Universidade do Minho
Escola de Engenharia
Mestrado Integrado em Engenharia Informática

Unidade Curricular de Laboratórios de Informática IV

Ano Letivo de 2020/2021

Code&GO

Mário Coelho (42865) Rita Teixeira (89494) Rui Armada (90468) Vasco Moreno (94194)

3 de março de 2022

LI4

Data de Receção	
Responsável	
Avalição	
Observações	

Code&GO

Mário Coelho (42865) Rita Teixeira (89494) Rui Armada (90468) Vasco Moreno (94194)

3 de março de 2022

Dedicado aos profissionais das tecnologias de informação

(formados ou em formação)

Resumo

O presente documento resume os principais trabalhos realizados na fase de concepção e preparação da codificação de uma aplicação informática. A aplicação em causa diz respeito a um guia de locais de interesse dedicado ao público a quem se dedica este documento – os profissionais das tecnologias de informação (formados ou em formação).

A aplicação foi pensada para funcionar à semelhança de um jogo estilo caça ao tesouro. No entanto, e tendo por base as preferências do público alvo, o "tesouro"serão pequenos desafios de programação. Assim, o objectivo é convidar os utilizadores da aplicação a conhecerem o mundo real que os rodeia, através da procura por locais com algum tipo de interesse patrimonial, cultural ou histórico, onde estarão escondidos os referidos problemas de programação.

Mediante a resolução dos problemas que forem sendo encontrados, cada utilizador poderá ir descobrindo novos problemas em novos locais de interesse, construindo assim uma reputação de programador bom e bem viajado.

É possível identificar no primeiro capítulo as principais motivações que levaram à conceção da aplicação. Segue-se a descrição dos requisitos da aplicação (segundo capítulo), bem como a especificação da mesma (terceiro capítulo). Os últimos dois capítulos do documento dizem respeito aos detalhes do sistema de dados que dará suporte à aplicação (quarto capítulo) e à idealização da interface que a aplicação deverá fornecer ao utilizador (quinto capítulo).

Área de Aplicação: Desenvolvimento de *Software*, Engenharia de *Software*, Jogos, Saúde e Fitness, Educativo, Cultura Geral.

Palavras-Chave: *Frontend*, *Backend*, *Gantt*, Base de Dados, *TIC*, *API*, textitProgramming, *Software Engeneering*.

Índice

Lista de Figuras

Lista de Tabelas

1 Introdução

Nas secções que se seguem apresenta-se o contexto em que este trabalho se insere, o problema ao qual pretende dar resposta, bem como a proposta de solução nesse sentido. Para além dos recursos e da calendarização espectável para o desenvolvimento da solução proposta, apresentam-se também as principais métricas que permitirão aferir o sucesso da mesma.

1.1 Contextualização

Na sequência da recente pandemia de COVID-19 que o mundo enfrentou, foram vários os problemas que, de alguma forma, foram secundarizados por força do impacto da mesma. Em simultâneo, muitos outros problemas tornaram-se mais evidentes e acabaram por beneficiar de uma maior atenção. Um bom exemplo desta segunda situação é o caso do sedentarismo. Com efeito, é sabido há vários anos que, sobretudo nos países mais desenvolvidos, o estilo de vida mais sedentário vinha a tornar-se um problema de saúde pública. A nível mundial, estima-se que o sedentarismo e a falta de exercício físico estejam na origem de cerca de cinco milhões de mortes por ano [1]. No caso particular de Portugal, segundo o Eurobarómetro de 2018, 74% da população raramente ou nunca realizava exercício físico, colocando assim o país no quinto lugar dos países mais sedentários da União Europeia [2]. Dois fatores que contribuem para estas percentagens são o tempo gasto a ver televisão, 60% dos rapazes e 56% das raparigas vêem diariamente pelo menos duas horas, e o tempo gasto em videojogos, sendo neste caso as percentagens 51% (rapazes) e 33% (raparigas) [1].

O foco do presente trabalho reside num subgrupo bem definido da população mais jovem – os estudantes de Engenharia Informática e cursos da área das Tecnologias da Informação e Comunicação (*TIC*) no geral. Analisando este grupo, verifica-se que a percentagem de rapazes é superior à de raparigas. Por outro lado, numa amostragem empírica do universo com quem o grupo contacta, verifica-se que as percentagens de tempo gasto por este público, entre televisão e videojogos, deverão ser superiores aos valores anteriormente apresentados para a população como um todo. Enquanto membros integrantes deste público, o grupo decidiu aproveitar o presente trabalho prático e desenvolver uma possível solução para o problema do sedentarismo entre os estudantes das *TIC*, tirando partido do conhecimento das coisas que motivam esta audiência específica.

Nos últimos anos são vários os estudos que apontam para a existência de uma relação direta entre a gamificação e a motivação para a atividade física [3][4]. Em particular, no

trabalho de Paulo Ricardo da Silva [5] é possível verificar que foi desenvolvida uma aplicação *Android* que, recorrendo a mapas e à realidade aumentada, fornece um jogo que estimula a realização de atividade física. Também Rui Manuel Neto Policarpo [6] recorreu à mesma estratégia, neste caso por forma a estimular o contacto com a natureza. Para isso, desenvolveu uma aplicação para *iOS* que, alimentada com percursos pedestres homologados pela Federação de Campismo e Montanhismo de Portugal, permite registar os percursos já realizados pelos utilizadores, fazendo com que a motivação de fazer mais percursos tivesse como efeito secundário a realização de mais atividades na natureza.

No contexto dos estudantes das *TIC*, também existem alguns casos de gamificação, embora não relacionados com a atividade física, mas sim com a atividade profissional. Um excelente exemplo disso é a plataforma *Codewars* [7]. Trata-se de uma plataforma de avaliação de codificação que tem por objetivo acelerar a melhoria técnica, a qualificação e a certificação dos seus utilizadores. No entanto, a forma como tudo isto se desenvolve está inserida num ambiente de jogo. Esse jogo consiste na resolução de pequenos exercícios de codificação, aqui designados por *katas*. Cada kata é criado pela comunidade, nas mais diversas linguagens de programação de modo a fortalecer diferentes habilidades de codificação. É possível que o utilizador se fixe numa só linguagem, ou que utilize mais do que uma, melhorando assim a sua performance em várias linguagens de programação. A ideia é resolver os *katas* diretamente no *browser* e, através da aplicação de casos de teste automáticos, ir avançando os vários níveis. Os *katas* são classificados desde os desafios de código para iniciantes até ao nível de especialista. Conforme se completem *katas* de classificação mais alta, o utilizador vai progredindo na classificação geral, sendo este o elemento motivacional do jogo.

1.2 Motivação e Objetivos

Na secção anterior foi identificado o problema do sedentarismo e da falta de exercício físico nos estudantes das *TIC*. Verificado que a gamificação se tem revelado uma boa solução na mitigação deste problema e que os estudantes de Informática são particularmente sensíveis aos videojogos, o presente trabalho pretende apresentar uma aplicação que motive os jovens entusiastas da Programação e *Software* a praticarem mais atividade física. Os autores do presente relatório decidiram, assim, tentar resolver este problema através de jogos para dispositivos móveis que deverão levar os seus utilizadores a passar mais tempo ao ar livre e a ter uma maior quantidade de interações sociais.

Deste modo, os principais objetivos do presente trabalho são:

- Redução do sedentarismo entre os jovens;
- Aumento da atividade física por parte dos jovens;
- Melhoria das competências técnicas de programação;
- Aumento do conhecimento do património dos locais onde os jovens circulem.

- Promover e dar conhecimentos a terceiros sobre o mundo da programação.

1.3 Justificação e utilidade do sistema

A aplicação a desenvolver no presente trabalho tira partido do elevado interesse que o público alvo tem por videojogos. Assim, se não fosse por mais nada, estava desde logo justificada a utilidade da aplicação. De facto, a existência de um público interessado num determinado produto, é em si mesmo um motivo válido para o seu desenvolvimento.

Por outro lado, e tendo em consideração o grande objetivo de reduzir o sedentarismo entre os jovens estudantes das TIC, acredita-se que uma aplicação que os leve a acreditar que se encontram num jogo, que têm uma reputação (virtual) a preservar e que esta depende da sua atividade (física), permitirá atingir com sucesso tal objetivo.

A aplicação tem uma outra grande utilidade relacionada com a melhoria das competências de programação dos seus utilizadores. Espera-se que a presente aplicação se possa incluir na categoria dos jogos inteligentes, i.e. jogos que estimulam o desenvolvimento da inteligência. Neste caso em particular, a resolução dos diferentes desafios de programação, condição necessária para aceder a novos níveis, permite atingir este objetivo.

Por outro lado, a já referida "reputação virtual"e a competitividade inerente, também se enquadram neste objetivo de melhoria das competências de programação. Basta pensar que para um jogador que deseje estar classificado, por exemplo, nos 100 primeiros postos, este vai ter que resolver muitos *katas*, sendo estes de dificuldades diferentes, sobre temas variados e até mesmo em diversas linguagens de programação. Posto isto, espera-se que haja um aumento significativo nas *skills* de programação dos utilizadores.

Foi também identificada, na fase de conceção da ideia para a aplicação, que o público alvo selecionado não tem por hábito visitar e conhecer o património, mesmo aquele que existe no local onde residem. Assim, através da colocação dos desafios de programação em locais com interesse patrimonial, consegue-se também fomentar nos jovens o conhecimento do património. Não é possível com a aplicação a desenvolver garantir que este conhecimento seja efetivo. É possível apenas garantir que os utilizadores são colocados no local onde o património se encontra. Se depois é despertado o interesse do utilizador para o que o rodeia, e acontece algum tipo de interação que o leva a conhecer o património, é algo a avaliar. Eventualmente isso poderá ser atingido através de novas funcionalidades a adicionar no futuro (e.g. *quiz* sobre o local visitado).

Adicionalmente, é previsível que a aplicação aqui apresentada venha a evoluir de forma gradual no futuro. Esta será desenvolvida de forma modular, de modo a permitir a fácil adição de novas funcionalidades no futuro.

