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 o seu progresso. 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 qualidade da informação vinda das *APIs* escolhidas e o controlo de versões de certas dependências utilizadas. Foram abordados conceitos 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 progress. 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 client and server. During the development of the project, we faced several challenges, including the quality of the data provided by the chosen *APIs* and the version control of some used dependencies. Concepts such as the implementation of algorithms for data processing, user authentication, and security techniques were convered. The project covers the use of technologies such as Express.js for the backend, MongoDB 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

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 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.

1.1 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 a criação, como o nome indica, de um parceiro de treino na forma de uma aplicação de telemóvel.

A componente de exercício físico 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 tomadas de decisão 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, entre elas o *signup*, *login* e *logout*.

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.

1.2 Estrutura do Documento

O presente documento está organizado da seguinte forma.

Na Secção 2 apresenta-se a análise do problema, explorando conceitos essenciais e enfatizando as funcionalidades do sistema. Dessa forma oferece-se uma visão mais

clara do objetivo do projeto.

Na Secção 3 descreve-se a solução implementada através da abordagem de tópicos essenciais, como a arquitetura do sistema e as várias tecnologias utilizadas na implementação da aplicação servidora e cliente.

Na Secção 4 explica-se de uma forma geral como cada funcionalidade é implementada tanto do lado do servidor como da aplicação móvel.

Na Secção 5 descreve-se a forma como o servidor foi implementado, abordando tópicos como a arquitetura usada, a lógica utilizada para a comunicação com a base de dados, o modelo de dados que representa o domínio do problema, a forma como os erros são tratados e a técnica usada para testar os vários componentes que compõem o servidor.

Na Secção 6 aborda-se a implementação da aplicação móvel tendo em especial atenção os ecrãs que constituem a aplicação, a organização, o contexto da autenticação e a interação da mesma com o servidor.

Por último, na Secção 7 descrevem-se os desafios encontrados ao longo do desenvolvimento da aplicação, tal como as possíveis melhorias a serem feitas.

2 Análise do Problema

Neste capítulo, apresentam-se os conceitos essenciais para a compreensão da solução desenvolvida, bem como uma descrição dos elementos que a constituem. São 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.

2.1 Conceitos Fundamentais

Os conceitos fundamentais da aplicação *WorkoutPal* envolvem a personalização de planos de treino, a monitorização nutricional e a consulta do progresso ao longo do tempo.

Personalização de Planos de Treino: A *WorkoutPal* oferece aos utilizadores a capacidade de criar, remover e personalizar planos de treino que se ajustam às suas necessidades e objetivos específicos. Esta personalização é importante 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 mesmo.

Monitorização Nutricional: A inclusão de funcionalidades de monitorização nutricional permitem aos utilizadores o registo e consulta dos alimentos consumidos no dia através de um sistema de leitura de código de barras e da pesquisa por um alimento através do nome, bem como o acompanhamento do consumo diário de calorias e nutrientes.

Progresso: A aplicação disponibiliza um conjunto de estatísticas relativas a peso corporal, *fitness* e nutrição para que os utilizadores tenham a possibilidade de analisar o seu progresso durante um período específico de tempo.

2.2 Funcionalidades

No que toca à autenticação o utilizador pode:

- **Criar conta**. Este processo ocorre na fase inicial do uso da aplicação. O utilizador necessita 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 na fase inicial tal como ocorre na criação de conta, onde necessita de preencher os campos com os seus dados e clicar no botão *Log In*.
- **Fechar sessão**. Para isso o utilizador precisará de 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.

Em relação ás funcionalidades de exercício físico, o utilizador pode:

- **Pesquisar exercícios**. Deve navegar até ao ecrã 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.

- **Criar planos de treino**. Para isso, o mesmo deve navegar para o ecrã dos planos de treino e carregar no botão para criar planos de treino("+" no canto inferior direito), que redirecionará o utilizador para o *modal* [13] onde serão pedidos o nome e descrição, opcional, do plano de treino. Após fornecer esses dados, pressiona-se no botão de adicionar.

