Universidade do Minho
Escola de Engenharia
Licenciatura em Engenharia Informática
Mestrado Integrado em Engenharia Informática

Unidade Curricular de Laboratórios de Informática IV

Ano Letivo de 2021/2022

MinhoPark

Inês Vicente a93269
Jorge Melo a93308
Marco Costa a93283
Miguel Martins a93280
Tomás Francisco a93193

29 de novembro de 2021

LI4

Data de Receção	
Responsável	
Avalição	
Observações	

MinhoPark

Inês Vicente a93269
Jorge Melo a93308
Marco Costa a93283
Miguel Martins a93280
Tomás Francisco a93193

29 de novembro de 2021

Resumo

Apresentamos *MinhoPark*, uma app mobile que tem como objetivo principal divulgar espaços verdes na região do Minho. É uma app com funcionalidades simples, e que vão diretas ao assunto: encontrar parques.

Este projeto foi desenvolvido prestando atenção áquilo que uma pessoa, de facto, daria importância numa aplicação deste género. Sendo assim, ela permite ao utilizador pesquisar por parques perto de si, ou de algum sítio na região do Minho à escolha, tendo em conta preferências especificadas por ele. Esses parques são apresentados numa lista e num mapa, podendo facilmente selecionar um deles para ver mais informações e percursos para lá chegar. Também é possível adicionar os parques a uma lista de favoritos, permitindo assim ao utilizador guardar parques que gostou de visitar, ou que tem curiosidade de visitar um dia.

Com isto, podemos dizer que esta aplicação dá bastante liberdade ao utilizador, sem fazer com que isso comprometa a facilidade de utilização. Consideramos esta aplicação algo útil com valor de mercado.

Área de Aplicação: Engenharia de Software, Programação mobile, Desenho e arquitectura de Sistemas de Bases de Dados.

Palavras-Chave: Bases de Dados Relacionais, Planeamento de Software, Desenvolvimento Backend, Desenvolvimento Frontend.

Índice

Lista de Figuras

Lista de Tabelas

1 Introdução

1.1 Contextualização

Atualmente tem existido uma aposta crescente, especialmente por parte dos municípios, nas áreas verdes e, com isto, os parques municipais e urbanos têm ganho um grande destaque no planeamento geográfico das cidades portuguesas. Estes parques, além de funcionarem como uma espécie de "casulo civilizacional", isolados, em certa medida, da maior parte das características da urbanização, também têm a vantagem de serem de acesso gratuito e serem local de uma grande panóplia de atividades.

Contudo, normalmente, este tipo de locais não são um ponto de referência para turistas ou até mesmo habitantes da região, uma vez que não são devidamente divulgados pelos Municípios. Ora, tendo em conta que, em muitos casos, um parque municipal não se limita a ser somente uma "área verde" fechada e livre de edificios, com abundante presença de vegetação, mas também um local de prática de atividades desportivas, convívio, comércio, divulgação cultural, entre outros, é necessário ter em conta que existiu um investimento económico considerável para a construção de infraestruturas que suportem estas atividades. Se compararmos esse investimento com a utilização que é muitas vezes dada a estes parques, é visível um claro subaproveitamento do potencial dos mesmos, devido, em grande parte, à falta de divulgação, que poderia ser aumentada com uma aplicação que fosse específica para parques.

De igual modo, existem outros tipos de parques, como parques de diversões, de merendas ou até mesmo parques naturais que poderiam ser utilizados com mais frequência se existisse uma aplicação que fornecesse ao utilizador informação genérica sobre os mesmos, podendo-o orientar na deslocação para o mesmo, caso o utilizador pretendesse.

1.2 Motivação e objetivos

Primeiramente, o grupo pretendia conceber um sistema que conseguisse divulgar uma área de turismo que, apesar de atual, indo de encontro a ideais ecológicos, ainda nos parece bastante subaproveitado. Assim sendo, inicialmente concebemos um sistema focado somente nos Parques Urbanos. Contudo, rapidamente nos apercebemos que existiam outros tipos de parques que, não sendo "áreas verdes"assumidas, são também eles infraestruturas, que provavelmente tiveram custos consideráveis para os municípios e/ou empresas privadas que os gerem, e que também são muito comumente subaproveitados.

Consideramos que haver uma aplicação que torne a descoberta de parques num processo simples e rápido vai fazer com que estes tenham, de facto, mais visibilidade.

Dito isto, a motivação principal do nosso sistema é divulgar os parques de uma dada região, não colocando de lado a motivação inicial de divulgação de "espaços verdes". Contudo, esta é uma tarefa que necessita de recursos, e apesar de não o parecer, a conceção de sistemas de *Software* tem custos consideráveis, necessitando, portanto, de uma entidade que, além de ajudar financeiramente a equipa que concebesse este sistema, também teria de fazer uma divulgação persistente no seio da população. Dado que a literacia informática dos municípios ainda não é a mais desenvolvida, este seria um projeto interessante para os mesmos financiarem com a ajuda de empresas que dele poderiam tirar proveito (nomeadamente as que gerem parques de diversões e de campismo). Além disso, poderiam também aproveitar a sua maior visibilidade e divulgar o projeto. Analisando o panorama geopolítico do país, o grupo acabou por limitar geograficamente a mesma à região histórica do Minho, uma vez que esta é uma região que além de ser conhecida pelas tradições e "áreas verdes", também tem vindo nos últimos anos a desenvolver projetos no setor das Tecnologias Informáticas. Este projeto teria, portanto, de resultar de uma colaboração dos diferentes municípios da região.

Além da divulgação de parques da região do Minho, implícita no nome MinhoPark, o sistema deverá fazê-lo de forma a apresentar uma lista limitada dos parques próximos de determinada localização, podendo esta ser a localização atual do utilizador ou outra que o mesmo defina, ser capaz de apresentar um conjunto de informações úteis sobre um parque específico, permitir ao utilizador adicionar e retirar parques de uma lista de favoritos e orientar o utilizador na sua deslocação para um parque.

1.3 Justificação e utilidade do sistema

Como já foi referido, o sistema foi concebido de forma a poder divulgar com maior eficácia um conjunto de infraestruturas que muito comummente são subaproveitadas. Além disso, a sua utilização permitirá expandir um tipo de turismo, o turismo verde, numa região portuguesa que é bastante conhecida pelas suas áreas verdes. Contudo, não se põem de lado outras potencialidades interessantes do mesmo, que podem ser expandidos a todos os tipos de parques, desde parques de diversões até parques de campismo.

1.4 Estabelecimento da identidade do projeto

Achamos que o nosso projeto tem valor na medida em que pode colmatar o problema que referimos anteriormente, haver falta de divulgação de espaços verdes. Notámos que há um espaço de mercado vazio nessa área e que pode ser aproveitado, daí a vantagem desta aplicação.

O projeto deve ser idealizado tendo em mente o seu potencial e parece evidente que limitá-lo a um universo meramente académico pode levar ao subaproveitamento de uma ideia interessante. Deste modo, a equipa responsável pela sua implementação, deve ter em conta que apesar de estar a desenvolver um projeto para a unidade curricular de Laboratórios de Informática IV, não deve ser descuidada, tentando ao máximo simular o processo de comunicação com os municípios minhotos, como já se deu a entender previamente.

