Relatório Final

LICENCIATURA TECNOLOGIAS DA INFORMAÇÃO

RELATÓRIO FINAL NO ÂMBITO DA UNIDADE CURRICULAR PROJETO TEMÁTICO EM DESENVOLVIMENTO DE APLICAÇÕES

Sumário: Descrição geral do sistema contextualizado por meio de requisitos e pela modelação com base em diagramas de classe, packages, casos de utilização, sequência e pelo modelo de dados persistentes que descreve a estrutura lógica da base de dados.

Grupo: Daniel Martins (Nº 83645)

Maria Santos (Nº 87965)

Ricardo Balreira (Nº 88078)

Tiago Cunha (Nº 69964)

Relatório Final

LICENCIATURA TECNOLOGIAS DA INFORMAÇÃO

RELATÓRIO FINAL NO ÂMBITO DA UNIDADE CURRICULAR PROJETO TEMÁTICO EM DESENVOLVIMENTO DE APLICAÇÕES

Grupo: Daniel Martins (Nº 83645)

Maria Santos (Nº 87965)

Ricardo Balreira (Nº 88078)

Tiago Cunha (Nº 69964)

Índice

1 Introdução 1

1.1 Visão geral do sistema 1

1.2 Cliente 2

1.3 Objetivos 3

2 Planeamento 4

3 Modelo de requisitos 9

3.1 Requisitos funcionais 9

3.2 Restrições e requisitos não funcionais 12

4 Modelo de Casos de Utilização 13

4.1 Visão geral 13

4.2 Atores 14

4.3 Descrição dos casos de utilização 15

4.3.1 Consultar 15

4.3.2 Ver perfil 16

4.3.3 Consultar lugares vagos de estacionamento 17

4.3.4 Reservar um lugar de estacionamento 18

4.3.5 Atualizar preço 19

4.3.6 Visualizar estatística 20

4.3.7 Consultar utilizadores 21

4.3.8 Criar conta 22

4.3.9 Pagar lugar de estacionamento 23

4.3.10 Registar matrícula 24

5 Modelo dinâmico 27

5.1 Funções de sistema....27

6 Modelo estrutural....30

6.1 Organização da solução....30

6.2 Diagrama de classes....32

7 Modelo de dados persistente....35

7.1 Descrição do problema na ótica da base de dados....35

Regras de Transposição:....37

11 Criptografia....54

13.1 Registo....58

13.2 LOGIN....58

13.3 Utilizador....59

13.3.1 Página Inicial....59

13.3.1.1 Ver Parques....59

13.3.1.2 Obter Conta Premium....59

13.3.1.3 Sobre....59

13.3.1.4 Terminar Sessão....59

13.3.2 Ver Perfil....60

13.3.2.1 Ver Dados....60

13.3.2.2 Adesão Conta Premium....61

13.3.2.3 Gestão de Matrículas....61

13.3.2.4 Histórico de Pagamentos....61

13.3.2.5 Classificar a Aplicação....62

13.3.2.6 Remoção de Conta....62

13.3.3 Administrador....62

13.3.3.1 Menu....62

13.3.3.2 Atualizar Preço....63

13.3.3.3 Estatísticas.....63

13.3.3.3.1 Média Ocupação 63

13.3.3.3.2 Afluência Local 63

13.3.3.3.3 Consultar Lucros 63

13.3.3.3.4 Classificação da App 64

13.3.3.4 Consultar Utilizador.....64

13.3.3.4.1 Top Utilizadores.....64

13.3.3.4.2 Multas.....64

13.3.3.4.3 Remover Utilizador 64

Índice de figuras

Figura 1 – Diagrama com a distribuição de tarefas (Parte 1) 5

Figura 2 - Diagrama com a distribuição das tarefas (Parte 2) 6

Figura 3 – Duração das tarefas (Parte 1).....7

Figura 4 - Duração das tarefas (Parte 2).....8

Figura 5 - Caso de utilização "Consultar" do diagrama de casos de utilização.....15

Figura 6 - Caso de utilização "Ver perfil" do diagrama de casos de utilização.....16

Figura 7 - Caso de utilização "Consultar lugares vagos" do diagrama de casos de utilização 17

Figura 8 - Caso de utilização "Reservar um lugar de estacionamento" do diagrama de casos de utilização 18

Figura 9 - Caso de utilização "Atualizar preços" do diagrama de casos de utilização 19

Figura 10 - Caso de utilização "Visualizar estatística" do diagrama de casos de utilização 20