1.4 Estabelecimento da identidade do projeto

A identidade da aplicação a desenvolver encontra-se resumida na Tabela 1.1. O nome escolhido pretende remeter para o principal objetivo do jogo subjacente à aplicação. Assim, a ideia principal é resolver um desafio de programação (*Code*, em Inglês) e partir para um novo local em busca do desafio seguinte (*Go*, em Inglês). A concatenação destes dois termos, substituindo a conjunção "e" (*and*, em Inglês, abreviado aqui através do símbolo "&") resulta então no nome final - Code&Go. Acredita-se que a aplicação tem potencial de internacionalização, pelo que a definição do nome se baseou na língua mais falada no mundo, ou seja, no Inglês.

Tanto pelo facto de a utilização da aplicação pressupor o conhecimento de alguma linguagem de programação como pela necessária autonomia para se deslocar entre diferentes sítios, esta aplicação terá de se colocar na faixa etária dos maiores de idade. Isto não impede que utilizadores menores possam também utilizá-la. No entanto, nesse caso é recomendável a presença de um adulto.

De modo a facilitar a procura da aplicação nas lojas de aplicações móveis, bem como publicitá-la a potenciais utilizadores, esta será associada às categorias de Saúde e Fitness, Educativo e Cultura Geral. Alguém com interesses numa ou mais destas categorias é um potencial utilizador e deverá assim conseguir descobrir a aplicação facilmente.

A aplicação a desenvolver servirá nesta primeira fase para provar o conceito subjacente. Assim, terá uma abrangência limitada. Nesta primeira versão da aplicação pretende-se desenvolver e testar o produto final no contexto do público estudantil do Departamento de Informática da Universidade do Minho. Assim, nesta versão designada "Gualtar Campus", os locais de interesse serão limitados aos limites do Campus de Gualtar da Universidade do Minho. Em futuras versões, e consoante o volume de utilizadores da aplicação vá escalando, é expectável o desenvolvimento de versões sucessivamente mais globais (e.g. Code&GO Braga, Code&GO Portugal, Code&GO Europe).

Nome:	Code&GO
Versão:	Gualtar Campus
Data de lançamento:	24 Janeiro 2022
Proprietário:	LIVG22
País:	Portugal
Categoria:	Saúde e Fitness, Educativo, Cultura Geral
Faixa etária:	18+

Tabela 1.1: Identidade da aplicação Code&GO

1.5 Identificação dos recursos necessários

Os recursos necessários para o desenvolvimento da presente aplicação podem ser categorizados em humanos, físicos, de gestão, de pesquisa, de reporte e, finalmente, de desenvolvimento. Nos parágrafos que seguem detalha-se um pouco cada uma destas categorias, enquanto na Tabela 1.2 se resumem os recursos necessários.

Tipo de recurso	Descrição	Fase
Humanos	4 engenheiros informáticos	Conceção e preparação
Humanos	4 engenheiros informáticos	Codificação
Humanos	5-10 utilizadores	Testes
Físicos	4 Computares	Conceção e preparação
Físicos	4 Computares	Codificação
Físicos	5-10 smartphones	Testes
Gestão	Grupo no Discord	Todas
Pesquisa	Acesso à Internet	Todas
Reporte	Plataforma Overleaf	Todas
Desenvolvimento	Adobe Illustrator	Conceção e preparação
Desenvolvimento	Visual Paradigm	Preparação
Desenvolvimento	Ambiente Integrado de Desenvolvimento (IDE)	Codificação/Testes
Desenvolvimento	Serviço de base de dados	Codificação/Testes
Desenvolvimento	Github	Codificação/Testes
Desenvolvimento	API do CodeWars	Codificação

Tabela 1.2: Recursos necessários para o desenvolvimento da aplicação.

Na categoria dos recursos humanos, incluem-se todas as pessoas direta e indiretamente envolvidas no desenvolvimento da aplicação. Isto inclui desde logo os quatro signatários do presente relatório, que foram os responsáveis pelas tarefas nele descritas. Estas tarefas incluem toda a conceção e preparação da fase de codificação da aplicação. Por sua vez, a fase de codificação vai necessitar uma equipa de outros quatro engenheiros informáticos (os elementos do grupo que vai implementar a aplicação descrita neste relatório). Adicionalmente, é ainda possível identificar como recursos humanos o conjunto de utilizadores que vão testar as versões de pré-lançamento da aplicação. Nesse sentido, deverão ser contactados cinco a dez alunos da Licenciatura em Engenharia Informática para representar tal papel e fornecer avaliações que permitem finalizar a primeira versão final da aplicação.

Quanto aos recursos físicos, encontram-se desde logo os computadores, que os engenheiros identificados anteriormente irão utilizar para realizar o seu trabalho, e os smartphones, que os utilizadores de teste irão utilizar para testar a aplicação. Note-se que os referidos smartphones devem estar equipados de um sistema de GPS que irá fornecer a localização não só do utilizador, mas também os locais a ser visitados pelo mesmo.

Seguidamente, os recursos de gestão incluem as ferramentas que permitem fazer o acompanhamento e gestão do desenvolvimento da aplicação. Nesta categoria, inclui-se a ferramenta Discord. Nesta foi criado um grupo privado onde os diferentes intervenientes no desenvolvimento da aplicação podem trocar ideias (em formato texto, áudio ou vídeo), anexar arquivos para partilha com a restante equipa ou ainda agendar eventos.

Na categoria de recursos de pesquisa inclui-se essencialmente a Internet. No geral, esta será a principal fonte de informação para suportar o desenvolvimento da aplicação. Em particular, será necessário ter acesso aos sites e fóruns de aplicações como Geocaching (www.geocaching.com), CodeWars (www.codewars.com), Triposo (www.triposo.com), entre outros, durante as fases de conceção e preparação, bem como na fase de codificação (a detalhar adiante).

A secção dos recursos de reporte tratam da plataforma Overleaf. Nesta plataforma serão criados e redigidos os relatórios que documento o desenvolvimento da aplicação. Esses relatórios estarão abertos a todos os elementos das diferentes equipas que colaborem para cada relatório especifico. Assim, a redação destes será contínua e síncrona, evitando-se tarefas de agregação e compatibilização de diferentes documentos, escritos por diferentes pessoas, no momento da preparação da versão final de cada relatório.

Na categoria de recursos de desenvolvimento incluiu-se, em primeiro lugar, o *Visual Paradigm*. Esta ferramenta foi utilizada na realização dos diferentes diagramas UML que suportam a preparação do desenvolvimento da aplicação, bem como a documentação da mesma. Em segundo lugar, esta categoria inclui as ferramentas de desenvolvimento necessárias e que serão definidas pela equipa de codificação. É previsível que, pelo menos, venha a ser necessário um ambiente integrado de desenvolvimento (IDE), onde a codificação será realizada, e um serviço de base de dados, que apoiará no desenvolvimento da base de dados que suportará a aplicação. Também um serviço como o *Github* será fundamental para garantir a compatibilidade entre o trabalho realizado por cada programador individualmente, bem como entre as sucessivas versões da aplicação.

Tal como referido anteriormente, será necessário acesso às API, e respetiva documentação, da aplicação *CodeWars*. Esta é de livre acesso e pode ser consultada em `dev.codewars.com`. A API indicada poderá ser utilizada para acelerar o processo de alimentação da aplicação a desenvolver. Em particular, a API do *CodeWars* permitirá carregar uma base de dados inicial de desafios de programação com diferentes níveis de dificuldade.

1.6 Maqueta do Sistema

A Figura 1.1 ilustra a maqueta da aplicação. Nesta é possível identificar o ciclo de vida da utilização da aplicação. Tudo se inicia com o registo de um novo utilizador na aplicação. Após um registo bem-sucedido, o utilizador entra na aplicação e configura os dados da sua conta. Em particular, define uma localização inicial, doravante designada de localização *HOME*, que

não poderá alterar no decorrer da utilização da aplicação.

Figura 1.1: Maqueta da aplicação.

A tela principal da aplicação consiste num mapa do local onde o utilizador se encontra centrado na sua posição atual. Este mapa tem alguns pontos assinalados como locais de interesse, bem como a sua localização *HOME*. Em cada um dos locais de interesse assinalados será possível encontrar desafios de programação (*kata*) para resolver. Consoante os locais de interesse sejam sucessivamente mais distantes da *HOME*, também os desafios serão de um grau de complexidade progressivamente maior.

Chegado a um local de interesse o *kata* aí existente é automaticamente desbloqueado. O utilizador pode resolver o desafio no local ou optar por resolvê-lo mais tarde uma vez que este passará a estar sempre disponível. Assim que o *kata* seja resolvido, o local onde este havia sido encontrado passa a constar na lista de locais visitados pelo utilizador. Por outro lado, os pontos associados ao *kata* serão adicionados à pontuação do utilizador.

Consoante o jogo vai avançando e o utilizador vai visitando mais locais e resolvendo mais *katas*, o mapa passará a ter os locais visitados e superados assinalados com cor distinta.

1.7 Definição de um conjunto de medidas de sucesso

De modo a poder avaliar o sucesso da aplicação, e com isso validar as premissas de base consideradas na fase de conceção da mesma, foram definidas algumas métricas. Assim, para cada um dos objetivos definidos, foi definida uma forma de avaliar o grau de execução do mesmo.