Tendo o ponto anterior em mente, a equipa de trabalho deve não só ser competente na elaboração de um projeto académico que a equipa docente considere aceitável como também deve observar os elementos do corpo docente como clientes (no caso entidades representativas dos municípios), devendo para tal manter frequentemente a comunicação com os mesmos, averiguando se o trabalho que têm feito e o que planeiam fazer está de acordo com os desígnios dos mesmos.

Nesse sentido, os desenvolvedores da aplicação ***MinhoPark*** devem ter como primeira prioridade o cumprimento dos prazos estabelecidos pelos seus clientes.

1.5 Identificação dos recursos necessários

Para o desenvolvimento do projeto é possível dividir os recursos necessários em duas vertentes: a vertente humana, os funcionários necessários para o projeto ser desenvolvido e a vertente tecnológica, as ferramentas de *Software* utilizadas.

1.5.1 Vertente humana

No que ao capítulo dos funcionários diz respeito, serão necessários 5, agrupados em 3 grupos, que comunicando entre eles, conseguem conceber a aplicação ***MinhoPark*** desde raiz. Será necessário **1 *Project Manager***. Este é o responsável por gerir todo o projeto, desde a comunicação com o cliente, passando pelo levantamento e análise de requisitos e frequente comunicação com o Engenheiro de *Software* e modelação que o mesmo desenvolve, até ao controlo do produto final que os programadores têm vindo a desenvolver. É em certo modo uma figura omnipresente em todo o projeto e o facto de comunicar diretamente com o cliente, permite-lhe ter uma visão bastante clara do que o mesmo pretende, sendo, por isso, essencial para a criação de *Mockups*, que dão uma ideia geral do que será a aplicação a nível mais superficial.

Será também essencial possuir **1 Engenheiro de *Software***. Este engenheiro possui formação suficiente para perceber como pode modelar a aplicação de forma a ir de encontro aos interesses do *Project Manager* e da empresa, não devendo descartar como o fazer de acordo a poder imaginar um bom produto final para o cliente. Este funcionário, deve, portanto, dominar todos os aspetos técnicos da Modelação, especificar que ferramentas de *Software* são mais adequadas qualitativamente e financeiramente para essa modelação e para os programadores utilizarem. Este funcionário, não está, contudo, impedido de ter de assumir funções de programador, uma vez que também é uma área que o mesmo deve dominar e assumir caso o projeto se encontre nalgum estado que o mesmo considere "crítico".

Por último o projeto deverá ter **3 Programadores:** Cada um deles será responsável por uma das camadas especificada mais à frente na Maqueta do Sistema. São estes 3 funcionários os responsáveis pela implementação final da aplicação, não deixando, por isso, de manter a comunicação com o *Project Manager* e com o Engenheiro de *Software*, averiguando frequentemente se estão a trabalhar de acordo com o que lhes foi solicitado.

1.5.2 Vertente tecnológica

Seguidamente, estão enumeradas algumas categorias, que podem ser consideradas fundamentais para conseguir desenvolver a aplicação ***MinhoPark*** com sucesso. Note-se que são somente fornecidas categorias, de modo a poder dar alguma liberdade às ferramentas efetivamente utilizadas. Contudo, são mesmo assim referidos alguns exemplos de forma a facilitar o processo de escolha dessas ferramentas.

Primeiramente, tendo em conta a importância da comunicação quer com cliente, quer entre os diferentes funcionários, é essencial utilizar um serviço de comunicação remota. Estes não só se revelam bastante práticos por evitar deslocações - por vezes desnecessárias - para reuniões curtas, também permitem uma fácil comunicação entre indivíduos que fisicamente se encontram bastante afastados. Além disso, vão de encontro a uma nova tendência, o trabalho remoto, bem como diminuir os possíveis contactos de risco em tempos de pandemia. São

alguns exemplos deste tipo de serviços o *Zoom*, o *Google Meet*, o *Discord* ou o *Microsoft Teams*.

Outro aspeto que já se referiu ter um peso considerável no desenrolar do projeto é o aspeto da modelação. Os vários digramas que se podem desenvolver através das técnicas de modelação, permitem ter uma visão bastante ilustrativa do aspeto e comportamento do programa. Neste caso, é inevitável aconselhar o uso da linguagem de modelação UML para o desenvolvimento desses diagramas. O *software Visual Paradigm* é bastante útil para a utilização da linguagem UML, mas existem outras alternativas como o *Draw.io*, o *Visio* ou o *Sketch*.

Por último, existe um conjunto de ferramentas que são muitas vezes colocadas como prioridade no processo de escolha, apesar das referidas anteriormente serem mais importantes. Estas são a linguagem de programação, recomendando-se que neste projeto específico se evite linguagens mais *low level*. São bons exemplos de linguagens para desenvolvimento de aplicações Android: o *Java* ou *Kotlin*. De igual modo, é também pertinente discutir que sistema de controlo de versões deve ser usado, uma vez que o projeto tem uma dimensão em que é fortemente recomendado que haja divisão de tarefas e trabalho em separado por parte dos programadores. Além disso, possuir um histórico de edições do repositório também pode revelar-se bastante útil no caso de alguma coisa correr mal. O *Git* é fortemente recomendado. É ainda recomendada a utilização de uma API e a conceção de uma Base de Dados, que possa auxiliar os programadores a facilmente desenvolver métodos de pesquisa na camada da Lógica de Dados. Esta camada será melhor especificada na Maqueta do Sistema e poderá tirar proveito de APIs como a do *Google Maps* ou da *Bing Maps* e de sistemas para gerir a Base de Dados, como o *SQL Server* ou *MongoDB*.