Figura 11 - Caso de utilização "Consultar utilizadores" do diagrama de casos de utilização 21

Figura 12 - Caso de utilização "Criar conta" do diagrama de casos de utilização 22

Figura 13 - Caso de utilização "Pagar lugar estacionamento" do diagrama de casos de utilização 23

Figura 14 - Caso de utilização "Registar matrícula" do diagrama de casos de utilização.24

Figura 15 - Casos de utilização do ator Convidado.....25

Figura 16 - Casos de utilização dos atores Utilizador Ordinário, SIBS e PayPal 25

Figura 17 - Casos de utilização do ator Utilizador *Premium*.....26

Figura 18 - Casos de utilização do ator Administrador.....26

Figura 19 - Sequência de receber fatura ou gerar multa pelo método SIBS.....28

Figura 20 - Sequência de receber fatura ou gerar multa pelo método PayPal 29

Figura 21 - Diagrama de sequência.....29

Figura 22 - Diagrama de packages.....30

Figura 23 - Publish Project 31

Figura 24 - Diagrama de classes 34

Figura 25 - Modelo de dados persistentes obedecendo às regras de transposição
.....
42

Figura 26 - Modelo de dados persistentes com otimização 46

Índice de tabelas

Tabela 1 - Requisitos Funcionais do Administrador 9

Tabela 2 - Requisitos funcionais do Utilizador *Premium*.....10

Tabela 3 - Requisitos funcionais do Utilizador Ordinário 10

Tabela 4 - Requisitos funcionais do Convidado.....11

Tabela 5 - Requisitos não funcionais 12

Tabela 6 - Descrição do papel a desempenhar por cada ator 14

Tabela 7 - Descrição do caso de utilização "Consultar" 15

Tabela 8 - Descrição do caso de utilização "Ver perfil".....16

Tabela 9 – Descrição do caso de utilização “Consultar lugares vagos de estacionamento” 17

Tabela 10 - Descrição do caso de utilização “Reservar um lugar de estacionamento" 18

Tabela 11 - Descrição do caso de utilização "Atualizar preço" 19

Tabela 12 - Descrição do caso de utilização "Visualizar estatística".....20

Tabela 13 - Descrição do caso de utilização "Consultar utilizadores".....21

Tabela 14 - Descrição do caso de utilização "Criar conta" 22

Tabela 15 - Descrição do caso de utilização "Pagar lugar de estacionamento".....23

Tabela 16 - Descrição do caso de utilização "Registar matrícula" 24

1 Introdução

1.1 Visão geral do sistema

Tendo em vista a implementação de um sistema exprimido por meio da modelação e por uma metodologia de desenvolvimento de software, o tema escolhido com base num conjunto de temas propostos pelo Projeto Temático englobava a obtenção de um sistema que fosse capaz de controlar o tráfego para municípios consoante um conjunto de variáveis que permitisse determinar o custo à adesão de um lugar de estacionamento, por exemplo. Qualquer aspeto que estivesse relacionado com a gestão de tráfego era bem-vindo pelo facto de ser um tema livre e bastante abrangente.

Com efeito, adaptou-se a proposta para uma cidade urbana onde há uma diversidade imensa de parques de estacionamento. O sistema permite estipular preços para o pagamento de um lugar de estacionamento consoante um determinado período do dia. Dependendo do período do dia (de manhã, hora de almoço, fim de semana…), os preços irão subir. Para isso, o sistema irá registar o número de lugares ocupados e livres num dado estacionamento e os períodos onde há maior influência, com o devido acesso a cada ponto de referência no mapa do concelho. Além disso, terá também a componente de poder reservar um lugar num estacionamento, caso este tenha uma conta *premium* (recompensa).

Atualmente há uma enorme concorrência e uma grande procura pelas pessoas em obter um lugar de estacionamento, sendo notório nos grandes centros urbanos. Pelo mesmo motivo que é necessário desenvolver sistemas que permitam reduzir essa condicionante que é um tormento constante. Embora seja impossível achar uma solução que agrade a todos (exemplo: uma pessoa prefere não optar pela conta *premium* e pagar no próprio local estando sujeita a perder o lugar assim que chegar ao destino), é uma motivação poder reduzir ao máximo esse tipo de questões que surgem no quotidiano.

Para acrescentar e tal como foi referido anteriormente, outro motivo que levou a escolher este tema foi pela diversidade de opções que existem e que se pode implementar.