Sendo a versão atual desenvolvida apenas para o público alvo do departamento de informática, estima-se que a população estudantil total abrangida será de aproximadamente 4000 alunos divididos por 2 cursos de primeiro ciclo, 5 cursos de segundo ciclo e 3 cursos de terceiro ciclo. No entanto, acredita-se que o impacto da aplicação seja maior dentro dos estudantes do curso de Engenharia Informática. Estes, considerando primeiro e segundo ciclos, representam cerca de 1000 alunos. Assim, estimando que apenas 10% destes instalem e utilizem a aplicação nos primeiros 6 meses da sua utilização, a primeira métrica de sucesso será alcançada se:

- **forem registados pelo menos 100 utilizadores nos primeiros 6 meses**

No entanto, o facto de estarem registados não implica que os utilizadores usem a aplicação e/ou pratiquem exercício físico. Assim, a segunda medida de sucesso será alcançada se:

- **metade dos utilizadores registados tiver atividade continuada durante pelo menos 2 meses**

As duas métricas anteriores permitem aferir do nível de realização dos objetivos de "redução do sedentarismo entre os jovens" e "aumento da atividade física por parte dos jovens". Quanto ao terceiro objetivo, "melhoria das competências técnicas de programação", considera-se que foi conseguido com sucesso se:

- **em média, a metade de utilizadores com atividade continuada, realizar cinco desafios de programação nos primeiros 2 meses**

O aumento do conhecimento do património dos locais onde os jovens circulem será aferido através de um questionário com perguntas sobre os locais onde os desafios foram realizados de modo a perceber o grau de conhecimento dos jovens sobre esses mesmos locais. Pretende-se assim saber se o facto de os jovens terem visitado os locais para desbloquear os desafios de programação teve como consequência o conhecimento do espaço físico envolvente ou se isso acabou por ser não se verificar. Assim, considera-se que este objetivo foi atingido se:

- **em média, a metade de utilizadores com atividade continuada, acertar em metade das perguntas que lhe serão feitas sobre o local visitado**

1.8 Plano de desenvolvimento

O plano de desenvolvimento da aplicação inclui duas fases distintas. A primeira fase coincide com o trabalho reportado no presente documento - Fundamentação e Especificação da aplicação. Esta foi previamente planeada de acordo com o diagrama de Gantt ilustrado na

Figura 1.2. Este diagrama serviu para a equipa que realizou o trabalho aqui reportado ter uma estimativa de tempo/recursos a alocar de modo a cumprir com os prazos estabelecidos.

A segunda fase do trabalho será adjudicada a uma outra equipa, pelo que o planeamento dos trabalhos associados a essa fase será essencialmente da responsabilidade dessa equipa. No entanto, é sempre bom para a entidade contratante ter uma noção dos custos e prazos associados ao trabalho que está a contratar. Adicionalmente, como, no presente caso, a entidade contratante tem conhecimento sobre os trabalhos em causa, do ponto de vista da execução, pode desde já preparar o seu planeamento interno dos trabalhos a adjudicar. Isto permitirá aferir se o valor do orçamento recebido, bem como os prazos de entrega associados, estão dentro do expectável para o tipo de trabalho. Assim, na Figura 1.3 apresenta-se o diagrama de Gantt relativo aos trabalhos da segunda fase.

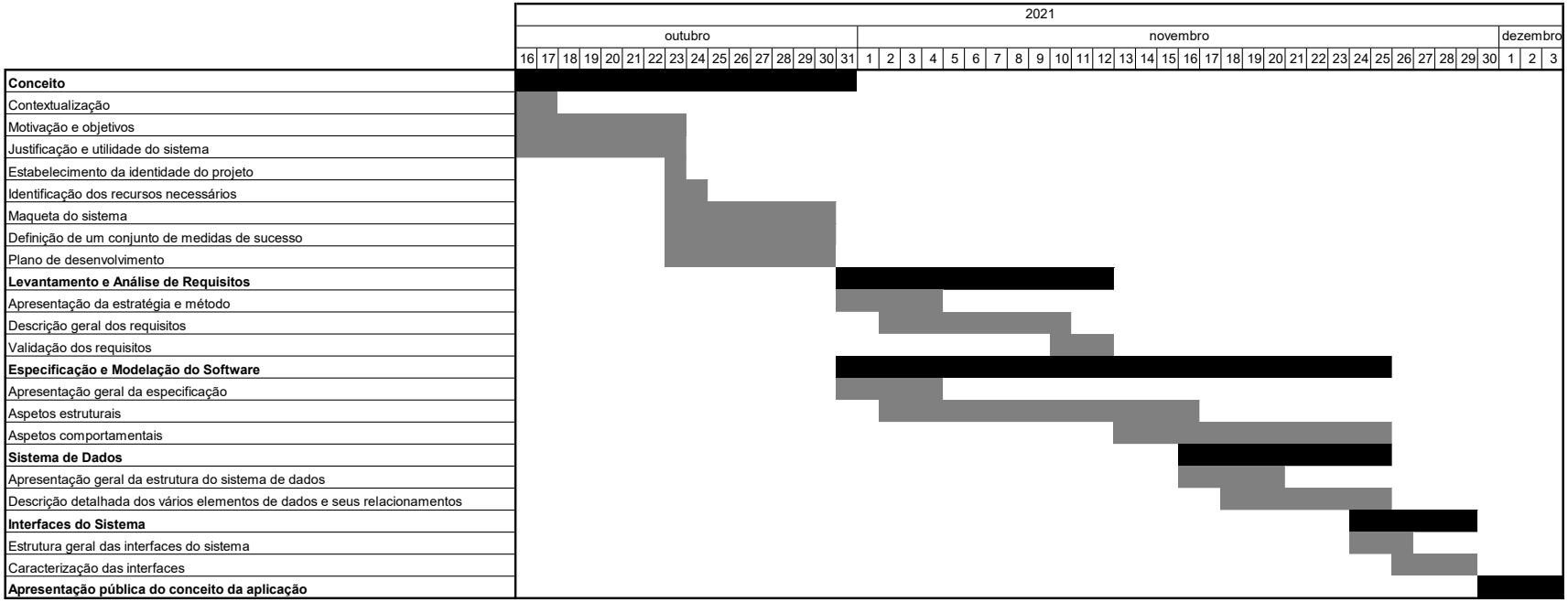

Figura 1.2: Diagrama de Gantt da fase de Fundamentação e Especificação.

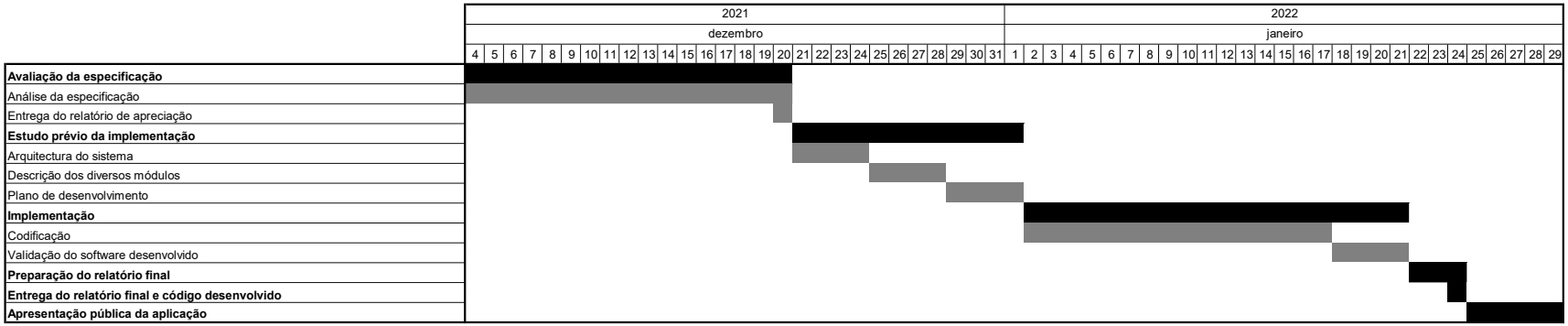

Figura 1.3: Diagrama de Gantt da fase de Implementação.

2 Levantamento e Análise de Requisitos

Após fundamentar a aplicação a ser desenvolvida, de forma clara e sucinta, pode-se avançar para o levantamento e análise dos requisitos que esta deverá satisfazer. No presente capítulo serão apresentados os requisitos funcionais e não funcionais da aplicação a ser implementada.

2.1 Apresentação da Estratégia e Método

A estratégia dos requisitos passou pela análise de algumas aplicações que serviram de inspiração para a aplicação a desenvolver no presente trabalho. Nomeadamente, para a componente do guia de locais de interesse foram visitadas e analisadas as aplicações Geocaching (`geocaching.com`) e Triposo (`triposo.com`). Para a componente motivacional, i.e. desafios de programação, foi analisada a aplicação *CodeWar*s (`codewars.com`).

O método de levantamento de requisitos consistiu no registo das impressões dos elementos do grupo durante a utilização das aplicações referidas. Após o registo nestas aplicações, foi possível realizar alguns testes e simulações de modo a identificar e definir as funcionalidades da aplicação em mãos.

Para lém disso, algumas destas aplicações possuem fóruns onde foi possível recolher mais alguns requisitos com base na experiência de utilização dos seus utilizadores. Em particular, a aplicação *CodeWars* possui um fórum (`codewars.com/topics`) e um canal de discussão no *Github* (`github.com/codewars/codewars.com/discussions`) onde uma comunidade bastante ativa de utilizadores vai fornecendo sugestões quase diárias de melhorias na aplicação.

Também a aplicação *Geocaching* possui um fórum (`forums.geocaching.com/GC/index.php`) de utilizadores bastante ativos (as atualizações neste caso ocorrem várias vezes ao dia).

Assim, considerando todas estas fontes de informação, bem como o gosto pessoal dos elementos do grupo, foi possível definir os requisitos que se apresentam nas secções seguintes.

2.2 Descrição Geral dos Requisitos Levantados

De modo a facilitar a posterior implementação do sistema de base de dados, os requisitos levantados foram agrupados em funcionais e não funcionais, conforme detalhado a seguir.

2.2.1 Requisitos Funcionais

Em engenharia de *software*, um requisito funcional define uma função do sistema ou componente. Isot é, um requisito funcional representa algo que o *software* faz, em termos de tarefas e serviços.