1.6 Maqueta do sistema

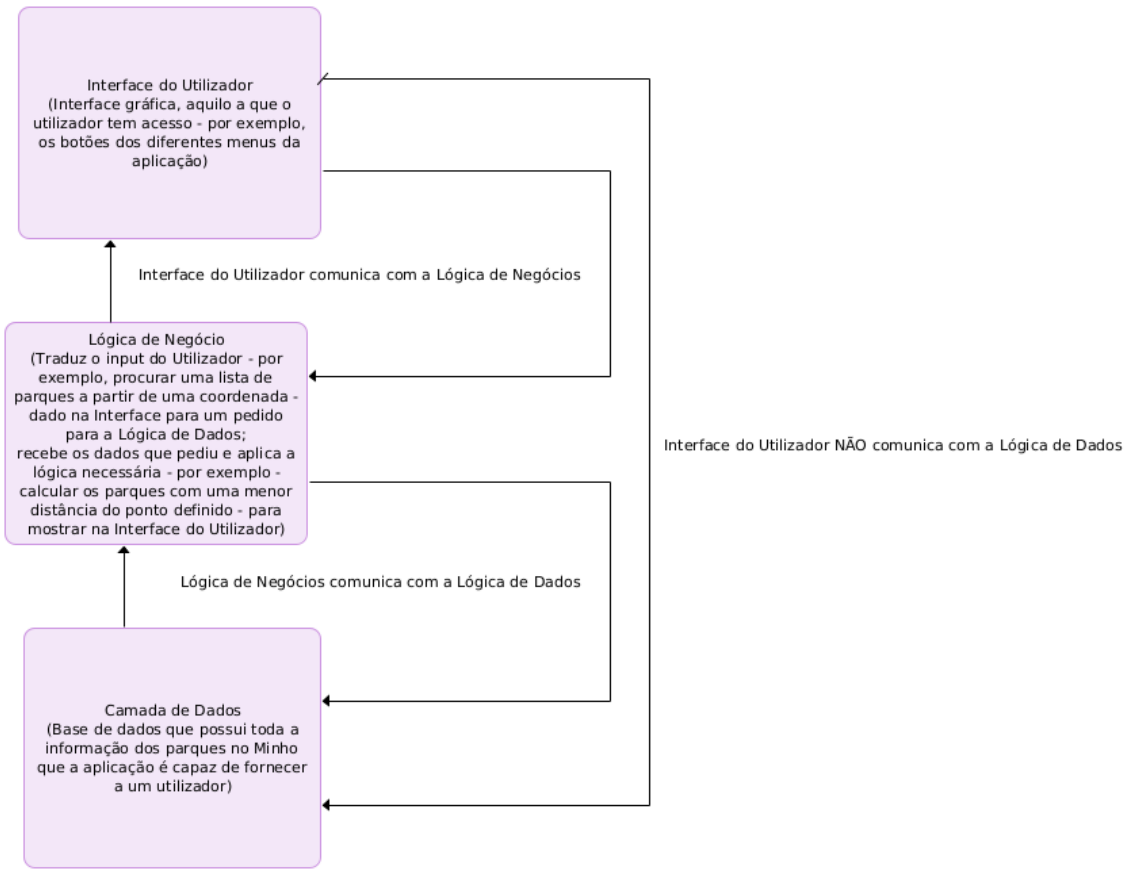


Figura 1.1: Maquete do Sistema

Este projeto segue uma arquitetura semelhante à de MVC chamada Model Delegate, que encapsula a interface da lógica de negócios e estas da camada de dados. Isto permite mudar uma destas componentes sem quebrar o restante código. Para além disso, esta liberdade cria a possíbilidade de ter, por exemplo, várias interfaces ou várias camadas de dados para o mesmo programa, desde que estas implementem as interfaces de ligação entre camadas.

1.7 Definições de um conjunto de medidas de sucesso

Uma grande parte do projeto que visa a criação da aplicação ***MinhoPark*** será financiada e publicitada pelos munícipios da região, gastando-se para tal recursos do Estado, que entram nas contas públicas e podem causar alguma polémica no caso de se lhes dar um mau uso. Assim sendo, foi definido um conjunto de objetivos de forma a poder avaliar o sucesso do projeto:

- O desenvolvimento do sistema de *Software* ***MinhoPark***, desde a sua fase inicial de modelação e especificação até à implementação não deverá ter um custo que exceda os valores estipulados pelas entidades financiadoras (€29047.3);
  Nota: O valor calculado representa 4% das despesas efetivas das Câmaras municipais do Minho no ano de 2019

- A aplicação deve permitir ao utilizador ter uma ideia concreta sobre um parque específico, não lhe apresentado informação inútil , nem uma quantidade demasiado sucinta de informação;

- O sistema deve ser capaz de processar todas as pesquisas, tendo mensagens de erro e excessão elucidativas para o utilizador;

- O sistema deve ser capaz de orientar o utilizador na sua deslocação para um parque, em qualquer dos meios de transporte;

- Qualquer utilizador, mesmo os que tenham menor literacia informática, devem utilizar facilmente a aplicação ***MinhoPark***, por esta ser intuitiva;

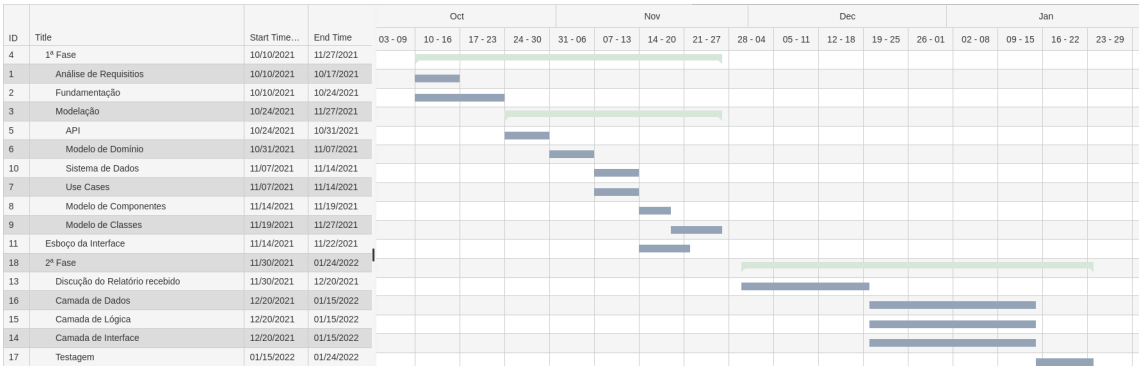
- Os municípios devem constatar um aumento de 20% na frequência dos parques;

- A manutenção do *Software* deve ser feita frequentemente através de atualizações, resolvendo de forma eficaz qualquer tipo de *bug* que seja encontrado;

- Indo de encontro ao primeiro ponto, as entidades financiadoras estipularam o prazo de entrega do sistema finalizado para o dia 24/01/2022. O sistema deve, portanto, estar finalizado e ser entregue nesse dia.

1.8 Plano de desenvolvimento (diagrama GANTT)

De forma a concluirmos o projeto num prazo aceitável e para nos coordenarmos melhor, fizemos um diagrama GANTT em que dividimos o trabalho em fases definindo para cada uma um prazo. Apesar de não ser perfeito, e de em certas fases nos termos atrasado e em outras adiantado, este serve para nos dar uma ideia bruta do tempo e trabalho necessários para a realização de cada fase e do projeto como um todo.



ID	Title	Start Time…	End Time
4	1ª Fase	10/10/2021	11/27/2021
1	Análise de Requisitios	10/10/2021	10/17/2021
2	Fundamentação	10/10/2021	10/24/2021
3	Modelação	10/24/2021	11/27/2021
5	API	10/24/2021	10/31/2021
6	Modelo de Domínio	10/31/2021	11/07/2021
10	Sistema de Dados	11/07/2021	11/14/2021
7	Use Cases	11/07/2021	11/14/2021
8	Modelo de Componentes	11/14/2021	11/19/2021
9	Modelo de Classes	11/19/2021	11/27/2021
11	Esboço da Interface	11/14/2021	11/22/2021
18	2ª Fase	11/30/2021	01/24/2022
13	Discução do Relatório recebido	11/30/2021	12/20/2021
16	Camada de Dados	12/20/2021	01/15/2022
15	Camada de Lógica	12/20/2021	01/15/2022
14	Camada de Interface	12/20/2021	01/15/2022
17	Testagem	01/15/2022	01/24/2022

Figura 1.2: Diagrama GANTT

2 Levantamento e Análise de Requisitos

2.1 Apresentação da estratégia e método

Para a análise de requisitos, foi utilizada uma pesquisa elaborada sobre os vários campos que a aplicação teria de satisfazer, não só para os utilizadores mas também para os *stakeholders*. Estes campos, depois, tornaram-se requerimentos, condições obrigatórias que a aplicação teria de concretizar para se poder considerar um sucesso.