1.2 Cliente

No que respeita ao cliente, uma vez que será ele que vai interagir com o sistema, haverá um conjunto de funcionalidades que este poderá desfrutar e ter acesso tirando partido do objetivo da aplicação. Do mesmo modo que o potencial utilizador do sistema estará ligado ao seguinte: uma faixa etária entre os 18 e os 60 anos, sendo uma mais-valia para a adequabilidade do produto; sabe usar dispositivos móveis; estão familiarizados com ecrãs táteis; ensino secundário ou superior; que tenha o hábito de se deslocar para o trabalho de transporte privado ou que o utilize regularmente durante o dia. Considerando a descrição acima, este utilizador/cliente será o mais plausível para o sistema em questão. Seguindo esta lógica, o cliente do sistema tem as seguintes possibilidades: registar-se na aplicação, criando uma conta com os dados pessoais; consultar preços dos parques de estacionamento distribuídos na cidade segmentados por zonas (zona norte, sul, centro, este e oeste); pagar lugar de estacionamento de uma forma mais eficiente, isto é, a pessoa apenas paga quando estiver fora do das linhas que delimitam o lugar, indicando que o lugar se encontra agora livre. O preço do lugar é calculado tendo em consideração a diferença entre o tempo de fim e o tempo de início na qual usufruiu o lugar e o produto do preço total. Caso a pessoa não tenha dinheiro suficiente, é gerada uma multa; consultar os seus pagamentos efetuados, bem como multas. Caso o utilizador seja apenas um *guest* (convidado) ou tenha uma conta *premium* terá, obviamente, restrições ou funcionalidades ilimitadas, respetivamente.

1.3 Objetivos

De maneira a enfatizar os objetivos de negócio por meio de uma analogia, imagine-se um empresário que acabou de abrir uma empresa. O seu foco será claramente obter o maior lucro estando sempre acima das expectativas dos seus clientes, conquistando confiança, obtendo sempre opiniões positivas. Para isso, terá como dever estudar as melhores técnicas de negócio, falar com as pessoas certas, contactar fornecedores e especialistas na área a investir, os melhores serviços, ser empreendedor, isto é, que seja capaz e que tenha a frieza de superar dificuldades, obstáculos e de resolver problemas e acima de tudo, ter as melhores condições de trabalho. O fator crucial será então a organização, permitindo que o negócio se possa expandir e crescer exponencialmente e nunca decrescer, sendo sempre consistente. É exatamente isto que se pretende atingir com o desenvolvimento deste sistema. Apesar das pessoas serem o recurso que é mais desafiador, ainda mais numa aplicação que incorpora gestão de tráfego, é necessário contabilizar imenso o conflito de interesses para poder conquistar o maior público-alvo possível. O objetivo aqui é então ir ao encontro dos interesses das pessoas, obter o melhor sistema de gestão de tráfego dos estacionamentos e encontrar a melhor solução para evitar com que o cliente tenha de se deslocar para pagar um lugar num estacionamento ou que tenha de apresentar um comprovativo de que pagou por esse mesmo lugar.

2 Planeamento

A fase de planeamento deve contemplar várias fases onde se definem os objetivos gerais e de segunda ordem, o tipo de tarefas, os recursos necessários à concretização dos objetivos e das tarefas, a pesquisa e o tipo de ligações (relações) entre tarefas. Por outras palavras, consiste em estipular tarefas pelos elementos do grupo, com objetivos e prazos. Deste modo, algumas tarefas vão depender do atual progresso de outras.

Considerando que numa primeira fase a noção das tarefas a desempenhar na implementação do sistema é pouco notória, pois as primeiras etapas do projeto consistiram apenas na modelação da aplicação com base em diagramas que representam de forma esquemática o papel do utilizador ao interagir com o sistema, é de afirmar que a sequência de tarefas estabelecida teve mais ênfase e domínio no desenvolvimento do projeto.

Com efeito, os diagramas seguintes (Figuras 1 e 2) demonstram o nome de cada atividade ou tarefa desempenhada no decorrer do projeto, enfatizando as fases de gestão da Base de Dados, o desenho das interfaces e a lógica de negócio (onde reside a lógica das decisões e o processamento de dados entre a interface e a base de dados) na qual se enquadra no modelo em três camadas da aplicação. Ainda é possível ver o período onde se tomaram debateram os requisitos funcionais e não funcionais, isto é, a funcionalidade do sistema, e ainda a modelação onde se modela o sistema por meio de diagrama de casos de utilização, classes e sequência. A figuras 3 e 4 dizem respeito ao início, término e duração das tarefas.