- **Remover planos de treino**. Para isso, o utilizador deve navegar para o ecrã dos planos de treino e carregar no botão *Delete* que aparece de baixo de cada plano de treino. Após pressionar este botão aparece um *pop-up* de confirmação, em que o utilizador deve carregar em *Yes* se quiser apagar, ou em *No* caso contrário.

- Para **adicionar um exercício ao plano de treino**, o utilizador deve aceder ao ecrã do exercício e pressionar o botão para o adicionar. Após carregar no botão o utilizador tem acesso a um *modal* [13] onde aparecem todos os planos associados à sua conta. Após a escolha de um plano de treino o exercício é adicionado. É possível adicionar o mesmo exercício a vários planos de treino, por isso, quando o utilizador estiver satisfeito, carrega na cruz para voltar fechar o *modal* [13].

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

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

Em relação à vertente da nutrição o utilizador pode:

- **Consultar o seu 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. Neste ecrã ao pressionar o registo de um alimento, o utilizador é redirecionado para o ecrã desse alimento onde poderá alterar a quantidade consumida caso assim o pretenda.

- **Pesquisar 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 e, de seguida a ser feita 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.

- **Registar um alimento**. Para registar um alimento o utilizador navega para o ecrã referente ao mesmo através dos métodos de pesquisa. Neste ecrã caso o utilizador deseje alterar a quantidade do alimento e por consequência os nutrientes associados, tem essa opção. Para além de informações relevantes ao alimento em questão, este ecrã 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.

- **Remover o registo de um alimento**. Para remover o registo de um alimento o utilizador necessita de pressionar o registo correspondente no seu registo diário por alguns segundos. Aparecerá um *modal* [13] com a opção de remoção deste registo, onde o utilizador é dado a escolher se pretende confirmar esta remoção ou não.

No que diz respeito ao registo do progresso, o utilizador pode:

- **Consultar o seu progresso**. Para que o utilizador consiga aceder ao seu progresso, o mesmo terá de clicar no botão *progress*, presente no menu. Dessa forma é redirecionado para o ecrã de progresso, onde encontrará um conjunto de gráficos contentores de informação sobre o seu progresso durante um certo período de tempo no que toca aos membros mais importantes da aplicação (peso, nutrição e *fitness*).

- **Atualizar o peso**. O utilizador ao pressionar no botão *progress*, presente no menu, navega para o ecrã relativo ao progresso onde encontra uma vista de atualização de peso. Nesta vista após o utilizador inserir o novo peso e carregar no botão *Update* o sistema atualiza o peso correspondente ao dia corrente.

3 Descrição da Solução

Neste tópico aborda-se 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 *APIs* 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 *APIs* escolhidas foram a *OpenFoodFacts API* [3] para alimentação/nutrição, uma vez que é das *APIs* mais completas da área, e a *ExerciseDB API* [1] 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* [1], foi tomada a decisão de fazer a clonagem diária de todos os exercícios para uma base de dados relacionada ao projeto devido à 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.