Requisitos do Sistema

- O sistema fornece uma página específica onde o utilizador tem a escolha de alterar as informações relativas à sua conta;
- O sistema disponibiliza um mapa interativo e uma ferramenta de pesquisa, permitindo aos utilizadores a seleção de uma zona;
- O sistema fornece uma página específica onde o utilizador tem à escolha realizar e consultar as informações de diferentes *katas*;
- O sistema fornece uma página específica onde o utilizador pode consultar a posição do *ranking* onde se encontra

Requisitos do Utilizador Geral

- O utilizador define/altera o nome de utilizador;
- O utilizador define/altera o raio de visão que permite observar os diferentes *katas*;
- O utilizador define de que tipos de desafios pretende receber notificações;
- O utilizador seleciona uma zona através do mapa iterativo;
- O utilizador usa a ferramenta de pesquisa para encontrar determinadas zonas geográficas;
- O utilizador usa a ferramenta de pesquisa para encontrar determinados *katas*;
- O utilizador pode visualizar a informação do local visitado;
- O utilizador pode registar/iniciar sessão com o seu endereço de email e a palavra-passe;

Requisitos do Administrador

- O administrador define/altera o nome de utilizador;
- O administrador define/altera o raio de visão que permite observar os diferentes *katas*;
- O administrador usa a ferramenta de pesquisa para encontrar determinados *katas*;
- O administrador usa a ferramenta de pesquisa para encontrar determinadas zonas geográficas;
- O administrador pode registar/iniciar sessão com o seu endereço de email e a palavra-passe;
- O administrador cria um *kata*;
- O administrador gere um *kata*;
- O administrador elimina um *kata*;

2.2.2 Requisitos não Funcionais

Requisitos não funcionais são os requisitos relacionados ao uso da aplicação em termos de desempenho, usabilidade, fiabilidade, segurança, disponibilidade, manutenção e tecnologias envolvidas.

- O sistema deve, para cada *kata*, possuir uma entrada na base de dados com as várias informações relativas à zona;
- O sistema deve armazenar os dados relativos ao utilizadores registados, de modo a que possam iniciar sessão;
- O sistema deve ser capaz de consultar as *API* utilizadas de forma a extrair os seu dados;
- O sistema deve, para cada utilizador, possuir uma entrada na base de dados com o *ranking* no qual se encontra posicionado;

2.3 Validação dos requisitos estabelecidos

Antes de se iniciar a implementação da aplicação é importante ter uma noção muito clara de quais os requisitos que a mesma deve cumprir. Isto consegue-se através do levantamento dos requisitos junto da entidade promotora.

Após se terem identificados os principais requisitos da aplicação e estes terem sido

devidamente descritos, é importante que estes sejam comunicados ao promotor de modo a validar as premissas de base antes de se iniciar o desenvolvimento. Por outro lado, como nem sempre a comunicação entre as partes é a melhor, convém que os requisitos identificados sejam validados o mais cedo possível, de modo a evitar ter de refazer trabalho já feito por este não corresponder ao esperado.

No caso da presente aplicação, a entidade promotora são os próprios elementos do grupo responsável pelo presente relatório, pelo que esta etapa acabou por ser mais simples. Ainda assim, é importante ter noção do porquê de se realizar a validação dos requisitos para projectos futuros.

3 Especificação e Modelação do Software

O presente capitulo começa por apresentar um possível modelo que traduz o funcionamento global da aplicação que se pretende desenvolver. De seguida são apresentados alguns detalhes sobre os principais aspetos estruturais e comportamentais que a aplicação deve conter. A apresentação de todos estes aspetos e modelos será baseada em diagramas UML adequados e adiante descritos.

3.1 Apresentação geral da especificação

Usando um dispositivo móvel (e.g. smartphone), um utilizador registado na aplicação *Code&GO* tem acesso a um mapa, por defeito centrado na sua localização atual. Nesse mapa estão assinalados os locais de interesse a ser visitados. O utilizador deverá escolher um dos locais e dirigir-se para lá de modo a desbloquear o desafio de programação (designado por *kata*) a ele associado.

Chegado ao local, o *kata* que aí se encontrava passa a estar disponível para o utilizador resolver. Após resolvido o *kata*, tanto o local de interesse como o *kata* associado passam a constar do histórico do utilizador.

Sendo o objetivo da aplicação promover o exercício físico, quanto mais longe da localização *HOME* se encontrar o local de interesse visitado, maior será a pontuação a atribuir e mais difícil será o *kata* aí existente.

Para tal ser possível, terá de existir uma base de dados de locais de interesse e outra de *katas*. Dependendo da distância da localização *HOME* a cada local de interesse, então o *kata* a associar aos locais terá de ter um grau de dificuldade distinto. Isto implica que, para utilizadores distintos, o mesmo *kata* apareça em locais distintos pois a localização *HOME* de cada utilizador é distinta, tal como será a distância aos locais de interesse.

As bases de dados de locais de interesse e *katas* tanto poderão ser alimentadas pelos administradores da aplicação como pelos utilizadores.

A Figura 3.1 apresenta o modelo de domínio para a aplicação em causa. Neste é possível ver as principais entidades da aplicação e as suas relações. Note-se que estas entidades e

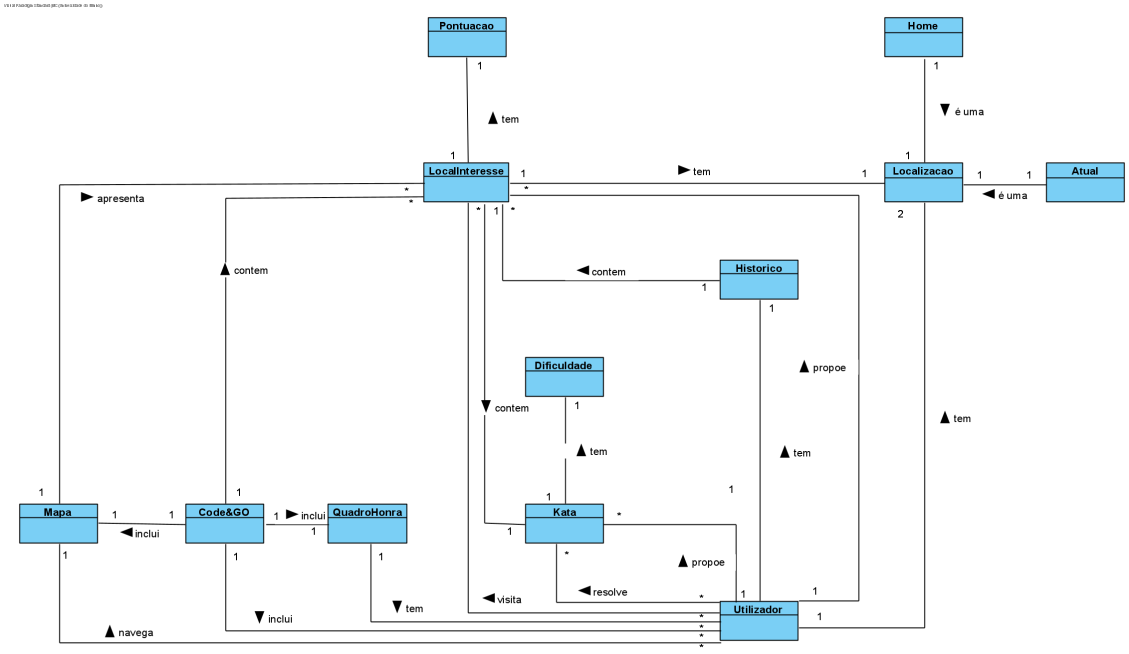

Figura 3.1: Modelo de domínio da aplicação.

relações são apenas candidatas a existir na arquitetura final da aplicação. De igual modo, outras entidades/relações poderão/deverão ser adicionadas de modo a satisfazer os requisitos definidos anteriormente.

3.2 Aspetos comportamentais

De modo a ilustrar os principais requisitos funcionais da aplicação, optou-se por detalhar os principais cenários de utilização correspondentes às principais funcionalidades que se espera que a aplicação tenha. Assim, na Figura 3.2 apresenta-se o diagrama de casos de uso a considerar na presente versão da aplicação. Note-se que, por forma a não prejudicar a análise deste diagrama, foi omitida a dependência entre o caso de uso "Iniciar Sessão"e os demais. No entanto, com exceção para o caso de uso "Registar", todos os outros só serão possíveis de ocorrer caso o "Iniciar Sessão"tenha ocorrido antes.

Nos parágrafos que se seguem cada caso de uso será detalhado de modo a facilitar a sua posterior implementação.

3.2.1 Casos de uso do administrador

Estão previstos cinco casos de uso específicos para o administrador. Nomeadamente,

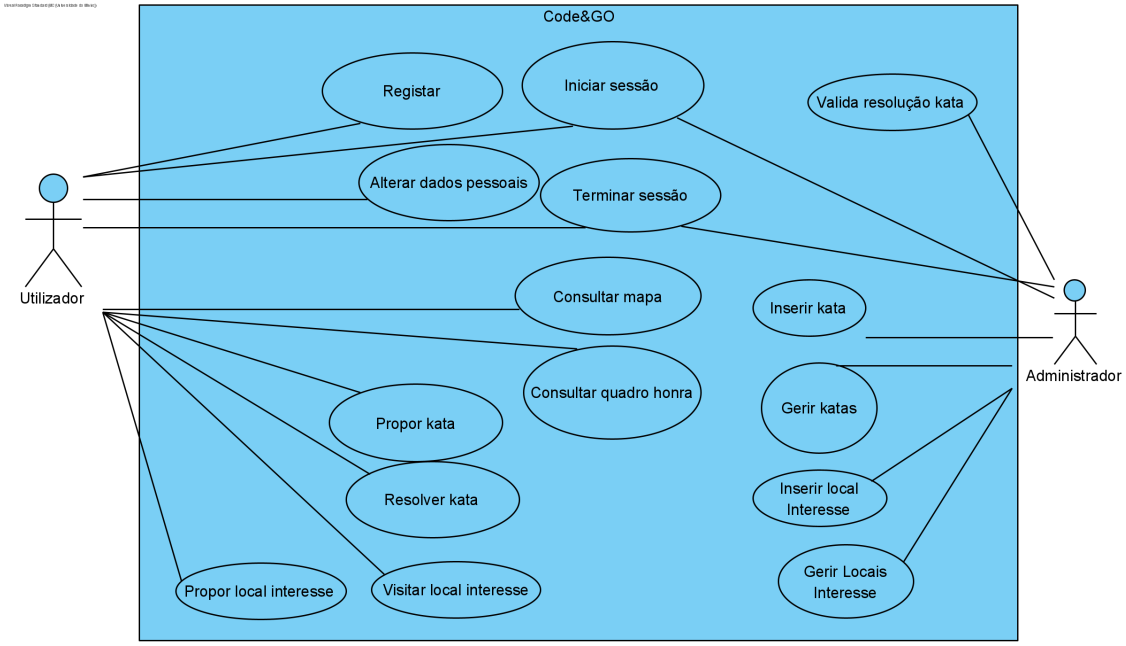

Figura 3.2: Diagrama de casos de uso

quatro dizem respeito à inserção e a gestão dos locais de interesse e dos katas. Esta gestão tem dois grandes cenários de intervenção por parte do administrador. Por um lado, é da sua responsabilidade ir alimentando a aplicação com novos locais de interesse e novos katas para enriquecer a aplicação. Por outro lado, tem de validar os locais de interesse e katas propostos pelos utilizadores. Em particular, deverá verificar que os locais de interesse não sejam duplicados ou inadequados, entre outros aspetos. Em relação aos katas, será responsabilidade do administrador definir o grau de dificuldade de cada uma das propostas recebidas pelo sistema.