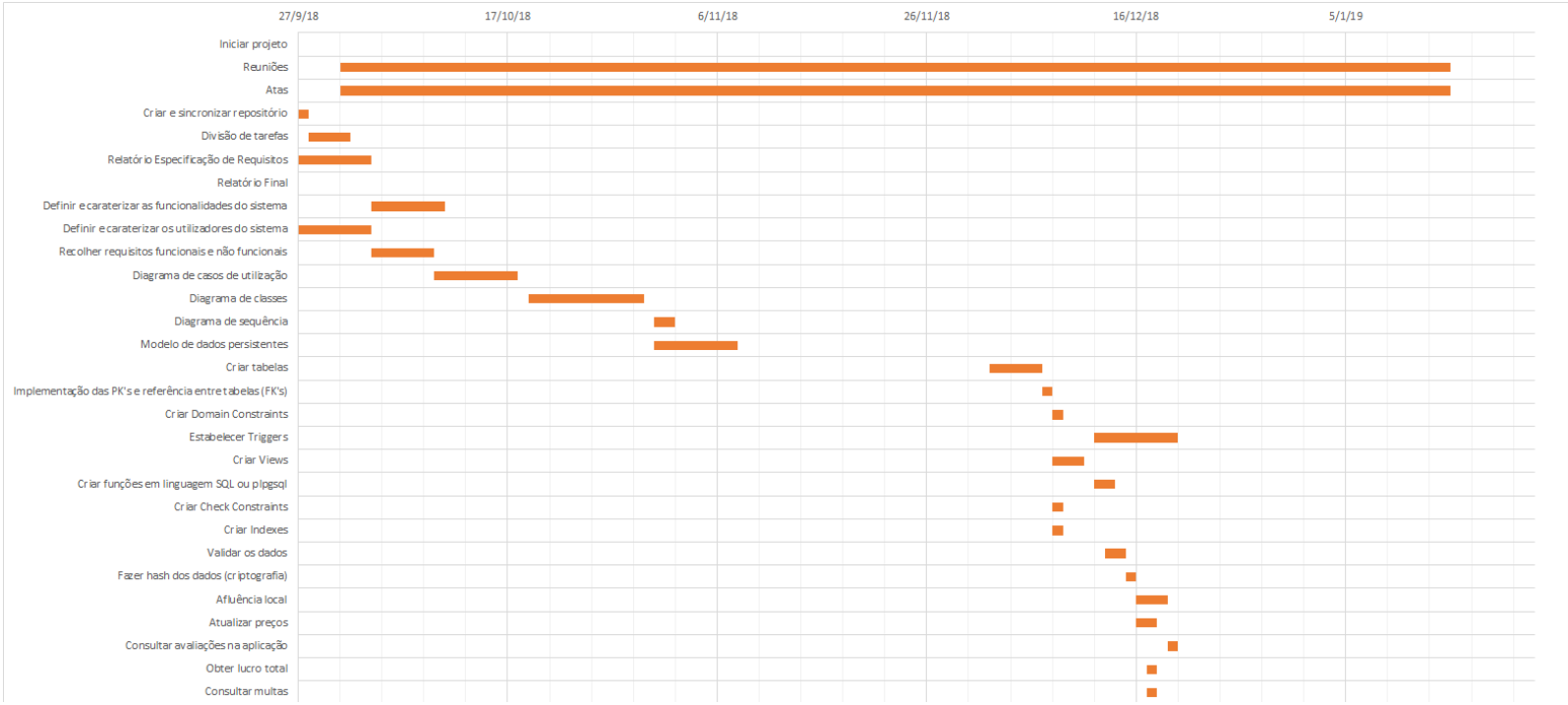

Figura 1 – Diagrama com a distribuição de tarefas (Parte 1)