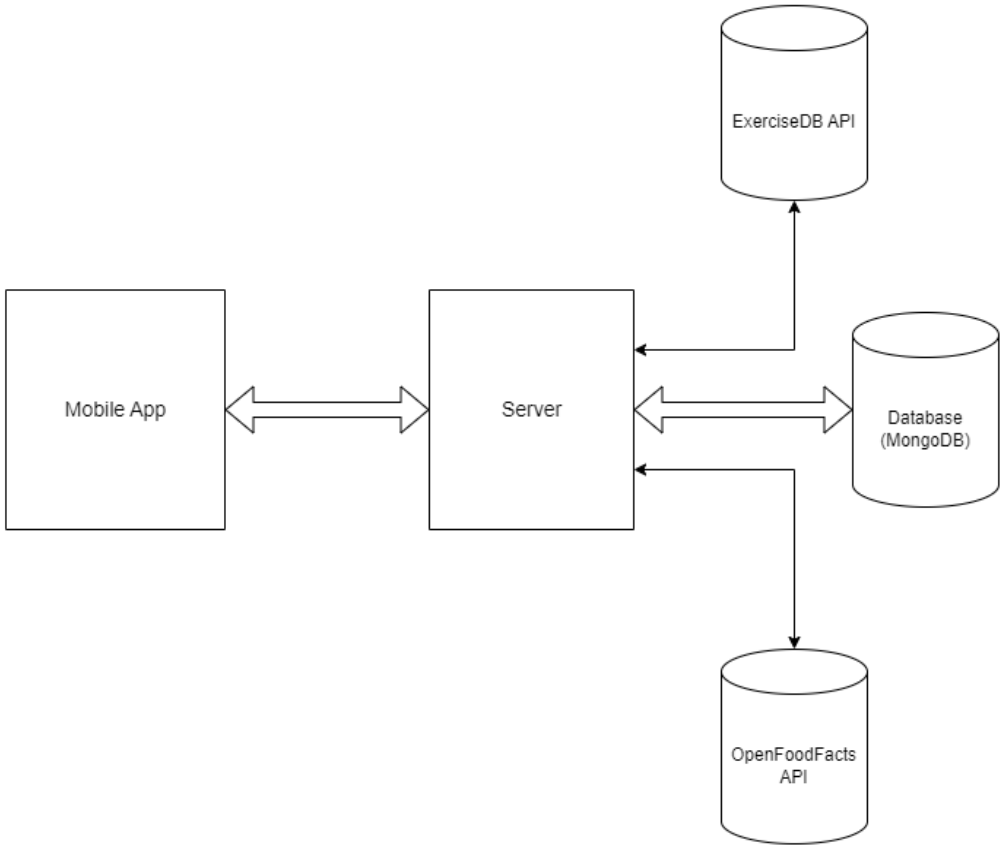

Figura 1: Arquitetura do Sistema

3.2 Tecnologias Utilizadas

Nesta subsecção abordam-se as várias tecnologias utilizadas para a implementação do servidor, tal como as que ajudaram a implementar a aplicação móvel. Também serão enunciadas as várias razões que levaram à decisão do uso das mesmas.

3.2.1 Servidor

Decidiu-se usar *NodeJS* [6] devido principalmente à intenção que houve de aprofundar competências relativas a este ambiente de execução e de *Typescript*. O código foi desenvolvido em *Typescript* em vez de *Javascript* com o intuito de precaver possíveis erros futuros, tal como de forma a clarificar a constituição dos dados utilizados na aplicação. Escolheu-se a framework *expressJS* para auxiliar a construção da *Api* e *MongoDB* [8] como base de dados. A escolha de *MongoDB* [8] advém desta ser uma base de dados com poucos custos de manutenção, tal como da facilidade de acesso aos dados que proporciona através da *framework Mongoose* [9].

3.2.2 Aplicação Móvel

A procura por um *stack* tecnológico que permitisse realizar uma aplicação móvel multi-plataforma levou a duas principais opções: *Kotlin Multiplataform* e *React Native*.

Devido à utilização de *Typescript* no *backend* e à intenção de explorar a *framework React* de forma mais aprofundada, foi decidido o uso da framework *React Native*.

Para a construção de uma aplicação *React Native* foram encontradas duas soluções: *React Native CLI* ou *Expo*. Foi escolhida a última devido à maior facilidade em realizar o *setup* da mesma em relação a *React Native CLI*, tal como pela disponibilização de componentes *template* a usar como ponto de partida que esta proporciona. O *Expo*, além de facilitar o controlo de versões, oferece também uma maneira centralizada de realizar o *deploy* da aplicação para ambientes *Ios* e *Android*.

4 Implementação do Servidor

4.1 Arquitetura

A implementação do servidor é baseada numa estrutura *API*->Serviços->*Data* (Figura 2), que permite uma separação clara entre as responsabilidades de cada camada.