2.2 Descrição geral dos requisitos (funcionais e não funcionais) levantados

2.2.1 Funcionais

- O utilizador inicia a aplicação:
  - O sistema mostra o menu principal ao utilizador, oferecendo opções para definir as preferências ou para escolher de uma lista de favoritos do utilizador.
- O utilizador define as suas preferências:
  - É mostrada uma secção de localização, onde pode ser escolhida a localização atual do dispositivo como ponto de início ou uma outra localização.
  - O sistema mostra uma lista de tipos que definem um parque como características deste.
  - São mostradas três opções sobre as quantidades de parques que o utilizador quer demonstrados na procura, os top 5 parques, os top 10 ou os top 15.
  - É mostrada uma lista de parques que correspondem aos parâmetros definidos anteriormente pelo utilizador.

- O utilizador pede ao sistema a lista de preferidos:
  - O sistema mostra uma lista dos parques que o utilizador marcou como favoritos.

2.2.2 Não funcionais

- O programa deve ser rápido o suficiente para executar uma procura por parâmetros.
- O programa deve ser seguro para todos os utilizadores, sem compremeter alguma espécie de dados.
- O programa deve ser possível de utilizar sem erros ou *bugs* frequentes.
- Devem exister requerimentos operacionais que definem como o sistema deve ser usado.
- Uma linguagem de programação deve ser específica pelos requerimentos operacionais.
- Os recursos no ambiente de desenvolvimento devem ser sempre utilizados se forem mencionados como necessários para o funcionamento do sistema.
- O ambiente de operações do sistema deve ser delimitado por certos parâmetros.
- O sistema deve possuir uma lista de requerimentos que determina se o sistema será aprovado por um regulador, um agente que verifica as várias condições do sistema.
- O sistema deve sempre, sem exceção, operar dentro dos parâmetros legais das suas funcionalidades, nunca quebrando nenhuma lei de forma alguma.
- O sistema deve ser aceitável para uso do público geral e outros potenciais utilizadores, mantendo sempre um carácter ético.

2.3 Validação dos requisitos estabelecidos

- As várias partes interessadas no projeto (investidores e utilizadores) devem ser igualmente atendidas pelo sistema, tendo em conta potenciais compromissos.
- Os requerimentos do sistema não podem estar em conflito, ou seja, não podem haver limitações contraditórias ou descrições diferentes para uma mesma funcionalidade do sistema.
- O documento dos requerimentos deve conter dentro de si todas as funções do sistema que são utilizadas e quaisquer limitações impostas por um utilizador do sistema.

- Tendo em conta o tempo do projeto e o orçamento, deve ser previsto se o sistema consegue ser implementado usando tecnologias atuais, sem exceder o tempo limite e sem exceder o orçamento.

- Os requerimentos do sistema devem sempre ser escritos de forma a poderem ser verificados. Esta verificação é feita através de testes que provam que o sistema cumpre cada requisito. Os testes podem ser os seguintes:

  - Análises de requerimentos para garantir que cada requerimento se encontra sem erros ou inconsistências.

  - A criação de um protótipo que os utilizadores e/ou outros clientes podem utilizar de forma a descobrir se satisfaz as suas necessidades.

  - Tornar todos os requerimentos testáveis de forma a descobrir se um programa é viável, em termos da facilidade da sua implementação.

3 Especificação e Modelação do Software

3.1 Apresentação geral da especificação

De forma a criarmos um programa que cumpra todos os requisitos propostos, decidimos fazer uma modelação simples que servirá de base para ser expandida funcionalmente no futuro. Assim o programa evita *bugs* maiores e não confunde o utilizador com um labirinto de menus e opções que nunca utilizará. Com isto, e de forma a evitar quaisquer fugas de informação sensivel do utilizador, tomamos a decisão de que todos os dados deste devem ser guardados localmente no seu dispostivo.

3.2 Cenários

Uma vez que estamos habituados, nas aulas de DSS, a conceber os diferentes elementos de modelação tendo por base um conjunto de cenários, optamos por colocar neste relatório os cenários que criámos para conceber toda a modelação que se seguirá. Deste modo, tanto os docentes como também o grupo que receber a nossa especificação, poderá ter uma visão mais crítica sobre a mesma, tendo por base os cenários que se seguem. Além disso, estes também oferecem um visão abstrata do *layout* da aplicação, que ficará mais claro no capítulo 5, onde apresentamos os *mockups* da mesma.

- **Cenário 1:** O utilizador procura uma lista de parques com as preferências previamente especificadas. Para isso, ele carrega no botão "PROCURA", no menu principal. É-lhe mostrada a lista de parques, assim como um mapa com todos esses parques apontados. O utilizador seleciona um dos parques para ver mais informações e a imagem deste. Depois de ver todas as informações que pretende, ele seleciona o botão "VOLTAR", abaixo da página, e retorna ao menu principal.

- **Cenário 2:** O utilizador adiciona/retira um parque à lista de favoritos. Para isso, aquando da procura, ele seleciona o botão estrela na foto. Se o parque não estiver na lista (indicado com a estrela transparente), o sistema adiciona-o, colocando a estrela amarela. Se o parque estiver na lista (indicado com a cor amarela da estrela) o sistema

retira-o, fazendo com que a estrela retorne ao seu estado transparente.

- **Cenário 3:** O utilizador quer visualizar a sua lista de favoritos. Para isso, ele seleciona "FAVORITOS"no menu principal. O sistema mostra uma lista com os parques que já foram adicionados aos favoritos. O utilizador pode retirar algo dos favoritos, clicando na estrela. Depois de obter todas as informações que pretende, ele seleciona o botão "VOLTAR", abaixo da página, e retorna ao menu principal.

- **Cenário 4:** O utilizador quer alterar as preferências de procura. Para isso, ele seleciona "PREFERÊNCIAS"no menu principal. Depois, ele pode escolher se pretende utilizar a sua localização atual, ou inserir outras coordenadas. Também pode selecionar diferentes tipos de parques a serem mostrados. Depois de ver todas as informações que pretende, ele seleciona o botão "VOLTAR", abaixo da página, e retorna ao menu principal.

- **Cenário 5:** Depois de ter escolhido um parque para visitar, o utilizador seleciona "MOSTRAR DIREÇÕES"no menu do parque selecionado. O sistema mostra então no mapa as direções desde a posição do utilizador até ao parque.