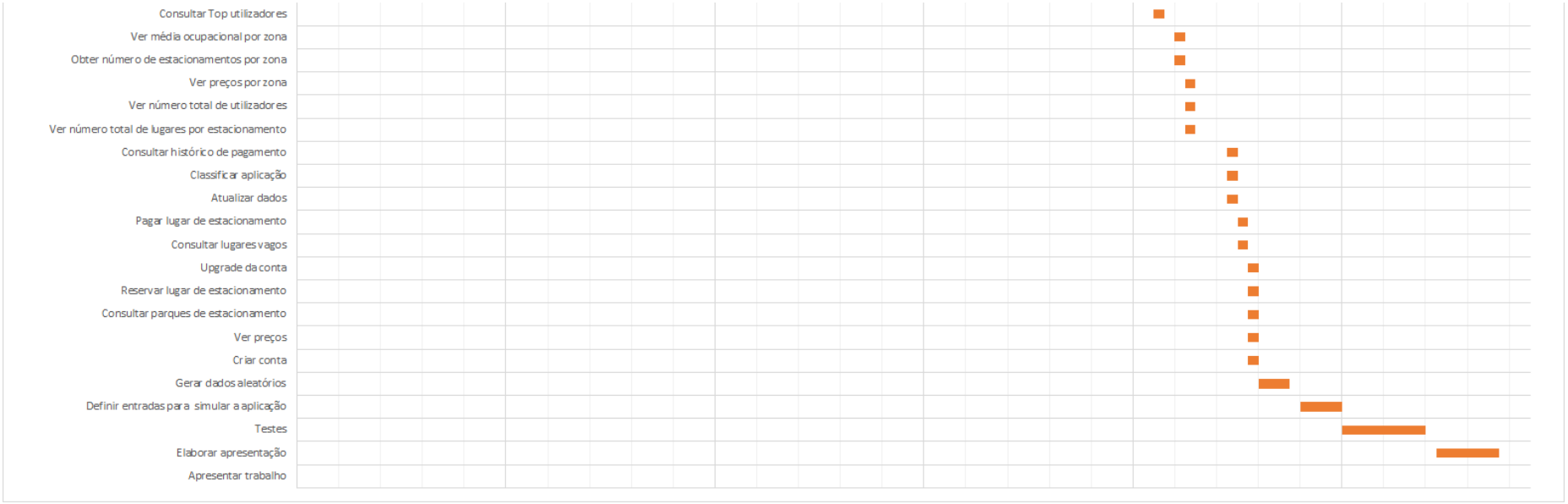

Figura 2 - Diagrama com a distribuição das tarefas (Parte 2)

Tarefas	Início	Duração	Término	Dependências
Iniciar projeto	27/09/2018	0	27/09/2018	
Reuniões	01/10/2018	106	15/01/2019	5II
Atas	01/10/2018	106	15/01/2019	6II
Criar e sincronizar repositório	27/09/2018	1	28/09/2018	5CI
Divisão de tarefas	28/09/2018	4	02/10/2018	6II
Relatório Especificação de Requisitos	27/09/2018	7	04/10/2018	
Relatório Final	01/12/2019	4	05/12/2019	
Definir e caraterizar as funcionalidades do sistema	04/10/2018	7	11/10/2018	9CI
Definir e caraterizar os utilizadores do sistema	27/09/2018	7	04/10/2018	12CI
Recolher requisitos funcionais e não funcionais	04/10/2018	6	10/10/2018	13CI
Diagrama de casos de utilização	10/10/2018	8	18/10/2018	14CI
Diagrama de classes	19/10/2018	11	30/10/2018	14CI
Diagrama de sequência	31/10/2018	2	02/11/2018	16CI; 18CI
Modelo de dados persistentes	31/10/2018	8	08/11/2018	14CI
Criar tabelas	02/12/2018	5	07/12/2018	15CI;16CI;17CI;18CI
Implementação das PK's e referência entre tabelas (FK's)	07/12/2018	1	08/12/2018	19CI
Criar Domain Constraints	08/12/2018	1	09/12/2018	20CI
Estabelecer Triggers	12/12/2018	8	20/12/2018	20CI
Criar Views	08/12/2018	3	11/12/2018	20CI
Criar funções em linguagem SQL ou plpgsql	12/12/2018	2	14/12/2018	20CI
Criar Check Constraints	08/12/2018	1	09/12/2018	20CI
Criar Indexes	08/12/2018	1	09/12/2018	20CI
Validar os dados	13/12/2018	2	15/12/2018	19CI
Fazer hash dos dados (criptografia)	15/12/2018	1	16/12/2018	19CI
Afluência local	16/12/2018	3	19/12/2018	23CI
Atualizar preços	16/12/2018	2	18/12/2018	20CI

Figura 3 – Duração das tarefas (Parte 1)