***API*:**

A *API* é responsável por lidar com o pedido 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 pedido:** A *API* recebe o pedido 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 responsabilizam-se por implementar a lógica da aplicação consoante o input recebido e o output resultante da chamada ao *Data*. Estes são a segunda camada de abstração entre a *API* e a *Data* e são responsáveis por:

- **Avaliação do input:** Nos Serviços são realizadas avaliações ao input recebido e são tomadas decisões como que operação correr ou que exceção lançar.
- **Chamar o módulo *Data*:** Os Serviços chamam o módulo *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*:**

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

- **Comunicar com a base de dados:** Este módulo trata da comunicação com a base de dados, incluindo a criação, leitura, atualização e exclusão de dados.
- **Comunicar com a *API OpenFoodFacts* e *ExerciseDB*:** É neste módulo onde ocorre também a comunicação com as *APIs* escolhidas para o projeto, incluindo a pesquisa de informações sobre alimentos e a obtenção de dados nutricionais relativamente à *OpenFoodFacts* e a clonagem dos dados disponibilizados pela *Api ExerciseDB*.

Benefícios da utilização desta arquitetura:

A estrutura *API*->Serviços->*Data* oferece benefícios como a separação de responsabilidades, isto é, cada camada tem uma responsabilidade clara e bem definida, o que facilita a manutenção e o desenvolvimento do servidor. Outro benefício é a flexibilidade que esta solução proporciona relativamente á possibilidade de substituição de qualquer uma das peças(*API*, Serviços e *Data*) sem afetar as restantes.

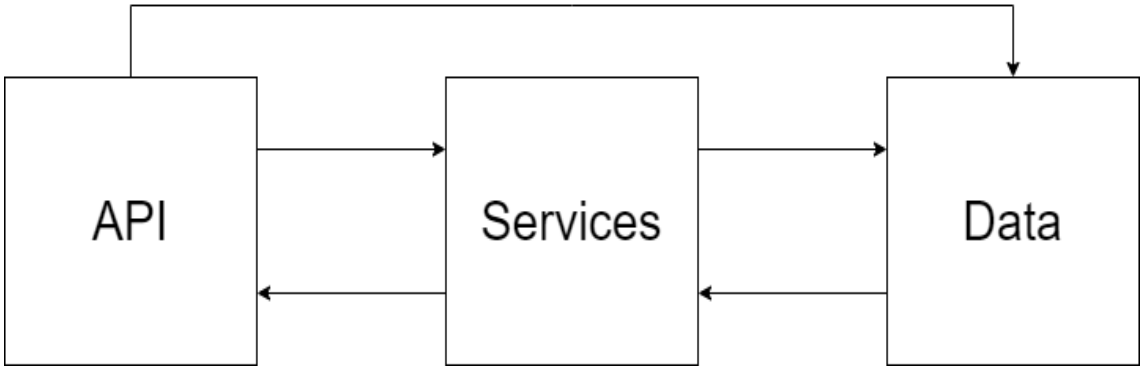

Figura 2: Arquitetura do Servidor

4.2 Comunicação com a base de dados

4.2.1 Conexão

A conexão com a base de dados é efetuada durante a inicialização do servidor, garantindo que a mesma seja estabelecida antes de qualquer operação ser realizada. Além disso, a conexão é fechada automaticamente quando o servidor é desligado.

É utilizada a biblioteca *Mongoose*, que fornece uma interface simples de usar e interagir com uma base de dados *MongoDB*. Para isso, criou-se dois *Schemas*, *exerciseSchema* e *userSchema*, que definem a estrutura dos dados armazenados.

4.2.2 Representação dos Dados

Os dois *Schemas* usados, *exerciseSchema* e *userSchema*, são criados com as mesmas propriedades dos tipos *User* e *Exercise* do domínio da aplicação presentes em anexo neste documento.