O quinto caso de uso específico do administrador está associado à validação da resolução dos katas por parte dos utilizadores. EM versões futuras da aplicação, esta validação deverá ser automaticamente feita pela aplicação, à semelhança do que acontece na aplicação CodeWars. No entanto, para esta versão inicial da aplicação, e de modo a simplificar a sua implementação, optou-se por delegar no administrador esta tarefa.

Adicionalmente, o administrador tem mais dois casos de uso relacionados com o início e fim de sessão na aplicação. Estes são comuns aos utilizadores ordinários, pelo que serão detalhados adiante junto com os demais casos de uso dos utilizadores.

De seguida apresenta-se a lista detalhada de cada um dos casos de uso referido.

- **Caso de uso: Inserir katas**
  Descrição: O administrador pode inserir novos katas no sistema.
  Cenário: O administrador pretende adicionar novos katas no sistema.
  Pré-Condição: -

Pós-Condição: Os katas inseridos passam a estar disponíveis no sistema.
Fluxo Normal:

1. O administrador pede para inserir novo kata no sistema indicando o seu nome
2. O sistema verifica se o nome está a ser utilizado
3. O administrador insere informação necessária para definir novo kata
4. O sistema valida os dados inseridos
5. O novo kata é armazenado na base de dados de katas do sistema

Fluxo Alternativo (1): [O nome inserido já está a ser utilizado] (passo 2)

2.1. O sistema verifica que o nome inserido já está a ser utilizado

2.2. O sistema alerta o administrador

2.3. Regressa a 1

Fluxo Alternativo (2): [A informação inserida não é válida] (passo 4)

4.1. O sistema verifica que a informação inserida não é válida

4.2. O sistema alerta o administrador

4.3. Regressa a 3

- **Caso de uso: Gerir katas**
  Descrição: O administrador pode gerir os katas propostos pelos utilizadores.
  Cenário: O administrador pretende verificar se os katas popostos pelos utilizadores são válidos para os adicionar ao sistema.
  Pré-Condição: Existem katas propostos pelos utilizadores à espera de aceitação.
  Pós-Condição: Os katas propostos passam a estar disponíveis no sistema.
  Fluxo Normal:

  1. O administrador pede ao sistema a lista de katas propostos
  2. O sistema fornece a lista de katas propostos
  3. O administrador seleciona um kata e valida a informação necessária para definir novo kata a partir dessa proposta
  4. O administrador classifica o kata de acordo com o seu grau de dificuldade
  5. O administrador insere o novo kata no sistema
  6. O novo kata passa a estar disponível na base de dados de katas do sistema

7. O administrador repete os passos 3 a 5 para os restantes katas na lista de propostas

Fluxo Alternativo (1): [A informação inserida não é válida] (passo 5)

5.1. O sistema deteta que a informação inserida não é válida

5.2. O sistema alerta o administrador

5.3. Regressa a 3

Fluxo Alternativo (2): [O kata proposto não tem condições de ser aceite] (passo 3)

2.1. O administrador não valida a informação do kata e rejeita-o

2.2. Regressa ao passo 3 e seleciona novo kata para analisar

- **Caso de uso: Inserir locais de interesse**
  Descrição: O administrador pode inserir novos locais de interesse no sistema.
  Cenário: O administrador pretende adicionar novos locais de interesse no sistema.
  Pré-Condição: -
  Pós-Condição: Os locais de interesse inseridos passam a estar disponíveis no sistema.
  Fluxo Normal:

  1. O administrador pede para inserir novo local de interesse no sistema indicando as suas coordenadas
  2. O sistema verifica se já existe local de interesse naquelas coordenadas
  3. O administrador insere informação necessária para definir novo local de interesse
  4. O sistema valida os dados inseridos
  5. O novo local de interesse é armazenado na base de dados de locais de interesse do sistema

  Fluxo Alternativo (1): [A informação inserida não é válida] (passo 4)

  4.1. O sistema verifica que a informação inserida não é válida

  4.2. O sistema alerta o administrador

  4.3. Regressa a 3

  Fluxo Exceção (1): [As coordenadas inseridas já existem no sistema] (passo 2)

  2.1. O sistema verifica que já existe local de interesse naquelas coordenadas

  2.2. O sistema alerta o administrador

2.3. Termina processo sem que o local de interesse tenha sido adicionado

- **Caso de uso: Gerir locais de interesse**
  Descrição: O administrador pode gerir os locais de interesse propostos pelos utilizadores.
  Cenário: O administrador pretende verificar se os locais de interesse popostos pelos utilizadores são válidos para os adicionar ao sistema.
  Pré-Condição: Existem locais de interesse propostos pelos utilizadores à espera de aceitação.
  Pós-Condição: Os locais de interesse propostos passam a estar disponíveis no sistema.
  Fluxo Normal:

  1. O administrador pede ao sistema a lista de locais de interesse propostos
  2. O sistema fornece a lista de locais de interesse propostos
  3. O administrador seleciona um local de interesse e valida a informação necessária para definir novo locais de interesse a partir dessa proposta
  4. O administrador insere o novo local de interesse no sistema
  5. O novo local de interesse passa a estar disponível na base de dados de locais de interesse do sistema
  6. O administrador repete os passos 3 e 4 para os restantes locais de interesse na lista de propostas

  Fluxo Alternativo (1): [A informação inserida não é válida] (passo 4)

  4.1. O sistema deteta que a informação inserida não é válida

  4.2. O sistema alerta o administrador

  4.3. Regressa a 3

  Fluxo Exceção (1): [O local de interesse proposto não tem condições de ser aceite] (passo 3)

  2.1. O administrador não valida a informação do local de interesse e rejeita-o

  2.2. Regressa ao passo 3 e seleciona novo local de interesse para analisar

- **Caso de uso: Validar resolução de katas**
  Descrição: O administrador pode validar as resoluções dos katas propostas pelos utilizadores.
  Cenário: O administrador pretende verificar se as resoluções dos katas propostas pelos utilizadores são válidos.
  Pré-Condição: Existem resoluções de katas propostas pelos utilizadores à espera de validação.

Pós-Condição: As resoluções de katas propostas pelos utilizadores passam a estar classificadas como corretas ou não.
Fluxo Normal:

1. O administrador pede ao sistema a lista das resoluções de katas propostas pelos utilizadores não corretas
2. O sistema fornece a lista das resoluções de katas propostas pelos utilizadores não corretas
3. O administrador seleciona uma resolução de kata e valida se esta está correta
4. O administrador classifica a resolução de kata como correta ou incorreta
5. O administrador repete os passos 3 e 4 para as restantes resoluções de katas na lista de propostas

Fluxo Exceção (1): [O administrador não quer selecionar resolução de kata] (passo 3)

2.1. O administrador não quer selecionar resolução de kata para análise

2.2. Termina o processo

3.2.2 Casos de uso do utilizador

No caso do utilizador estão previstos vários casos de uso agrupados, por tipo de funcionalidade, em quatro conjuntos. O primeiro conjunto diz respeito a cenários de cariz administrativo. Isto inclui o registo na aplicação e a alteração posterior dos dados do utilizador, bem como o inicio e fim de sessão na aplicação. O segundo conjunto diz respeito às consultas. Essencialmente existirão consultas ao mapa, para identificar novos locais de interesse, e consultas ao quadro de honra, um dos elementos motivacionais da aplicação. O terceiro conjunto diz respeito aos cenários envolvendo os katas. É possível ao utilizador resolver um kata existente ou propôr um novo kata para a aplicação. O último conjunto está associado aos locais de interesse. Tal como no anterior, será possível visitar um local de interesse ou propor um novo local de interesse.

De seguida apresenta-se a lista detalhada de cada um dos casos de uso referido.

- **Caso de uso: Registar**
  Descrição: O utilizador cria uma conta.
  Cenário: O utilizador não possui uma conta no sistema e pretende criar uma.
  Pré-Condição: -
  Pós-Condição: O utilizador cria uma conta.
  Fluxo Normal:

  1. O utilizador insere nome, email e password

2. O sistema verifica se o nome está a ser utilizado

3. O sistema verifica se o email está a ser já utilizado

4. A conta é criada e armazenada no sistema

Fluxo Alternativo (1): [O email inserido já está a ser utilizado] (passo 1)

1.1. O sistema deteta que o email inserido já está a ser utilizado por outra conta

1.2. O sistema alerta o utilizador

1.3. Regressa a 1

Fluxo Alternativo (2): [O nome inserido já está a ser utilizado] (passo 1)

1.1. O sistema deteta que o nome inserido já está a ser utilizado por outra conta

1.2. O sistema alerta o utilizador

1.3. Regressa a 1

- **Caso de uso: Iniciar Sessão**
  Descrição: O utilizador inicia sessão.
  Cenário: O utilizador pretende utilizar a aplicação, para isto é necessário iniciar a sessão na sua conta Code&GO.
  Pré-Condição: A conta de utilizador existe no sistema.
  Pós-Condição: O utilizador encontra-se com a sessão inciado no sistema.
  Fluxo Normal:

  1. O utilizador insere as suas credenciais

  2. A sessão é iniciada

  Fluxo Alternativo: [Credenciais incorretas] (passo 1)

  1.1. O sistema informa que as credenciais estão incorretas

  1.2. Regressa a 1

- **Caso de uso: Terminar Sessão**
  Descrição: O utilizador termina a sessão.
  Cenário: O utilizador tem a sessão iniciada e agora pretende encerrá-la.
  Pré-Condição: O utilizador encontra-se com a sessão iniciada.
  Pós-Condição: O utilizador termina a sessão.
  Fluxo Normal:

  1. O utilizador seleciona a opção de terminar a sessão

2. A sessão é terminada

- **Caso de uso: Alterar dados Pessoais**
  Descrição: O utilizador altera os dados pessoais.
  Cenário: O utilizador encontra-se com a sessão inciada e pretende alterar os dados da sua conta.
  Pré-Condição: O utilizador tem a sessão iniciada.
  Pós-Condição: O utilizador altera os dados da sua conta.
  Fluxo Normal:

  1. O utilizador insere os dados novos
  2. O sistema valida os novos dados
  3. A conta é atualizada com os novos dados

  Fluxo Alternativo: [Os novos dados são inválidos] (passo 2)

  2.1. O sistema informa que os dados inseridos são inválidos

  2.2. Regressa a 1

- **Caso de uso: Consultar mapa**
  Descrição: O utilizador pode ver locais de interesse no mapa.
  Cenário: O utilizador pretende verificar os locais de interesse disponiveis no mapa para selecionar novo local a visitar.
  Pré-Condição: -
  Pós-Condição: O utilizador visualiza o mapa e os eventuais locais de interesse disponíveis.
  Fluxo Normal:

  1. O utilizador pede ao sistema para ver o mapa
  2. O sistema devolve o mapa
  3. O utilizador navega pelo mapa

- **Caso de uso: Consultar quadro de honra**
  Descrição: O utilizador pode ver o quadro de honra
  Cenário: O utilizador pretende ver o quadro de honra para saber em que posição se encontra.
  Pré-Condição: -
  Pós-Condição: O utilizador visualiza o quadro de honra.
  Fluxo Normal:

  1. O utilizador pede ao sistema para ver o quadro de honra
  2. O sistema devolve o quadro de honra

3. O utilizador navega pelo quadro de honra

- **Caso de uso: Propor katas**
  Descrição: O utilizador pode propor novos katas.
  Cenário: O utilizador pretende propor novos katas para o sistema.
  Pré-Condição: -
  Pós-Condição: O kata proposto passa a estar disponível no sistema.
  Fluxo Normal:

  1. O utilizador pede para inserir novo kata no sistema indicando o seu nome
  2. O sistema verifica se o nome está a ser utilizado
  3. O utilizador insere informação necessária para definir novo kata
  4. O sistema valida os dados inseridos
  5. O novo kata é armazenado na lista de katas propostos para posterior validação pelo administrador

  Fluxo Alternativo (1): [O nome inserido já está a ser utilizado] (passo 2)

  2.1. O sistema verifica que o nome inserido já está a ser utilizado

  2.2. O sistema alerta o utilizador

  2.3. Regressa a 1

  Fluxo Alternativo (2): [A informação inserida não é válida] (passo 4)

  4.1. O sistema verifica que a informação inserida não é válida

  4.2. O sistema alerta o utilizador

  4.3. Regressa a 3

- **Caso de uso: Resolver katas**
  Descrição: O utilizador pode resolver um kata.
  Cenário: O utilizador pretende resolver um kata introduzindo a sua solução no sistema.
  Pré-Condição: O utilizador tem katas por resolver ou katas cuja resposta foi considerada incorreta pelo administrador.
  Pós-Condição: O kata fica com uma resposta inserida a aguardar por validação do administrador.
  Fluxo Normal: Fluxo Normal:

  1. O utilizador pede ao sistema a lista dos seus katas por resolver
  2. O sistema fornece a lista dos seus katas por resolver

3. O utilizador seleciona um kata para resolver
4. O utilizador insere o código com a sua resposta e submete-o
5. O utilizador repete os passos 3 e 4 para os restantes katas na lista dos seus katas por resolver

Fluxo Exceção (1): [O utilizador não pretende resolver kata] (passo 3)

2.1. O utilizador não escolhe kata para resolver

2.2. Termina o processo

- **Caso de uso: Propor local de interesse**
  Descrição: O utilizador pode propor novo local de interesse.
  Cenário: O utilizador pretende propor novo local de interesse para o sistema.
  Pré-Condição: -
  Pós-Condição: O local de interesse proposto passa a estar disponível no sistema.
  Fluxo Normal:

  1. O utilizador pede para inserir novo local de interesse no sistema indicando as suas coordenadas
  2. O sistema verifica se já existe local de interesse naquelas coordenadas
  3. O utilizador insere informação necessária para definir novo local de interesse
  4. O sistema valida os dados inseridos
  5. O novo local de interesse é armazenado na lista de locais de interesse propostos para posterior validação pelo administrador

  Fluxo Alternativo (1): [A informação inserida não é válida] (passo 4)

  4.1. O sistema verifica que a informação inserida não é válida

  4.2. O sistema alerta o utilizador

  4.3. Regressa a 3

  Fluxo Exceção (1): [As coordenadas inseridas já existem no sistema] (passo 2)

  2.1. O sistema verifica que já existe local de interesse naquelas coordenadas

  2.2. O sistema alerta o utilizador

  2.3. Termina processo sem que o local de interesse tenha sido adicionado

- **Caso de uso: Visitar local de interesse**

Descrição: O utilizador pode visitar locais de interesse que constem no mapa.
Cenário: O utilizador pretende visitar um local de interesse disponiveis no mapa.
Pré-Condição: A consulta do mapa está ativa.
Pós-Condição: O local de interesse escolhido passa a constar no histórico do utilizador. O kata aí existente passa a constar na lista de katas por resolver do utilizador.
Fluxo Normal:

1. O utilizador seleciona no mapa um local de interesse a visitar
2. O sistema valida se as coordenadas atuais do utilizador coincidem com as do local de interesse
3. O local de interesse é adicionado ao histórico do utilizador
4. O kata existente no local de interesse selecionado é adicionado à lista de katas por resolver do utilizador

Fluxo Exceção (1): [O utilizador não se encontra no local de interesse] (passo 2)

2.1. O sistema verifica que as coordenadas do local de interesse são diferentes das do utilizador

2.2. O sistema alerta o utilizador

2.3. Termina processo

Fluxo Exceção (2): [O local de interesse já foi visitado] (passo 2)

2.1. O sistema verifica que as coordenadas do local de interesse são iguais às do utilizador

2.2. O sistema verifica que o local de interesse consta no histórico do utilizador

2.3. Termina processo

3.3 Aspetos estruturais

Com base nos casos de uso definidos anteriormente, e tendo em conta as diferentes funcionalidades que o sistema terá de suportar por forma a responder a esses casos de uso, foi possível definir uma primeira proposta de arquitetura para o sistema a desenvolver. Esta materializa-se no diagrama de pacotes apresentados na Figura 3.3.

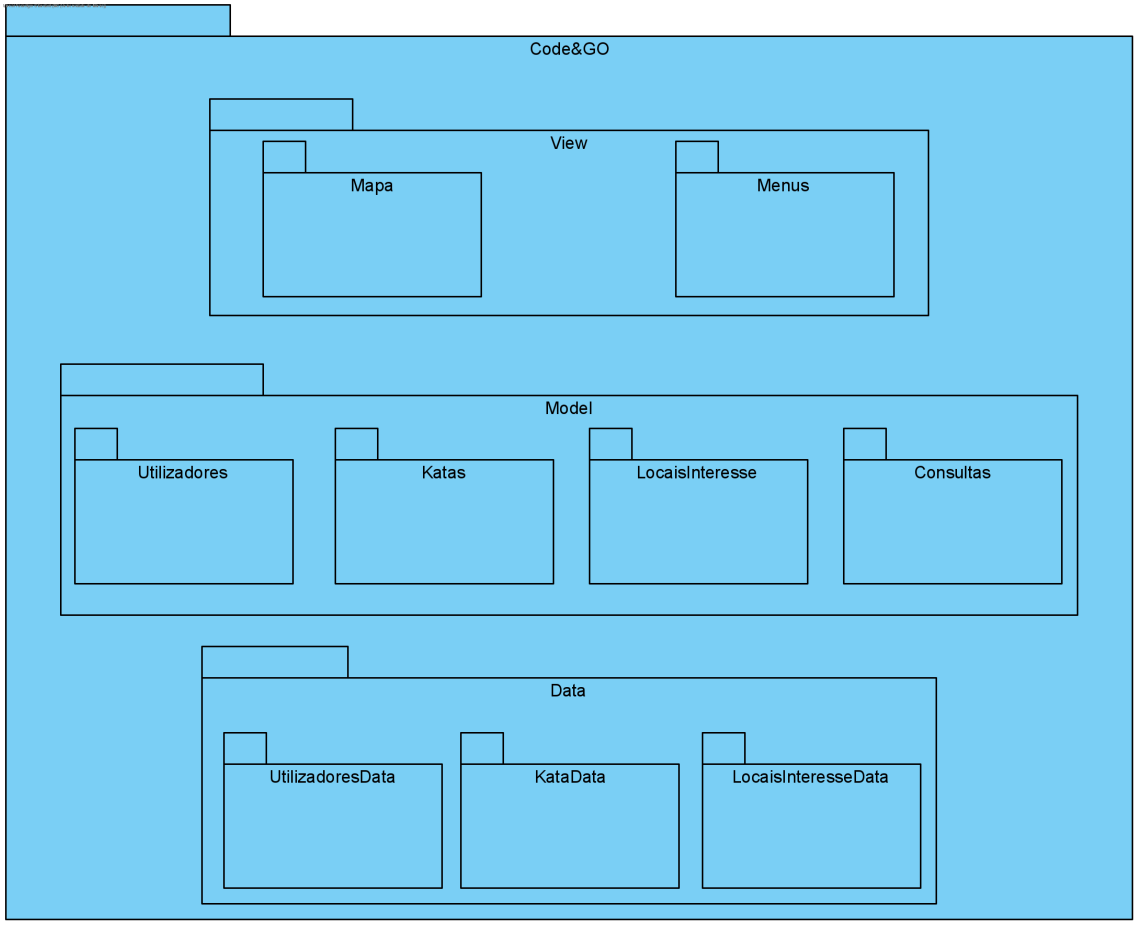

Figura 3.3: Diagrama de pacotes proposto para a aplicação.