Tarefas	Início	Duração	Término	Dependências
Consultar avaliações na aplicação	19/12/2018	1	20/12/2018	20CI
Obter lucro total	17/12/2018	1	18/12/2018	24CI
Consultar multas	17/12/2018	1	18/12/2018	20CI
Consultar Top utilizadores	18/12/2018	1	19/12/2018	23CI
Ver média ocupacional por zona	20/12/2018	1	21/12/2018	23CI
Obter número de estacionamentos por zona	20/12/2018	1	21/12/2018	
Ver preços por zona	21/12/2018	1	22/12/2018	20CI
Ver número total de utilizadores	21/12/2018	1	22/12/2018	
Ver número total de lugares por estacionamento	21/12/2018	1	22/12/2018	
Consultar histórico de pagamento	25/12/2018	1	26/12/2018	23CI
Classificar aplicação	25/12/2018	1	26/12/2018	20CI
Atualizar dados	25/12/2018	1	26/12/2018	
Pagar lugar de estacionamento	26/12/2018	1	27/12/2018	
Consultar lugares vagos	26/12/2018	1	27/12/2018	
Upgrade da conta	27/12/2018	1	28/12/2018	20CI
Reservar lugar de estacionamento	27/12/2018	1	28/12/2018	
Consultar parques de estacionamento	27/12/2018	1	28/12/2018	20CI
Ver preços	27/12/2018	1	28/12/2018	20CI
Criar conta	27/12/2018	1	28/12/2018	20CI
Gerar dados aleatórios	28/12/2018	3	31/12/2018	
Definir entradas para simular a aplicação	01/01/2019	4	05/01/2019	
Testes	05/01/2019	8	13/01/2019	11II
Elaborar apresentação	14/01/2019	6	20/01/2019	11CI
Apresentar trabalho	23/01/2019	0	23/01/2019	

Figura 4 - Duração das tarefas (Parte 2)

3 Modelo de requisitos

Descrevendo as funcionalidades do sistema, isto é, especificando o que o sistema deve fazer obrigatoriamente, exigindo codificação (Tabelas 1, 2, 3, 4), e como o sistema se deve comportar tendo em conta, por exemplo, a usabilidade e desempenho (Tabela 5), as tabelas 1, 2, 3, 4 referem-se aos requisitos funcionais e a tabela 5 aos requisitos não funcionais.

3.1 Requisitos funcionais

Referência - numeração do requisito

Descrição - texto descritivo do requisito

Tipo - grau de prioridade do requisito (alto, médio, baixo)

Tabela 1 - Requisitos Funcionais do Administrador

Ref[a]	**Requisitos funcionais**	**Tipo**
RFA. 1	Validar acesso quando faz a autenticação certificando que é a pessoa cujo papel é gerir alterar dados na aplicação	Alto
RFA. 2	Ter acesso à média ocupacional por zona e estacionamentos, obtendo ainda toda a informação da zona ou estacionamento, total de lugares e de lugares ocupados	Médio
RFA. 3	Consulta todos os utilizadores registados no sistema	Alto
RFA. 4	Gere todos os preços base de cada parque do sistema, podendo alterá-los ao seu critério	Alto
RFA. 5	Remove utilizadores que deixaram de utilizar a aplicação ou que a utilizem para fins maliciosos	Médio
RFA. 6	Visualizar a estatística de vendas, como por exemplo: quantos utilizadores usam a aplicação para reservar lugares, pagar o estacionamento pela aplicação, o top de utilizadores que mais renderam financeiramente pela aplicação e ainda o lucro total	Médio

Refª	**Requisitos funcionais**	**Tipo**
RFA. 7	Consultar o número de utilizadores que receberam multas e uma lista com toda a informação de cada multa	Médio

Tabela 2 - Requisitos funcionais do Utilizador *Premium*

Refª	**Requisitos funcionais**	**Tipo**
RFUP. 1	Reserva um lugar de estacionamento no parque escolhido, alterando um estado do lugar de livre para ocupado indicando que o lugar já está a ser utilizado ou reservado	Alto
RFUP. 2	Pagar lugar de estacionamento num preço mais acessível, relativamente ao utilizador Ordinário	Alto

Tabela 3 - Requisitos funcionais do Utilizador Ordinário

Refª	**Requisitos funcionais**	**Tipo**
RFU. 1	Validar acesso com número de telemóvel e palavra-passe quando faz a autenticação	Alto
RFU. 2	Pode ter acesso ao seu perfil, tais como: dados pessoais, pagamentos e multas. Por outras palavras, pode ver o seu histórico de pagamento. Pode também classificar a aplicação.	Médio
RFU. 3	Atualizar um campo do seu perfil como por exemplo: palavra-passe, morada, código-postal, número de telemóvel, IBAN, matrícula…	Médio
RFU. 4	Consultar o número de lugares vagos existentes no estacionamento desejado	Alto
RFU. 5	Fazer um upgrade da conta, pagando um x preço de forma a tornar-se num utilizador *Premium*	Alto