4.2.3 Gestão Transacional

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 capturados e o *rollback* da transação é realizado, garantindo a consistência dos dados. No bloco *finally*, a sessão é fechada, garantindo a libertação dos recursos e da conexão com a base de dados.

4.3 Tratamento de Erros

Para facilitar o tratamento de erros, foi usado o *handler apiErrorHandler* que é responsável pelo tratamento de 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 manipular respostas HTTP. A resposta de erro HTTP é composta por um código de status HTTP e uma mensagem de erro.

4.4 Testes

Para garantir a qualidade e a confiança do servidor, foi realizada uma bateria de testes, baseadas na técnica *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 terem de ser usados *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 nas mesmas. 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.

4.5 *Deploy*

Para que a *API* pudesse ser disponibilizada publicamente, optámos por alojar o servidor no serviço *Fly.io* [15]. Escolheu-se este serviço porque o mesmo oferece a ferramenta *flyctl*, que facilita o processo de *deploy*, permitindo realizá-lo com apenas dois comandos: *fly launch* e *fly deploy*.

O comando *fly launch* é responsável por inicializar uma nova aplicação no *Fly.io*, configurando automaticamente os parâmetros necessários e gerando os arquivos de configuração essenciais. Em seguida, o comando *fly deploy* realiza o upload do código da aplicação para o servidor e inicia a aplicação, tornando-a acessível publicamente.

Graças a esta automatização, conseguimos acelerar o processo de disponibilização da API, reduzindo o tempo e os esforços necessários para colocar o servidor em funcionamento. Com a API alojada no *Fly.io*, é possível fazer pedidos *HTTP* ao servidor através do seguinte *URL*: *https://backend-workoutpal.fly.dev*

5 Implementação da Aplicação Móvel

A aplicação é constituída por quatro conjuntos principais de ecrãs, os ecrãs de autenticação, *fitness*, nutrição e progresso.

Na altura do acesso à aplicação por parte do utilizador, o mesmo é redirecionado para o ecrã de *Log In* caso não esteja autenticado ou para o ecrã principal (*Fitness Main Screen*) caso contrário.

No cenário em que o utilizador deseja terminar a sessão, existe a opção de *Log Out* presente no ecrã *Menu* que pode ser acedido a partir de qualquer ecrã com a exceção dos ecrãs relativos à autenticação.

O grafo de navegação da aplicação está presente na figura abaixo.