3.3 Aspetos estruturais

3.3.1 Modelo de Domínio

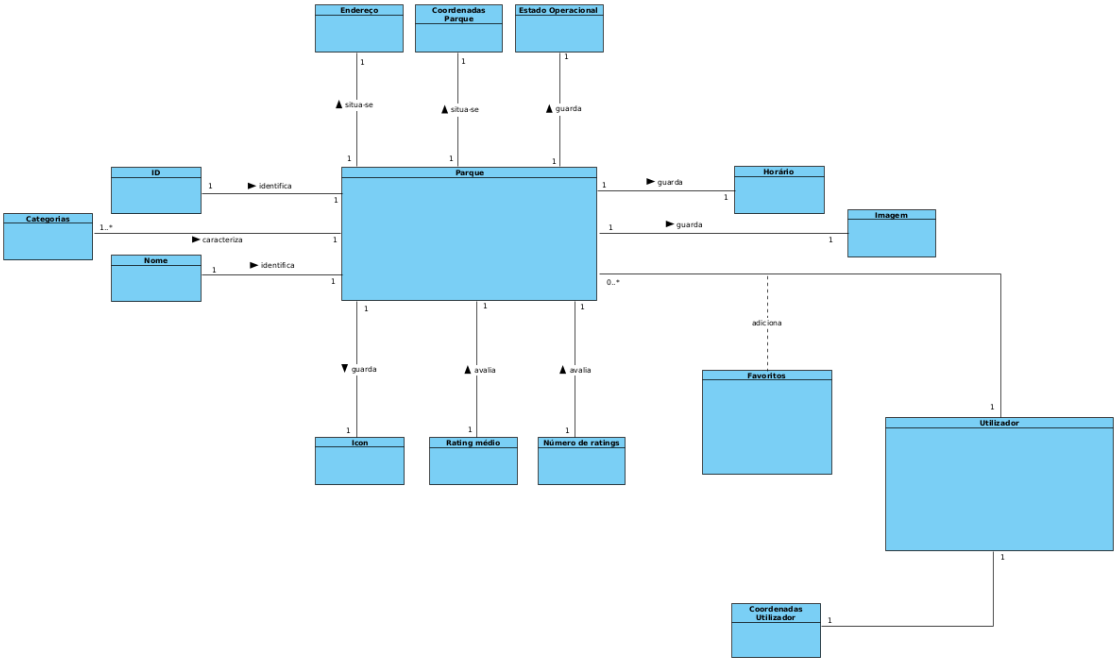


Figura 3.1: Diagrama Modelo de Domínio

A parte central do programa são obviamente os parques. Estes são descritos pelas várias entidades representadas na imagem acima. Para além das entidades que serão guardadas na base de dados o parque possui um estado operacional (se está aberto ou fechado) que deve ser calculado no momento da pesquisa deste.

Cada parque é definido por uma ou mais categorias. Estas categorias têm um *icon* associado, que é o que aparece no mapa.

Relativamente ao utilizador, apesar desta aplicação não possuir um sistema de autenticação, este existe e possui uma lista de favoritos que lhe permite encontrar os parques que mais frequenta com facilidade. Mais ainda, o sistema deve calcular as coordenadas do utilizador para lhe recomendar parques próximos e para lhe indicar um percurso até ao parque que escolheu.

3.3.2 Modelo de Componentes

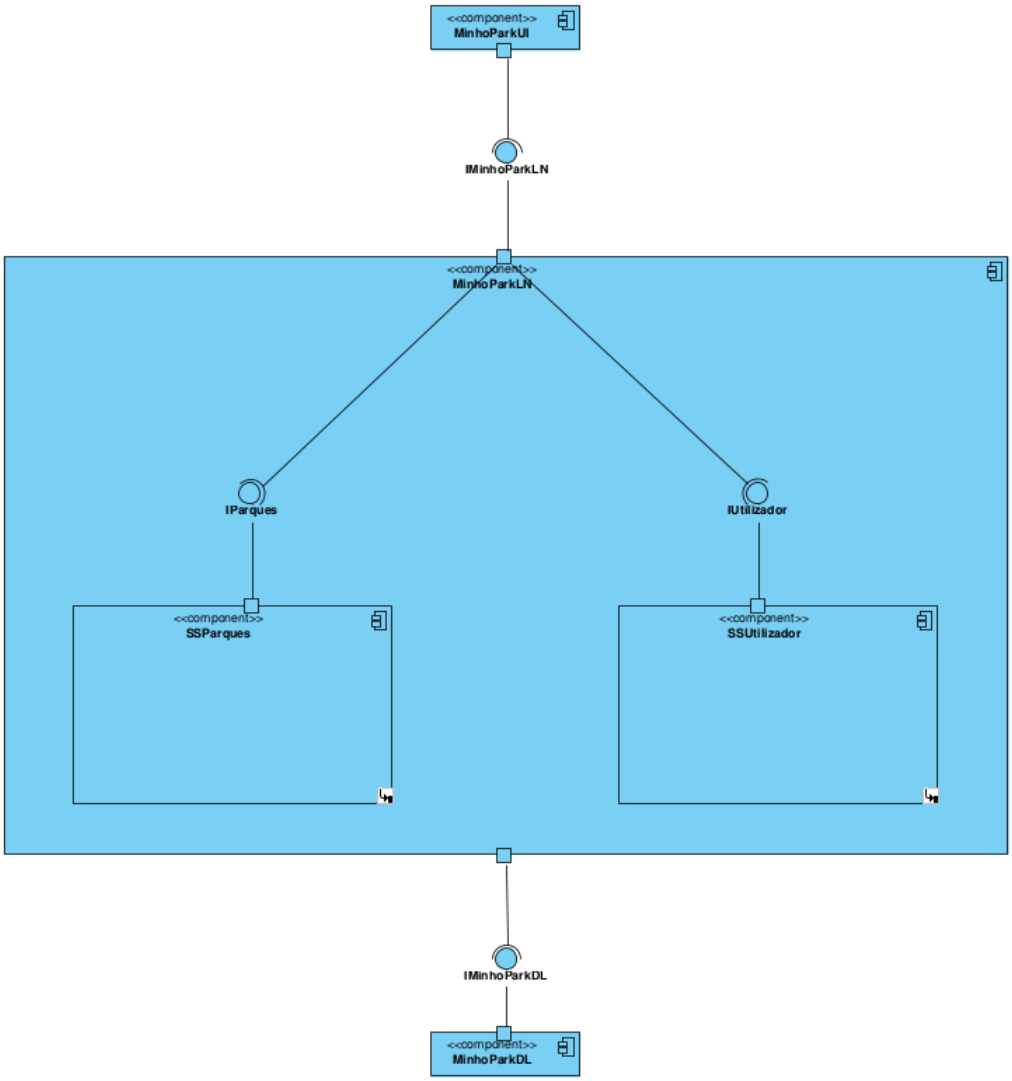


Figura 3.2: Diagrama Modelo de Componentes

A camada lógica do nosso programa foi dividida em dois subcomponentes, um para parques e outro para utilizadores. Esta separação permite uma maior modularidade na estrutura do programa o que facilita a sua extensão no futuro, para além de reduzir o famoso código "esparguete".