Ref[a]	**Requisitos funcionais**	**Tipo**
RFU. 6	Após a saída do lugar de estacionamento, o lugar fica livre e é gerado um custo tendo em conta o tempo de ocupação e o preço da zona respetiva	Alto
RFU. 7	Esclarece dúvidas sobre a aplicação aos utilizadores, como por exemplo, “Como pago o tempo que estive estacionado?”	Baixo
RFU. 8	Classificar aplicação, atribuindo uma nota numa escala de 0 a 5 e um comentário do grau de satisfação da aplicação	Baixo
RFU. 9	O utilizador pode apenas registar, no máximo, quatro matrículas podendo utilizar os estacionamentos com os veículos que tenham essas matrículas associadas	Médio

Tabela 4 - Requisitos funcionais do Convidado

Ref[a]	**Requisitos funcionais**	**Tipo**
RFC. 1	Consulta os diversos parques que estão registados no sistema e respetivos preços	Alto
RFC. 2	Regista-se no sistema preenchendo os dados pessoais e registando uma palavra-passe caso se queira tornar num utilizador Ordinário	Alto

É importante frisar que, a partir do momento que um Convidado se regista na aplicação e se torna num utilizador Ordinário, usufrui das mesmas opções que o Convidado, isto é, pode consultar os parques de estacionamento e respetivos preços. O mesmo se aplica ao utilizador *Premium* que tem direito a realizar tudo o que o utilizador Ordinário e o Convidado fazem, e ao Administrador que pode ter acesso a todas as funcionalidades que ambos os utilizadores e Convidado têm. Por esta razão, a herança que existe nalguns tipos de papéis acaba também por se considerar num requisito funcional.

3.2 Restrições e requisitos não funcionais

Tabela 5 - Requisitos não funcionais

Ref[a]	**Requisito não funcional**	**Tipo**
RNF. 1	Obter uma conta *premium* por tempo de adesão, isto é, pelo período que a pessoa já possui uma conta registada na aplicação	Baixo
RNF. 2	Tornar eficiente e rápida uma pesquisa por um determinado conjunto de dados na base de dados	Médio
RNF. 3	Aumentar o preço da respetiva zona em função do tráfego, ou seja, se houver bastante adesão numa hora pouco habitual numa determinada os preços sobem	Baixo
RNF. 4	Alterar os preços de todas as zonas num determinado período do dia (por exemplo: de manhã, hora de almoço, sexta-feira e sábado à noite)	Baixo
RNF. 5	Tempo de resposta nas ações executadas pelo utilizador, por exemplo, na validação dos dados	Médio

4 Modelo de Casos de Utilização

4.1 Visão geral

Dependendo da utilidade e dos requisitos que fundamentam um determinado sistema numa situação ou tema em específico, o diagrama de casos de utilização tem como objetivo capturar as funcionalidades da aplicação tal como é visto pelos utilizadores. Especifica o contexto do problema e esquematiza os requisitos funcionais. Imagine-se no caso de utilizar um terminal multibanco. O utilizador ao interagir com o sistema terá um conjunto de casos de utilização que este, por sua vez, oferece. Casos esses que podem ser: levantar ou depositar dinheiro, ver saldo, verificar operações, entre outros. Qualquer interação que exista entre o sistema e o utilizador define-se como um caso de utilização.

No que toca à aplicação, de um modo geral, haverá um conjunto de atores que interagirão com a *app* exigindo o surgimento de outros atores externos para o controlo da opção escolhida pelo utilizador sendo, nalguns casos, devido à dependência na confirmação do pedido (exemplo: levantar dinheiro exige que um ator externo como o SIBS – Sociedade Interbancária de Serviços – verifique o valor inserido com o saldo no cartão). Atores como, por exemplo, o convidado, apenas tem a possibilidade de consultar preços e estacionamentos e proceder ao registo na aplicação, o utilizador Ordinário usufrui dessas funcionalidades incluindo o pagar um lugar de estacionamento que irá estabelecer ligação com o ator SIBS devido ao que foi referido no exemplo anterior, consultar lugares vagos e ainda ver perfil. Por fim, o ator utilizador Ordinário ao tornar-se um utilizador *premium* (compra da aplicação) tem a vantagem de poder reservar um lugar de estacionamento e o administrador tem a possibilidade de consulta e gestão de dados que são guardados no sistema (Tabela 6).