A divisão de pacotes apresentada consiste na definição de três pacotes principais, seguindo a metodologia MVC. Um pacote contendo os componentes responsáveis pela interface com o utilizador, designado *View*. Neste estarão, agrupadas em dois pacotes, as classes associadas à apresentam dos mapas e dos demais menus.

Um outro pacote, designado *Model*, contém toda a lógica de negócio da aplicação. De modo a mais facilmente separar as diferentes funcionalidades que a aplicação deverá suportar, prevê-se a criação de pelo menos quatro pacotes distintos dentro do *Model*. Um para as classes com responsabilidade sobre a gestão dos utilizadores da aplicação. Outro para as classes associadas à gestão dos *katas*, incluindo a sua validação. Um outro para as classes responsáveis pelas funcionalidades relativas aos locais de interesse. Por fim, um outro pacote para gerir as demais funcionalidades de consulta genérica à aplicação.

O terceiro grande pacote, seguindo o padrão MVC, diz respeito ao pacote de dados, designado por *Data*. Neste estarão definidas as funcionalidades relacionadas com as trocas de informação entre a aplicação e o sistema de base de dados de suporte. Uma vez que se prevê sobretudo o armazenamento persistente de informação relativa aos utilizadores, locais

de interesse e *katas*, sugere-se assim a criação de pacotes para as classes que gerem estes três tipos de dados.

Deve notar-se que a solução final de arquitetura é da responsabilidade da equipa que fará a implementação e codificação da aplicação. Com efeito, aspetos como o tipo de linguagem e tecnologia a utilizar podem conduzir à alteração da arquitetura proposta.

4 Conceção do Sistema de Dados

O sistema de dados que servirá de suporte à aplicação a desenvolver tem de ser pensado de modo a permitir responder aos requisitos levantados anteriormente. Assim, deverá conter um conjunto de entidades representativo das principais funcionalidades da aplicação. Estas deverão ter definidos entre si relacionamentos que vão de encontro aquelas funcionalidades.

4.1 Apresentação geral da estrutura do sistema de dados

Na Figura 4.1 apresenta-se o modelo lógico da base de dados a implementar. Este foi baseado no *MySQL Workbench* pelo que, no caso de o serviço de base de dados a adotar em fase de implementação não ser o *MySQL*, poderão ser necessárias alterações à solução proposta.

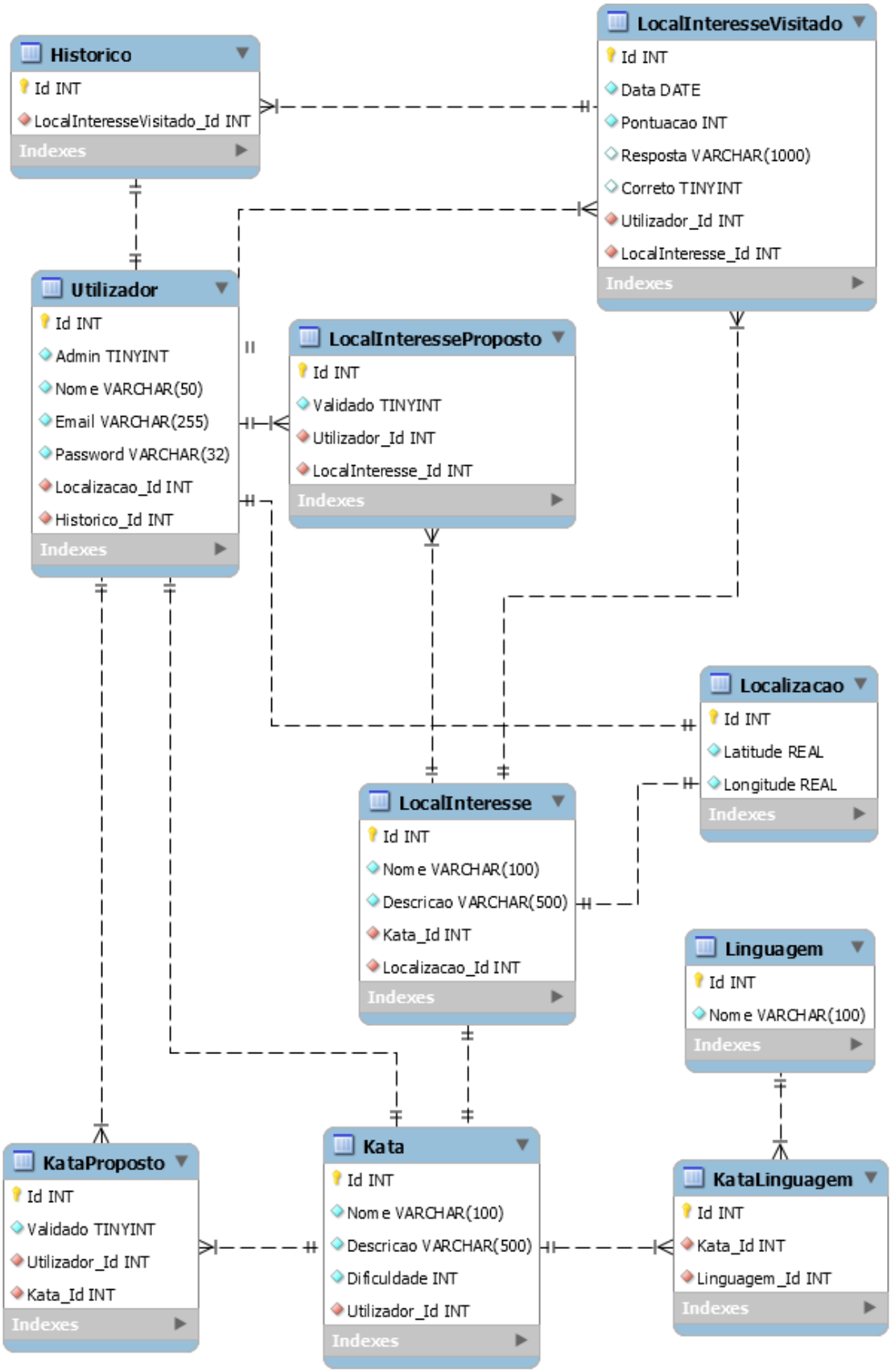

Figura 4.1: Modelo lógico da base de dados da aplicação.

Conforme ilustrado na figura, existem essencialmente três entidades principais na base de dados, a saber, *Utilizador*, *Kata* e *LocalInteresse*. Tendo em conta que deverá ser mantido o registo dos *katas* e dos locais de interesse propostos pelos utilizadores, o modelo de base de dados prevê as tabelas *KataProposto* e *LocalInteresseProposto* para este efeito.

Existe ainda a necessidade de manter um registo do histórico de utilização da aplicação por parte do utilizador. Isto levou à necessidade de prever as tabelas *Historico* e *LocalInteresseVisitado*.

As restantes três tabelas são necessárias para funcionalidades auxiliares da aplicação. Nomeadamente, a tabela *Localizacao* permitirá facilmente alimentar o mapa com a indicação de todos os locais de interesse e localização de utilizadores existente na aplicação. As tabelas *Linguagem* e *KataLinguagem* permitem definir as linguagens de programação em que cada *kata* pode ser resolvido, bem como saber quantas e quais as linguagens utilizadas pelos diferentes *katas* da aplicação.

De modo a permitir povoar a primeira versão da aplicação, será necessário fornecer informação sobre um conjunto de locais de interesse e de *katas*. Em relação aos locais de interesse, sendo estes apenas espaços existentes no Campus de Gualtar da Universidade do Minho, recomenda-se que nesta primeira versão se usem os espaços definidos no site que a universidade disponibiliza com o mapa do campus (`campi.uminho.pt/CG.html`).

No caso dos *Katas*, e com o intuito de facilitar o povoamento desta primeira versão, sugere-se a utilização da API da aplicação *CodeWars* (`dev.codewars.com`). Conforme descrito na documentação, é muito simples obter a listagem dos *katas* propostos por diferentes utilizadores. De seguida, é possível obter os detalhes de cada *kata*.

Na Figura 4.2 apresenta-se um diagrama detalhando o fluxo de informação utilizando a API do *CodeWars*. Primeiro, será necessário aplicar as *queries* que a API disponibiliza para obter listas de *katas*. O segundo passo será trabalhar essas listas de modo a obter apenas os campos descrição e dificuldade pois apenas esses serão usados na presente aplicação. O terceiro passo será adicionar os dados na base de dados da aplicação. Por fim, o quarto passo, consiste em fazer testes para confirmar que os dados introduzidos aparecem corretamente aos utilizadores.

Figura 4.2: Fluxo de dados utilizando API CodeWars.

4.2 Descrição detalhada dos vários elementos de dados e seus relacionamentos

Nos parágrafos que se seguem detalha-se cada uma das tabelas da base de dados, ou seja, detalham-se as diferentes entidades existentes na aplicação e os relacionamentos entre estas.

Começa por se descrever as entidades:

- Utilizador
  Dos atributos abaixo indicados nenhum poderá ser nulo.
  - Id: chave que identifica inequivocamente cada Utilizador usando um valor inteiro;
  - Admin: indica se o Utilizador é ou não administrador da aplicação, recorrendo a um valor booleano;
  - Nome: string de até 50 caracteres indicando o nome do Utilizador;
  - Email: string de até 255 caracteres indicando o email do Utilizador;
  - Password: string de até 32 caracteres indicando a password do Utilizador;
  - Localizacao_Id: valor inteiro que identifica na tabela Localizacao, a localização do Utilizador;
  - Historico_Id: valor inteiro que identifica na tabela Historico, o histórico do Utilizador.
- Kata
  Dos atributos abaixo indicados nenhum poderá ser nulo.
  - Id: chave que identifica inequivocamente cada Kata usando um valor inteiro;
  - Nome: string de até 100 caracteres indicando o nome do Kata;
  - Descricao: string de até 500 caracteres indicando a descrição detalhada do Kata, ou seja, explicando o problema a resolver;
  - Dificuldade: valor inteiro indicando o nível de dificuldade do Kata;
  - Utilizador_Id: valor inteiro que identifica na tabela Utilizador, o utilizador que criou o Kata, podendo este ser um utilizador ordinário ou o próprio administrador da aplicação.
- LocalInteresse
  Dos atributos abaixo indicados nenhum poderá ser nulo.