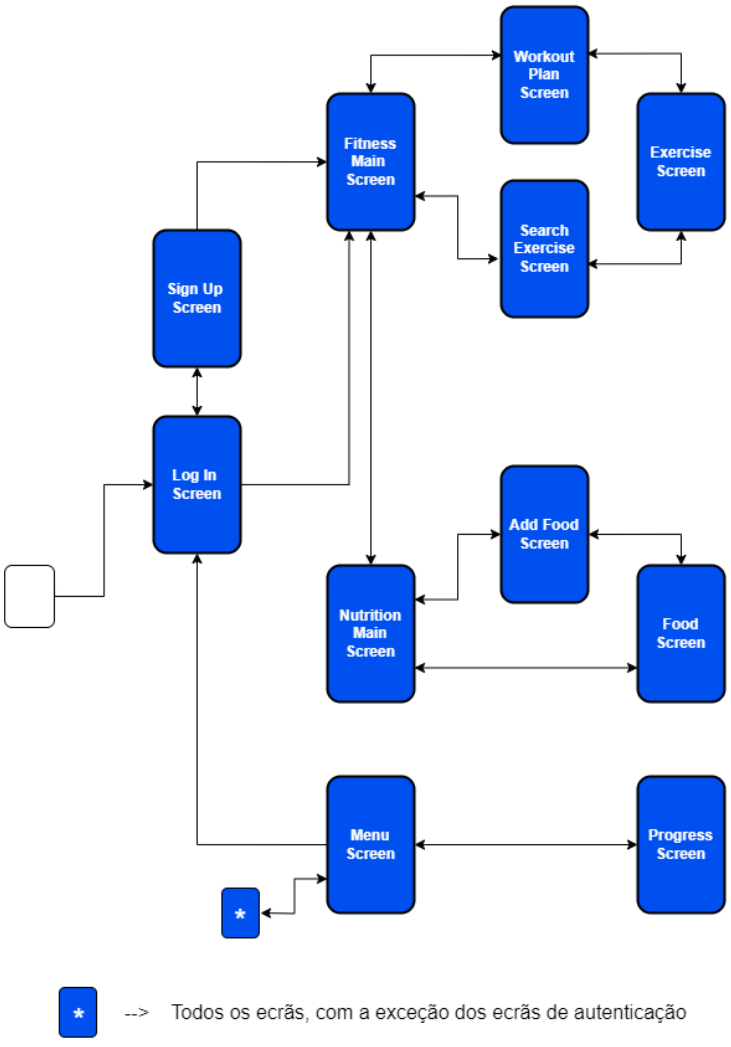

Figura 3: Grafo de navegação

5.1 Organização

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

Para a pasta *app* a decisão foi de divergir do método usual de representar as rotas em *React Native Expo* de forma a ter uma representação mais personalizada para o resultado pretendido. Esta é constituída de forma geral pelas pastas *auth*, *exercises*, *food*, *progress*, *menu* e o ficheiro *layout*.

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 realizada a comunicação com o servidor.

5.2 Contexto da autenticação

A autenticação de um utilizador ocorre logo no início de forma a dar total liberdade para o uso da aplicação após este passo. Caso o utilizador já tenha iniciado sessão previamente, não necessita de o fazer cada vez que acede á aplicação, sendo a sessão armazenada para este efeito.

5.3 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 apenas de 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.

5.4 *Deploy*

Com o intuito de disponibilizar a aplicação publicamente, optou-se por criar um *APK (Android Package)* para possível *download* a partir do repositório do projeto (`https://github.com/fontes71/WorkoutPal/releases/tag/Build-1.0.0`).

Para o efeito usufruiu-se do serviço *EAS (Expo Application Services)* [16] disponibilizado pela *Expo*. O mesmo proporciona ferramentas para fazer *build* do projeto tanto para *ios* como *android*, porém para realizar o *build* para *ios* é necessário um dispositivo com sistema operativo *macOS*. Por essa razão, foi apenas realizado o *deploy* para *android*.

6 Implementação das Funcionalidades

Nesta secção descreve-se a forma como cada operação disponível foi implementada tanto do lado do servidor como da aplicação móvel.

6.1 Autenticação

6.1.1 *Signup*

A aplicação cliente ao usar a operação de *signup*, disponibilizada pelo servidor, faz um pedido que inclui 3 informações importantes (nome, email, *password*) das quais são feitas verificações como por exemplo a 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 à 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.

6.1.2 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 operação é invocada após o utilizador inserir os dados pedidos e carregar no botão de *Log In*.

6.1.3 *Logout*

No menu de opções da *WorkoutPal* é disponibilizada a opção de *logout*. Aquando do utilizador pressionar 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.

6.2 *Fitness*

6.2.1 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 pesquisa 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 o nome do exercício, a parte do corpo, o equipamento necessário e o músculo alvo. Para obter mais informações detalhadas sobre cada exercício, como as 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 exercício, onde se encontram informações adicionais para ajudar os utilizadores a entender melhor como realizar o exercício de maneira correta.

6.2.2 Planos de treino

Na aplicação são também fornecidas aos utilizadores ferramentas para criar, remover, e personalizar os seus próprios planos de treino. O ecrã principal de *Fitness*, que realiza um pedido ao servidor para obter a lista de planos associados à sua conta, apresenta o nome e descrição de cada plano, com um botão *"Delete"* para poder apagar os mesmos. 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, é efetuado um pedido ao servidor para obter as informações dos exercícios incluídos no plano. 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 principal de *Fitness*, que redireciona o utilizador para um *modal* [13]. 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 remoção de um plano de treino é feita através de um botão presente de baixo de cada plano de treino, onde é apresentado um *pop-up* de confirmação. Ao fazer o pedido, o servidor verifica se existe um plano de treino com o mesmo nome associado à conta. Em caso de existir, o mesmo é removido. Em caso de erro, é retornado um erro indicando que o plano não 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* [13]. 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 de carregar no plano ao qual deseja adicionar o exercício. 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 na 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 cruz de retorno para fechar o *modal* [13].

