.
- carbs: Carbohidratos do alimento.

***Exercise*:**

- id: Identificador único do exercício.
- name: Nome do exercício.
- bodyPart: Parte do corpo trabalhada.
- equipment: Equipamento necessário.
- gifURL: *URL* do *GIF* demonstrativo do exercício.
- target: Músculos visados pelo exercício.
- secondaryMuscles: *Array* com os músculos secundários envolvidos.
- instructions: *Array* com as instruções passo a passo do exercício.

Este modelo de dados é essencial para a organização e funcionamento eficiente da aplicação *WorkoutPal*, garantindo a integridade dos dados e a correta representação das informações relacionadas aos utilizadores, planos de treino, exercícios e alimentação.

5.3.1 Gestão Transacional

A gestão transacional é um processo crítico na aplicação *WorkoutPal*, pois envolve a manipulação de dados confidenciais e a necessidade de garantir a consistência e integridade dos dados. Para isso, utilizamos o *handler* "*transactionHandler*" que começa uma sessão *MongoDB* para beneficiar de "*causal consistency*", "*retryable writes*" e "*transactions*".

Criação da Sessão:

A sessão é criada utilizando a biblioteca *Mongoose*, que fornece uma interface simples para trabalhar com a Base de Dados *MongoDB*. A sessão é utilizada para estabelecer uma conexão com a Base de Dados e para executar operações de leitura e escrita.

Criação da Transação:

Após criar a sessão, é criada uma transação para executar os pedidos necessários à Base de Dados. A transação é utilizada para garantir que todas as operações sejam executadas de forma consistente e que os dados sejam mantidos num estado consistente.

***Try-Catch*:**

O *handler* "*transactionHandler*" contém um *try-catch* que permite lidar com erros que possam ocorrer durante a execução da transação. No *try*, a ação é executada e, se for bem-sucedida, o *commit* é realizado. No *catch*, os erros são apanhados e o *rollback* da transação é realizado, garantindo que os dados sejam mantidos num estado consistente. No *finally*, a sessão é fechada, garantindo que a conexão com a Base de Dados seja libertada e que os recursos sejam libertados.

Benefícios da Gestão Transacional:

A gestão transacional utilizando o *handler* "*transactionHandler*" oferece vários benefícios, incluindo: **Consistência dos dados:** A transação garante que os dados sejam mantidos num estado consistente e que as operações sejam executadas de forma consistente. **Integridade dos dados:** Como é mantida a consistência dos dados, é proporcionada uma integridade superior dos mesmo. **Recuperação de erros:** O *try-catch* permite lidar com erros que possam ocorrer durante a execução da transação, garantindo que os dados sejam mantidos num estado consistente e de integridade elevada.

Em resumo, a gestão transacional é um processo crítico na aplicação *WorkoutPal*, e o *handler* "*transactionHandler*" é utilizado para garantir a consistência e integridade dos dados, bem como a recuperação de erros.

5.4 Tratamento de Erros

O tratamento de erros é um processo crítico , pois envolve a manipulação de dados confidenciais e a necessidade de garantir a consistência e integridade dos dados. Para isso, utilizamos o *handler* "*apiErrorHandler*" que é responsável por tratar dos erros que podem ocorrer durante a execução da aplicação.

Handler "apiErrorHandler":

O *handler* "*apiErrorHandler*" é composto por um *try-catch* que permite lidar com erros que possam ocorrer durante a execução da aplicação. No bloco *try*, a ação pretendida é executada e, se for bem-sucedida, o processo continua normalmente. No bloco *catch*, qualquer exceção que ocorra é apanhada e tratada.

Mapeamento de Erros:

No bloco *catch*, a exceção é mapeada para um erro HTTP, o que permite que a aplicação retorne uma resposta de erro HTTP adequada. Isto permite que o cliente seja informado sobre o erro que ocorreu.

Resposta de Erro HTTP:

A resposta de erro HTTP é produzida utilizando a biblioteca *Express.js*, que fornece uma interface para trabalhar com respostas HTTP. A resposta de erro HTTP é composta por um código de status HTTP e uma mensagem de erro.

Em resumo, o tratamento de erros é um processo crítico, e o *handler* "*apiErrorHandler*" é utilizado para garantir a consistência e integridade dos dados, bem como a recuperação de erros.

5.5 Testes

Para garantir a qualidade e a confiança do servidor, foram realizadas várias baterias de testes, baseadas na técnica de testes *White-Box*. Esta técnica permite simular os resultados de certas peças de código, de forma a que apenas se tenha de testar o *output* da peça pretendida, sem a preocupação de usar *inputs* reais.

Testes dos Serviços:

Nos testes dos serviços, as funções da *Data* utilizadas têm os *outputs* simulados para que apenas se avalie a lógica existente nos mesmo. Isto permite que os testes se concentrem na lógica dos serviços, sem a necessidade de lidar com *inputs* reais.