3.3.3 Modelos de Classes

SSParques

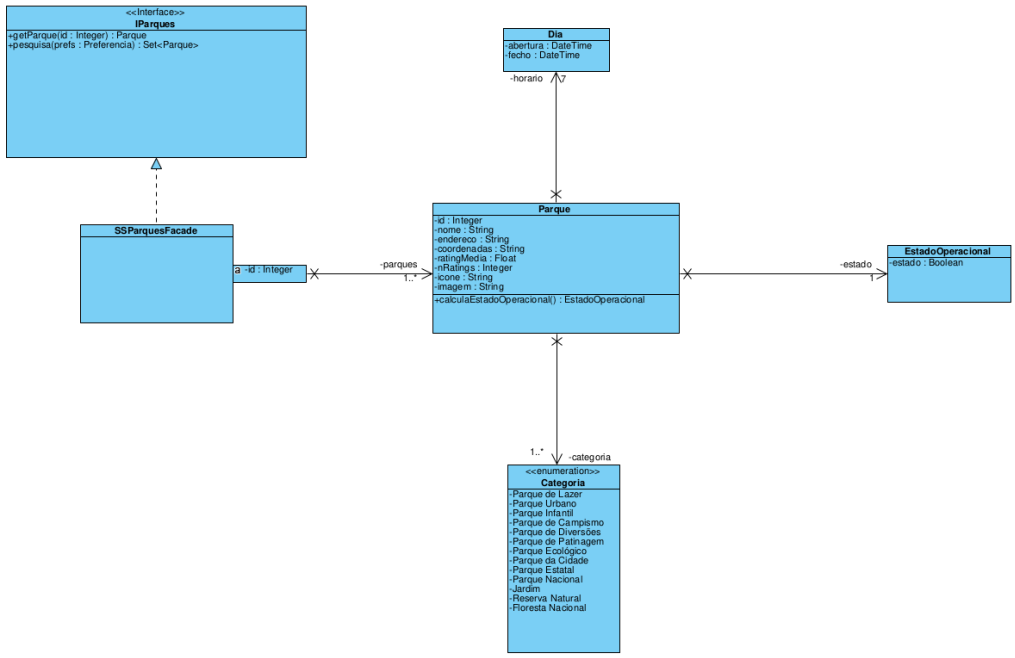


Figura 3.3: Diagrama Modelo de Classes do SSParque

Visto que esta componente é apenas de leitura existem dois métodos disponíveis ao resto do programa:

- **getParque(id:Integer):** Este método deve devolver o parque correspondente ao número de identificação.
- **pesquisa(prefs:Preferencia):** Recebe as preferências do utilizador e retorna uma lista de parques relevantes.

Nota: Ambos os métodos devem calcular o estado operacional dos parques antes de os retornarem.

SSUilizadores

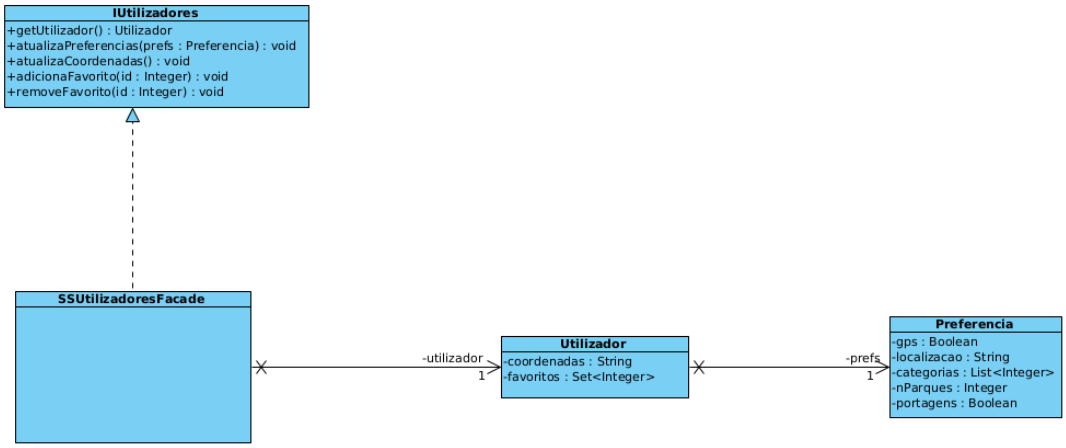


Figura 3.4: Diagrama Modelo de Classes do SSUtilizador

O subsistema de utilizadores é bastante simples, contendo apenas o *facade*, o utilizador e as preferências. Porém decidimos mesmo assim separa-lo das outras componentes para permitir uma fácil expansibilidade das funcionalidades desta vertente da aplicação.

Os métodos que a *facade* disponibiliza ao restantes sistemas são os seguintes:

- **getUtilizador():** Este método apenas retorna uma cópia do utilizador.
- **atualizaPreferencias(prefs:Preferencia):** Atualiza as preferências do utilizador com as novas preferências.
- **atualizaCoordenadas():** Utiliza as capacidades de GPS do dispositivo para obter as suas coordenadas e atualizar a posição do utilizador.
- **adicionaFavorito(id:Integer):** Adiciona o id do parque à lista de favoritos.
- **removeFavorito(id:Integer):** Remove o id do parque da lista de favoritos caso este exista.

Nota: Os dados devem ser gravados para o disco sempre que são alterados. Para além disso o método *getUtilizador* deve obter o utilizador do disco. Ambas leitura/escrita devem ser feitas através de *DAO (Data Access Objects)*.

3.4 Aspetos comportamentais

3.4.1 Modelo de *Use Cases*

Diagrama de *Use Cases*

A aplicação *MinhoPark* será algo bastante simples. Esta não necessitará de tratar qualquer tipo de comunicação entre mais que 1 ator, nem terá um sistema de autenticação, visto que na nossa opinião não faz sentido para o tipo de funcionalidades que irá oferecer. Deste modo, o diagrama representado na Figura 3.5, representa todos os *Use Cases* que consideramos serem necessários para o único ator (o Utilizador) comunicar com o Sistema.