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.

6.2.3 Registo de um plano de treino

Ao aceder à página do plano de treino, os utilizadores encontram um botão no canto inferior 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*.

6.3 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.

6.3.1 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 de pesquisa 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. Caso não encontre o resultado que procura na primeira pesquisa, ao mover o ecrã para baixo novos resultados são disponibilizados de encontro com a pesquisa realizada. O servidor envia um pedido à *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 à 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.

6.3.2 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. O utilizador ao pressionar onde está exposta a quantidade pode alterá-la, alterando também os nutrientes de forma proporcional. 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 à 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.

6.3.3 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 à base de dados de forma a encontrar o utilizador em questão. De seguida é feita uma procura pelos registos do dia atual do utilizador e este é retornado ao cliente de forma a ser apresentado. Ao pressionar num dos registos, o utilizador é redirecionado para o ecrã do alimento desse mesmo registo, podendo alterar a quantidade consumida.

6.4 Progresso

Com o intuito de representar o progresso, o utilizador tem acesso a estatísticas relacionadas a cada membro principal disponibilizada pela aplicação (Peso, *Fitness* e Nutrição). A informação disponibilizada pelo servidor inclui um conjunto de todos os dias durante um período especificado, podendo esse ser semanal, mensal ou anual. A mesma é disponibilizada na aplicação *WorkoutPal* através de gráficos mensais.

6.4.1 Atualização do peso

Para que o utilizador tenha acesso ao seu progresso em termos de peso corporal o mesmo pode atualizar o seu peso diariamente acedendo ao ecrã de progresso presente no menu. Abaixo do gráfico contentor da informação do peso está presente

a possibilidade de atualizar o mesmo. Para que isso seja possível o utilizador necessitará inserir o peso corrente e clicar no botão **Update**. Aquando do clique do botão é realizado um pedido ao servidor, que contem o novo peso, o dia corrente e o *token* de autorização do utilizador autenticado, informação esta que é utilizada para atualizar o peso do dia especificado para o novo peso inserido.

6.4.2 Estatística

O utilizador tem acesso a estatística acedendo ao ecrã de progresso presente no menu. Aquando da navegação para o ecrã em causa é realizado um pedido automático ao servidor que recebe como parâmetros o *token* de autorização do utilizador autenticado, a data corrente e o período. Este pedido retorna uma lista de todos os dias em que o utilizador atualizou alguma informação, atualização do peso, realização de um plano de treino, etc.

- O gráfico relativo ao peso corporal representa o peso diário consoante as atualizações que o utilizador realizou durante o período em causa
- O gráfico representativo da componente de *fitness* contem a informação sobre a quantidade de planos de treino que foram realizados diariamente durante o período especificado
- Relativamente a componente nutricional o utilizador tem acesso a um gráfico contentor das calorias diárias consumidas pelo mesmo consoante os alimentos registados durante o período de tempo em causa.

7 Conclusão

7.1 Desafios

Os maiores desafios encontrados ao longo do desenvolvimento do projeto foram a aprendizagem de novas tecnologias usadas para o desenvolvimento do trabalho, assim como questões como a qualidade dos dados provenientes de *APIs* e o controlo de versões.

De forma a aprender novas formas de desenvolvimento de software foi decidido utilizar *React Native Expo* para a construção deste projeto, visto que, apesar de utilizar *React*, tecnologia com a qual já houve contacto prévio, como o foco é o desenvolvimento para clientes móveis, provou-se inicialmente desafiante alterar noções prévias de como realizar código em *React*.