Testes da *API*:

Nos testes da *API*, acontece o mesmo, só que são simulados os *outputs* dos serviços.

6 Implementação da Aplicação Móvel

6.1 Arquitetura

A arquitetura da aplicação móvel pode ser condensada em quatro principais pastas: *app*, *assets*, *domain* e *services*.

A pasta *app* reflete a estrutura a seguir em projetos que utilizam *React Native Expo*, ou seja, cada ecrã está associado a um ficheiro *index*, presente numa pasta cujo nome corresponde ao ecrã em questão. Presente na raiz da pasta *app* estão a pasta *(tabs)* e *auth*. O termo *tabs* é um nome reservado no *React Native Expo*, utilizado para implementar o *layout* de abas. Este *layout* é utilizado para dividir a aplicação em dois componentes, o de *fitness* e de alimentação, em que cada um tem associado um botão correspondente na parte inferior do ecrã. Desta forma, todos os elementos presentes dentro desta pasta beneficiam da interface com os dois botões. Visto que os elementos de autenticação não o fazem, estes são guardados na pasta correspondente, no mesmo nível que a pasta *tabs*. Para além destes elementos, caso fosse necessário acrescentar algo ao *layout* de algum dos ecrãs, foram utilizados ficheiros com o nome reservado *layout*. ver se é preciso mudar alguma coisa aqui

Na pasta *assets* são armazenados elementos que complementam a aplicação. É aqui que são definidos os componentes comuns da aplicação móvel, incluindo as fontes utilizadas, funções e componentes utilizados, tal como as imagens e estilos aplicados ao longo da aplicação.

A pasta *services* é o ponto de contacto com o exterior da aplicação móvel, onde é estabelecido contacto com o servidor. ver se é preciso acrescentar coisas aqui (como a parte da autenticação com a google)

6.2 Contexto da autenticação

De forma a que seja dada maior liberdade ao utilizador, as operações de autenticação não são mandatórias para o uso da aplicação.

Estas operações são necessárias quando é preciso associar a um utilizador dados que necessitam de ser registados, como é o caso da adição de um alimento ao seu registo diário, à criação de planos de treino, tal como á apresentação de estatísticas do utilizador. Se o utilizador tentar executar qualquer uma destas operações sem estar devidamente autenticado, será automaticamente redirecionado para a página de *login*. Excluindo essas situações, todas as outras funcionalidades da aplicação estão disponíveis sem a necessidade de autenticação e podem ser acedidas livremente.

6.3 Interação com o servidor

A interação entre a aplicação *WorkoutPal* e o servidor é feita através de pedidos *HTTP*. Os recursos disponibilizados pelo mesmo podem ser públicos, o que significa que a aplicação móvel tem de apenas fazer o pedido ao servidor com as informações requeridas pelo pedido e recebe o recurso em resposta, ou privados, que para além da informação passada no pedido também requer um *token* de autenticação/autorização válido para que o recurso lhe seja disponibilizado.

Estes pedidos podem acontecer de 2 formas. Uma que acontece quando o utilizador clica num botão existente no ecrã, como por exemplo, clique no botão de

Log In, e outra forma em que é realizado um pedido ao servidor de forma indireta após clique num botão, como acontece no caso em que o utilizador faz *Log In* com uma conta *Google*, porém após uma resposta válida por parte da *API* disponibilizada pela *Google*, é realizado indiretamente um pedido ao servidor realizando esta mesma ação.

6.4 Tratamento de Erros

6.5 Testes

7 *Deploy*

7.1 Servidor

7.2 Aplicação Móvel

A *WorkoutPal* é uma aplicação móvel desenvolvida para apoiar os utilizadores na sua jornada para um estilo de vida mais saudável e equilibrado. Embora não tenhamos recursos financeiros para suportar os custos de *deploy* para as lojas oficiais da *Google* e *Apple*, estamos comprometidos em fornecer uma opção para que os utilizadores possam testar a aplicação.

Para isso, temos disponibilizado o *APK* da aplicação para *Android* no nosso *GitHub*, através do sistema de *release*. Isso permite que os utilizadores baixem e instalem a aplicação diretamente do nosso repositório, sem precisar esperar pela aprovação nas lojas oficiais. Para *iOS* temos no nosso repositório as instruções de como fazer *build* e correr a aplicação para que seja possível testar a aplicação nestes dispositivos.

No futuro, o nosso objetivo é encontrar fundos suficientes para poder disponibilizar a aplicação nas lojas oficiais da *Google* e *Apple*, tornando-a mais acessível para todos os utilizadores.

8 Conclusão

8.1 Dificuldades

8.2 Trabalho Futuro

8.3 Resultados Alcançados