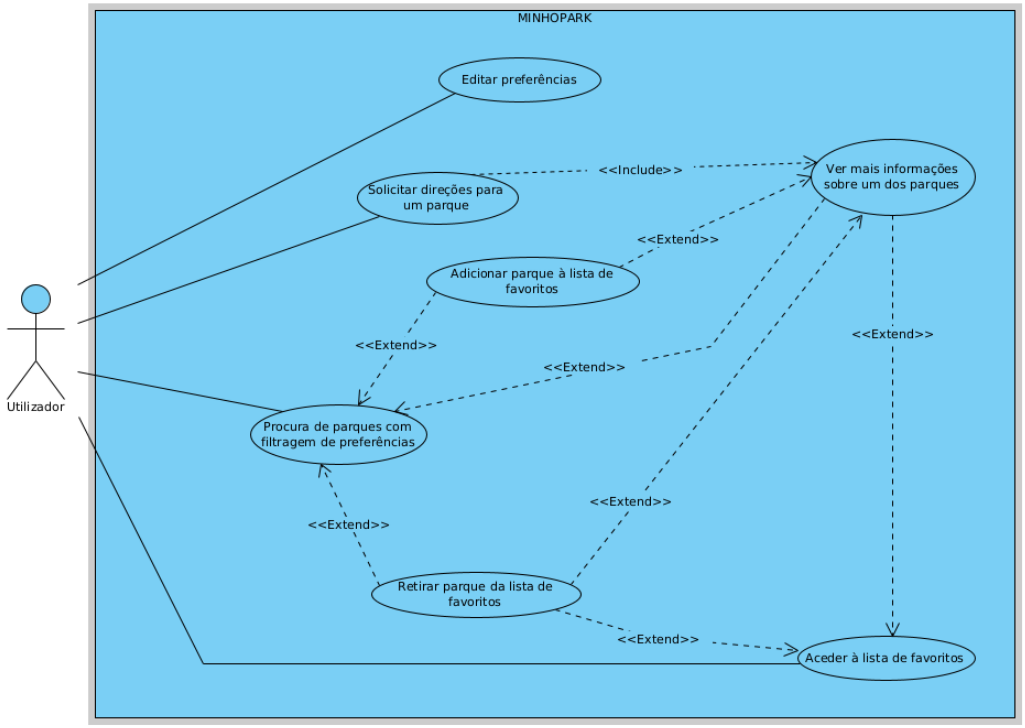


Figura 3.5: Diagrama de *Use Cases*

Especificações dos *Use Cases*

Seguidamente podemos observar as especificações de cada um dos *Use Cases* considerados no diagrama da Figura 3.5. Foram estes que constituiram a base para a definição de APIs, que permitiram a conceção dos restantes elementos da Modelação do sistema.

Use Case		
Use case:	Editar preferências	
Ator:	Utilizador	
Descrição:	O utilizador altera as preferências de pesquisa	
Cenários:	4	
Pré-condição:	True	
Pós-condição:	As preferências são atualizadas.	
	Utilizador	Sistema
Cenário normal	2. O utilizador edita as preferências (localização, tipo de parque, extensão da lista e opção de portagens).	1. O sistema mostra a página de edição. 3. O sistema atualiza as preferências.
Cenário de Exceção (1) [Utilizador insere coordenadas inválidas.] (passo 2)		2.1. O sistema informa o utilizador que as coordenadas são inválidas.

Tabela 3.1: *Use Case* Editar preferências

Use Case		
Use case:	Solicitar direções para um parque	
Ator:	Utilizador	
Descrição:	O sistema mostra ao utilizador um percurso até ao parque	
Cenários:	5	
Pré-condição:	O utilizador está na página de informações do parque	
Pós-condição:	O sistema mostra o percurso até ao parque, podendo orientar o utilizador na sua deslocação	
	Utilizador	Sistema
Cenário normal	1. «include» Mostrar mais informações sobre um dos parques. 5. O utilizador escolhe a opção que pretende utilizar.	1. «include» Mostrar mais informações sobre um dos parques. 2. O sistema obtém as coordenadas do utilizador. 3. O sistema calcula os vários percursos até ao parque para os diferentes meios de deslocação. 4. O sistema mostra as diferentes opções de deslocação ao utilizador (A PÉ/DE CARRO/DE TRANSPORTES PÚBLICOS) num mapa interativo onde o utilizador pode observar as diferenças entre os diferentes percursos. 6. O sistema mostra o percurso no mapa, atualizando-o à medida que o utilizador se desloca.
Cenário de Exceção (1) [Utilizador não escolhe nenhuma das opções e carrega no botão "VOLTAR".] (passo 5)		5.1. O sistema volta ao menu das informações sobre o parque.

Tabela 3.2: *Use Case* Solicitar direções para um parque

Use Case		
Use case:	Ver mais informações sobre um dos parques	
Ator:	Utilizador	
Descrição:	O utilizador obtém mais informações sobre um parque específico	
Cenários:	1	
Pré-condição:	Existe lista que responda às preferências do utilizador	
Pós-condição:	O utilizador obteve informação sobre um dos parques	
	Utilizador	Sistema
Cenário normal	1. O utilizador escolhe da lista o parque sobre o qual pretende obter mais informações	2. O sistema mostra as informações (nome, horário, 1 imagem, média e número de *ratings*, categorias de parque e endereço).

Tabela 3.3: *Use Case* Ver mais informações sobre um dos parques

Use Case		
Use case:	Adicionar parque à lista de favoritos	
Ator:	Utilizador	
Descrição:	Permite ao utilizador de guardar um parque na lista de favoritos	
Cenários:	2	
Pré-condição:	Parque a adicionar não está na lista de favoritos	
Pós-condição:	Existe um novo parque na lista de favoritos	
	Utilizador	Sistema
Cenário normal		1. O sistema verifica que o parque não pertence à lista de favoritos. 2. O sistema guarda o parque na lista de favoritos.

Tabela 3.4: *Use Case* Adicionar parque à lista de favoritos

Use Case		
Use case:	Retirar parque da lista de favoritos	
Ator:	Utilizador	
Descrição:	Permite ao utilizador remover o parque da lista de favoritos	
Cenários:	3	
Pré-condição:	O parque a remover está na lista de favoritos	
Pós-condição:	O parque não está na lista de favoritos	
	Utilizador	Sistema
Cenário normal		1. O sistema remove o parque da lista de favoritos

Tabela 3.5: *Use Case* Retirar parque da lista de favoritos

Use Case		
Use case:	Aceder à lista de favoritos	
Ator:	Utilizador	
Descrição:	Permite ao utilizador aceder à lista de favoritos	
Cenários:	3	
Pré-condição:	True	
Pós-condição:	O sistema mostra a lista de favoritos	
	Utilizador	Sistema
Cenário normal		1. O sistema mostra a lista de favoritos.

Tabela 3.6: *Use Case* Aceder à lista de favoritos

Use Case		
Use case:	Procura de parques com filtragem de prefências	
Ator:	Utilizador	
Descrição:	Permite ao utilizador pesquisar por parques utilizando uma filtragem de preferências	
Cenários:	4	
Pré-condição:	True	
Pós-condição:	O utilizador recebe uma lista de parques relevantes	
	Utilizador	Sistema
Cenário normal		1. O sistema procura por parques de acordo com as preferências escolhidas 2. O sistema mostra uma lista de parques relevantes, assim como um mapa com todos eles indicados.

Tabela 3.7: *Use Case* Procura de parques com filtragem de preferências

Algumas considerações sobre os *Use Cases*

O grupo de trabalho tentou conceber *Use Cases* o mais sucintos possível de modo a que seja não só mais fácil para o grupo que irá tratar a implementação perceber o objetivo pretendido, como também para o nosso grupo conceber os restantes elementos da modelação. Contudo, existem aspetos que poderão causar alguma confusão a quem ler os *Use Cases* e que pretendemos, portanto, clarificar.

No *Use Case* "Editar Preferências", podemos observar que o utilizador pode no menu da edição da preferências, selecionar se, caso futuramente a sua opção de deslocação seja o seu veículo, pretende ou não portagens (como podemos observar no *mockup*), apesar desta opção parecer mais arcaica, replicando alguns sistemas de GPS, em que o utilizador selecionava as suas preferências antes da procura de um destino. A história ensinou-nos que estes sistemas perderam completamente o mercado, quando ferramentas mais rápidas e práticas como o *Google Maps* surgiram. Contudo, no caso específico desta aplicação (que como já mencionámos, pretendemos que seja o mais intuitiva e rápida possível para o utilizador), a lógica desses sistemas revela-se bastante pertinente. Isto, porque desta forma o utilizador não terá de estar a selecionar se pretende ou não portagens sempre que efetua um pedido de direções.