Outro desafio encontrado foi a qualidade de informação da *API* de alimentos *Open Food Facts* que, apesar de muito extensa, revelou-se ter uma qualidade de informação inferior à esperada.

Para a execução da aplicação de forma atualizada foi utilizada a aplicação *Expo Go*, aplicação oficial para *React Native Expo*. No decorrer do uso da mesma, ao longo do desenvolvimento da aplicação, passou a ser mandatória a versão mais recente de *React Native Expo*. Por ser consideravelmente recente, esta mudança levantou certos problemas provenientes não só desta mesma atualização, mas também do controlo de versões de outros *packages* utilizados.

7.2 Melhorias

No início do desenvolvimento do projeto foi tomada a decisão de não guardar os dados provenientes da *API Open Food Facts* numa base de dados própria tal como é feito com os dados da *API Exercise DB*, visto que como a *API* é muito extensa, fazê-lo na sua totalidade teria custos associados, enquanto que parcialmente levaria a um número muito reduzido de resultados. Contudo, considera-se que suportar o armazenamento destes dados na sua totalidade é crucial para a otimização desta solução visto que a *API* referida impõe limites de sobrecarga de pedidos, que num cenário de elevada utilização revelar-se-ia problemático.

Uma outra possível otimização seria o modelo de base de dados utilizado. De forma a otimizar o espaço utilizado, devido principalmente à inevitável repetição de alimentos registados por um ou vários utilizadores, considera-se que uma mudança para uma base de dados de modelo relacional revelou-se mais adequada.

8 Referências

Referências consultadas a 01-05-2024

1. *ExerciseDB API* (https://exercisedb.io)
2. *ExerciseDB Endpoints* (https://rapidapi.com/justin-WFnsXH_t6/api/exercisedb)
3. *OpenFoodFacts API* (https://world.openfoodfacts.org)
4. *OpenFoodFacts API Documentation* (https://world.openfoodfacts.org/data)
5. *Typescript Documentation* (https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html)
6. *NodeJs Documentation* (https://nodejs.org/docs/latest/api/)
7. *ExpressJs Documentation* (https://expressjs.com/en/5x/api.html)
8. *MongoDB Documentation* (https://www.mongodb.com/docs/)
9. *Mongoose Documentation* (https://mongoosejs.com/docs/)
10. *Mongoose Typescript Support* (https://mongoosejs.com/docs/typescript.html)
11. *React Native Documentation* (https://reactnative.dev/docs/getting-started)
12. *Expo Documentation* (https://docs.expo.dev)
13. *Expo Modals Documentation* (https://docs.expo.dev/tutorial/create-a-modal/)
14. *React Native Chart Kit* (https://www.npmjs.com/package/react-native-chart-kit)
15. Fly.io Documentation (https://fly.io/docs/getting-started/launch/)
16. EAS Docs (https://expo.dev/eas)

9 Anexos

Modelo de Dados

***User*:**

- 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.
- workoutPlans *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 "yyyy-MM-dd".
- loggedFood: *Array* de objetos *Food*
- workoutPlansDone *Array* de *id's* dos planos de treino realizados no dia.
- weight: Peso do utilizador

***Food*:**

- id: Identificador único do alimento.
- name: Nome do alimento.
- brand: Marca do alimento
- quantity: Quantidade do alimento e sistema metrico associado
- imageUrl: Imagem representativa do alimento
- mainNutrients: Objeto do tipo *MainNutrients*
- secondaryNutrients: Objeto do tipo *SecondaryNutrients*

***MainNutrients*:**

- calories: Calorias do alimento.
- protein: Proteína do alimento.
- fat: Gordura do alimento.
- carbs: Carboidratos do alimento.

***SecondaryNutrients*:**

- fiber: Fibra do alimento.
- saturatedFat: Gordura saturada do alimento.
- salt: Sal do alimento.
- sodium: Sódio do alimento.
- sugars: Açúcar 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.