  - Id: chave que identifica inequivocamente cada LocalInteresse usando um valor inteiro;
  - Nome: string de até 100 caracteres indicando o nome do LocalInteresse;
  - Descricao: string de até 500 caracteres indicando a descrição detalhada do LocalInteresse, ou seja, explicando qual o aspeto com interesse que é possível ver naquele local;
  - Kata_Id: valor inteiro que identifica na tabela Kata o Kata que existe em cada local de interesse;
  - Localizacao_Id: valor inteiro que identifica na tabela Localizacao a localização de cada local de interesse.

- Localizacao
  Dos atributos abaixo indicados nenhum poderá ser nulo.
  - Id: chave que identifica inequivocamente cada Localizacao usando um valor inteiro;
  - Latitude: valor real indicando a latitude da Localizacao;
  - Longitude: valor real indicando a longitude da Localizacao.

- Linguagem
  Dos atributos abaixo indicados nenhum poderá ser nulo.
  - Id: chave que identifica inequivocamente cada Linguagem de programação usando um valor inteiro;
  - Nome: string de até 100 caracteres indicando o nome da Linguagem de programação.

- Historico
  Dos atributos abaixo indicados nenhum poderá ser nulo.
  - Id: chave que identifica inequivocamente cada entrada do Historico usando um valor inteiro;
  - LocalInteresseVisitado_Id: valor inteiro que identifica na tabela LocalInteresseVisitado o local de interesse visitado a que este registo se refere.

Segue-se a descrição dos relacionamentos entre as diferentes entidades:

- KataLinguagem
  Dos atributos abaixo indicados nenhum poderá ser nulo.

  - Id: chave que identifica inequivocamente cada KataLinguagem usando um valor inteiro;
  - Kata_Id: valor inteiro que identifica na tabela Kata o Kata a que este relacionamento se refere;
  - Linguagem_Id: valor inteiro que identifica na tabela Linguagem a linguagem de programação a que este relacionamento se refere.

- KataProposto
  Dos atributos abaixo indicados nenhum poderá ser nulo.
  - Id: chave que identifica inequivocamente cada KataProposto usando um valor inteiro;
  - Validado: indica se o KataProposto já foi validado pelo administrador da aplicação, recorrendo a um valor booleano;
  - Utilizador_Id: valor inteiro que identifica na tabela Utilizador o utilizador a que este relacionamento se refere;
  - Kata_Id: valor inteiro que identifica na tabela Kata o Kata a que este relacionamento se refere.

- LocalInteresseProposto
  Dos atributos abaixo indicados nenhum poderá ser nulo.
  - Id: chave que identifica inequivocamente cada LocalInteresseProposto usando um valor inteiro;
  - Validado: indica se o LocalInteresseProposto já foi validado pelo administrador da aplicação, recorrendo a um valor booleano;
  - Utilizador_Id: valor inteiro que identifica na tabela Utilizador o utilizador a que este relacionamento se refere;
  - LocalInteresse_Id: valor inteiro que identifica na tabela LocalInteresse o local de interesse a que este relacionamento se refere.

- LocalInteresseVisitado
  Dos atributos abaixo indicados só "Resposta"e "Correto"podem ter valor nulo, uma vez que dependem da input do utilizador e do administrador, respectivamente. Todos os outros não.
  - Id: chave que identifica inequivocamente cada LocalInteresseVisitado usando um valor inteiro;
  - Data: valor do tipo data indicando o dia, mês e ano em que o local de interesse

foi visitado;

- Pontuacao: valor inteiro dos pontos a atribuir ao local consoante a sua distância entre a localização HOME do utilizador e a localização do LocalInteresseVisitado;
- Resposta: string de até 1000 caracteres indicando a resposta que o utilizador deu para o kata.
- Correto: indica se a resposta dada pelo utilizador está correta, recorrendo a um valor booleano;
- Utilizador_Id: valor inteiro que identifica na tabela Utilizador o utilizador a que este relacionamento se refere;
- LocalInteresse_Id: valor inteiro que identifica na tabela LocalInteresse o local de interesse a que este relacionamento se refere.

5 Esboço dos Interfaces do Sistema

Uma *User Interface (UI)* é definida através dos recursos de uma aplicação que permitirá ao utilizador interagir com o sistema. E o poder de uma *UI* devidamente aplicada é inegável. A interface é a "cara"do projeto, isto é, deverá permitir que o público-alvo veja claramente as opções ao seu dispor de modo a manter a atenção do utilizador. Ou seja, uma interface bem concebida, deverá facilitar as interações entre o utilizador e a aplicação.

É essencial elaborar propostas de interface antes do desenvolvimento do código, pois estes desenhos permitem concessionar, em conjunto com a modelação do projeto, algumas das funcionalidades que o sistema deverá suportar. Para além disso, é possível, através do ponto de vista do utilizador, saber a maneira mais eficaz de implementar o código.

Apresentam-se, seguidamente, os *mockups* para a aplicação *mobile* do trabalho a desenvolver.

Figura 5.1: *Mockups* do Sistema.

Nesta figura é possível ver do lado esquedo a página inicial da aplicação, solicitando o registo (*Sign up*) ou o início de sessão (*Sign in*) ao utilizador. Ao centro, é possível ver a página com o mapa, indicando a localização atual do utilizador, bem como vários locais de interesse possíveis de visitar. Finalmente do lado direito da figura, apresenta-se o aspeto da página de resolução de Katas, onde é possivel ao utilizador inserir o código com a sua solução do Kata e depois submeter.

6 Conclusões e Trabalho Futuro

Desde que se começou a abordar o assunto dos *Locais de Interesse*, a equipa de trabalho foi muito cuidadosa no que diz respeito ao contexto e às justificações para o desenvolvimento da aplicação. Assim, o primeiro passo realizado foi uma breve pesquisa de mercado. *Este tipo de software seria interessante? Esta aplicação seria atraente para o público-alvo em mente?* Essas primeiras perguntas foram de extrema importância, pois, se se tratasse de uma aplicação inútil, o trabalho a realizar teria sido em vão. Felizmente, através do contato direto com programadores em fase de formação, essa etapa foi relativamente simples.

A etapa seguinte levou o grupo a pensar nos requisitos fornecidos pelo "cliente"e a desenvolver um modelo do sistema. Entre outros aspetos, a base de dados foi um tema importante quando se abordou esta fase do projeto.

A gestão do software descrita neste trabalho, ou pelo menos os estágios iniciais do processo de gestão, foram considerados pelo grupo como uma oportunidade incrível de aprendizagem. Permitiu melhorar o conhecimento sobre vários temas, desde a recolha de requisitos até ao desenvolvimento da especificação de um sistema de software. Algumas destas tarefas já haviam sido realizadas noutros contextos, mas sempre de forma desligada. Aqui, foi possível experimentar todas em conjunto. Posto isto, o grupo acredita que foi bem sucedido na tarefa de elaborar um planeamento consistente, que permitirá um bom inicio à fase de implementação da aplicação *Code&GO*.

Em relação à aplicação propriamente dita, o grupo considera que esta primeira versão permitirá validar o conceito e mostrar a viabilidade real da aplicação. No futuro, existem várias melhorias a fazer, algumas das quais se referem de seguida:

- alteração da forma de validação das resoluções de katas propostas pelos utilizadores. Para tal terá de se prever um servidor dedicado a essa tarefa. Adicionalmente, terão se prever, para cada kata, conjuntos de testes suficientemente abrangentes, que permitam correr a resolução do utilizador no servidor e validar com os testes existentes

- adição de novos mapas, com vista a definir as versões Code&GO Braga, Code&GOPortugal, Code&GO Europe

- tal como se fez para o povoamento da base de dados de katas, é igualmente possível utilizar API de aplicações de turismo para alimentar a lista de locais de interesse. Na presente versão tal não se aplica, mas no futuro, e para os mapas de maior alcance, seria bastante útil usar esta estratégia

7 Referências

[1] The Lancet, A sporting chance: physical activity as part of everyday life, Lancet. 398 (2021) 365. https://doi.org/https://doi.org/10.1016/S0140-6736(21)01652-4.

[2] A.S.D. Moreira, Atividade física e comportamento sedentário dos sócios do GCP, (2019), Mestrado em Exercício e Saúde, Ramo Aprofundamento de Competências Profissionais, Universidade de Lisboa. https://www.repository.utl.pt/handle/10400.5/19913

[3] D. Johnson, S. Deterding, K.-A. Kuhn, A. Staneva, S. Stoyanov, L. Hides, Gamification for health and wellbeing: A systematic review of the literature, Internet Interv. 6 (2016) 89–106. https://doi.org/https://doi.org/10.1016/j.invent.2016.10.002.

[4] D. S. Patricio, Gamificação: Uma ferramenta para aumentar a frequência de atividade física em adolescentes com excesso de peso, (2017). Dissertação (Programa de Pós-Graduação em Saúde Pública - PPGSP) - Universidade Estadual da Paraíba, Campina Grande, 2017. http://dspace.sti.ufcg.edu.br:8080/jspui/handle/riufcg/16547

[5] P.R. da Silva, Discoverer: um aplicativo móvel para estímulo de atividades físicas com realidade aumentada, (2021). Mestrado de dupla diplomação com a UTFPR - Universidade Tecnológica Federal do Paraná e o Instituto Politécnico de Bragança. https://bibliotecadigital.ipb.pt/handle/10198/23296

[6] R. M. N. Policarpo, Melhoria de contacto com a natureza através da utilização de tecnologias móveis, (2016). Dissertação de mestrado. Lisboa: ISCTE-IUL. https://repositorio.iscte-iul.pt/handle/10071/12999

[7] Codewars: www.codewars.com

Lista de Siglas e Acrónimos

Para os efeitos do presente documento irão ser utilizados os seguintes Acrónimos e Siglas para a melhor compreensão do documento.

CnG Code&GO

BD Base de Dados

UML Unified Modelling Language

API Aplication Programming Interface

GPS Global Positioning System

IDE Integrated Development Environment

TIC Tecnologias da Informação e Comunicação

UI User Interface