Este raciocínio acabou por ser estendido para as restantes preferências que o utilizador pode selecionar. Assim sendo, quando um utilizador seleciona a opção de procura, receberá logo a resposta do sistema, não tendo de passar por um processo de seleção de preferências. O utilizador terá no entanto, de o fazer previamente, mas esta é uma opção que se mostra especialmente vantajosa para utilizadores que raramente queiram alterar o que desejam procurar. Além disso, dado o número limitado de opções de preferências, esta opção de modelação e futura implementação, mostra-se ainda mais pertinente.

4 Conceção do sistema de dados

4.1 Apresentação geral da estrutura (esquema) do sistema de dados

A figura 4.1 mostra o modelo lógico da base de dados do projeto. Estão aqui, portanto, as várias tabelas utilizadas na base de dados e os seus atributos.

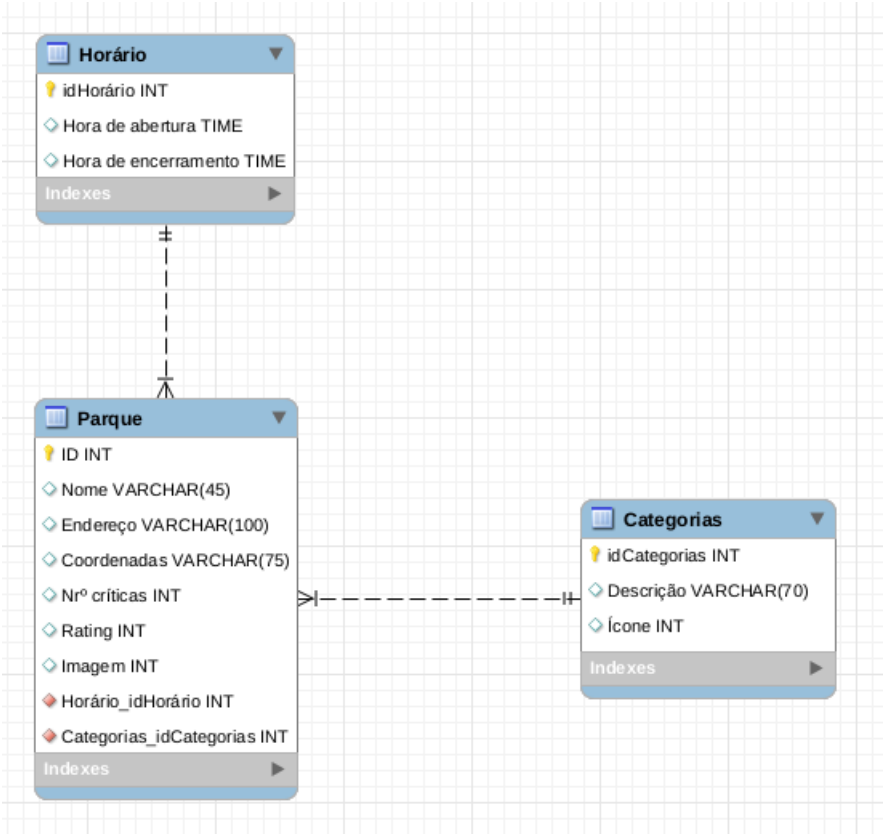


Figura 4.1: Sistema lógico da base de dados

4.2 Descrição detalhada dos vários elementos de dados e seus relacionamentos

- **Parque**
  - **ID**: Identificador de cada parque.
  - **Nome**: Nome do parque.
  - **Endereço**: Local onde se encontra o parque (rua, freguesia, etc.).
  - **Coordenadas**: Coordenadas GPS para localizar o parque no mapa.
  - **Nrº críticas**: Nrº total de críticas que um determinado parque tem.
  - **Rating**: Média de todas as críticas de um parque.
  - **Imagem**: Foto do parque.
- **Horário**
  - **idHorário**: Inteiro de 1 a 7 a representar o dia da semana (1 é segunda, 2 é terça, etc.). Um parque pode ter um horário diferente dependendo do dia da semana, logo temos um identificador para o dia.
  - **Hora da abertura**: Hora em que o parque abre.
  - **Hora de encerramento**: Hora em que o parque fecha.
- **Categorias**
  - **idCategorias**: Identificador das várias categorias que um parque pode ter. Havendo um total de 13 categorias já enumeradas anteriormente, este valor irá variar entre 1 e 13.
  - **Descrição**: Descrição do que cada uma das categorias é.
  - **Ícone**: Imagem que aparecerá no mapa a representar o parque, cada categoria terá um ícone diferente.

5 Esboço das Interfaces do Sistema

Neste capítulo pode-se encontrar uma sugestão daquilo que se pretende que a aplicação final replique com a maior precisão no que às interfaces do utilizador diz respeito. As seguintes figuras além de ilustrarem de forma mais sugestiva as funcionalidades da aplicação ***MinhoPark***, também evidenciam o quão intuitiva esta deve ser, de forma a poder ser utilizada por qualquer tipo de utilizador, não existindo uma quantidade assustadora de *widgets* desnecessários.

Figura 5.1: Menu principal

Figura 5.2: Menu de escolha de preferências (scrolled up)

Figura 5.3: Menu de escolha de preferências (scrolled down)

Figura 5.4: Lista de favoritos guardada (scrolled up)

Figura 5.5: Mapa com os parques mais próximos, assim como lista com os nomes desses parques

Figura 5.6: Depois de selecionar um parque, é mostrado mais informações sobre parque e opção de mostragem do caminho

Figura 5.7: Depois de selecionar uma opção de mostragem de caminho e carregar em "IR", aparece o caminho especificado no mapa

6 Conclusões e Trabalho Futuro

Ao contrário de projetos anteriores, o foco desta fase do projeto era conceptualizar e especificar detalhadamente uma aplicação idealizada pelo grupo, o *MinhoPark*: uma app focada no turismo de áreas verdes. Tal como todas as aplicações, a nossa sofre de uma série de aspetos negativos e positivos, que passamos agora a enumerar:

- Aspetos fracos:
  - Uma database limitada: Utilizar uma database de parques de apenas a região do Minho restringe muito o utilizador, caso este fosse utilizado num contexto menos académico.
- Aspetos positivos:
  - Arquitetura do programa: A modelação do nosso programa foi feita com modularidade em mente. Isto facilita a manutenção do código antigo e adição de novas funcionalidades.
  - Personalização: O utilizador tem bastantes opções de personalização daquilo que quer procurar.
  - User-friendly: Mesmo assim, aplicação é fácil e rápida de utilizar.

Em suma, tendo especial atenção em tentar fazer uma aplicação que seja de simples implementação e bem modelada, o grupo considerou que conseguiu com sucesso criar uma especificação realista que segue os objetivos e parametros propostos para esta fase do trabalho.

Referências

- 10ª edição do livro Software Engineering de Ian Sommerville (http://iansommerville.com/software-engineering-book/).

Lista de Siglas e Acrónimos

DAO Data Access Object

DSS Desenvolvimento de Sistemas de Software