Na modelação do sistema com base num diagrama de casos de utilização, que também se encontra nos anexos, é possível observar o seguinte: a disposição de mais do que um caso de utilização de cada ator por meio de um package, a generalização que existe entre os atores Utilizador, Utilizador *Premium* e Administrador e o uso das relações *extend* (opcional) e *include* (exemplo: validar acesso é um caso obrigatório pelo que deverá ser integrado num *include*) entre os casos de utilização.

Os seguintes subtópicos contextualizam o que fora dito anteriormente por meio de tabelas, relativamente à descrição da interação de cada ator no sistema (Tabela 6) e à abordagem lógica de cada caso de utilização referindo os atores em comum, prioridade no desempenho da tarefa, objetivo e a que requisitos funcionais se enquadra e ainda com um esquema referente ao diagrama de casos utilização integral (Tabelas do subtópico "Descrição dos casos de utilização").

4.2 Atores

Tabela 6 - Descrição do papel a desempenhar por cada ator

Ator	**Descrição**
Convidado	Pessoa que pode consultar os parques de estacionamento ou os respetivos preços e proceder ao registo na aplicação;
Utilizador Ordinário	Utilizador registado que pode efetuar as mesmas operações que o convidado podendo, para além disso, visualizar os lugares vagos, pagar um lugar de estacionamento, esclarecer dúvidas, registar uma matrícula temporária e ainda, ver perfil
Utilizador Premium	Pessoa que efetua as mesmas operações dos atores descritos anteriormente, tendo ainda o privilégio de reservar um lugar
Administrador	Pessoa que define os preços base de forma manual, analisa estatísticas e consulta utilizadores, para além das opções que o utilizador ordinário e *premium* realizam
SIBS e PayPal	Atores que simbolizam o método de pagamento que o utilizador pretende ao pagar um lugar de estacionamento

4.3 Descrição dos casos de utilização

4.3.1 Consultar

Tabela 7 - Descrição do caso de utilização "Consultar"

Nome:	**Consultar**
Atores:	Convidado, Utilizador Ordinário, Utilizador *Premium*, Administrador
Prioridade:	Alto
Finalidade:	Consultar os vários parques de estacionamento ou os respetivos preços;
Requisitos Funcionais:	RFC. 1, RFC. 2

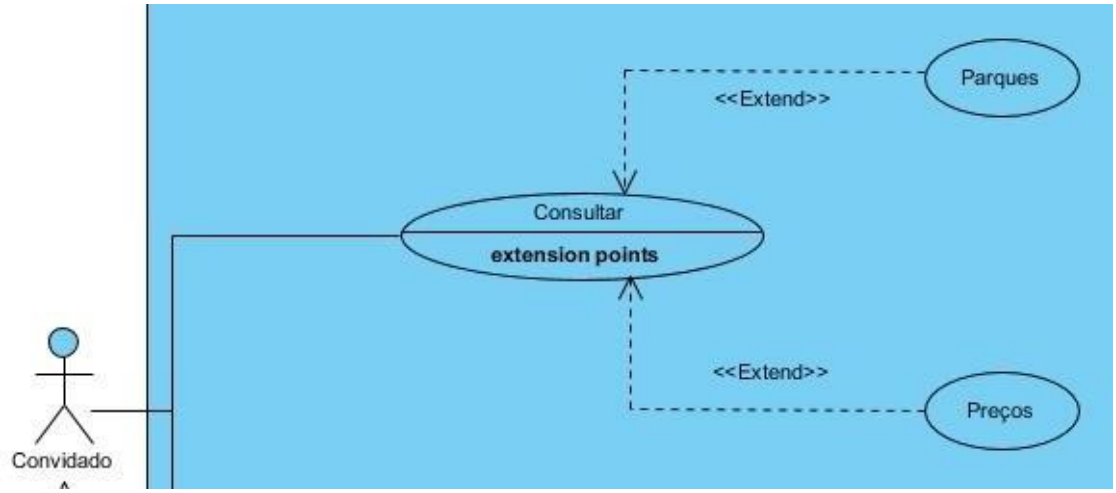

Figura 5 - Caso de utilização "Consultar" do diagrama de casos de utilização

4.3.2 Ver perfil

Tabela 8 - Descrição do caso de utilização "Ver perfil"

Nome:	**Ver perfil**
Atores:	Utilizador Ordinário, Utilizador *Premium*, Administrador
Prioridade:	Médio
Finalidade:	Consultar detalhes e informações da conta, tais como: ver o histórico, adquirir uma conta *premium* com a compra da aplicação, classificar a aplicação, adicionar comentário ou ainda atualizar os dados;
Requisitos Funcionais:	RFU. 2

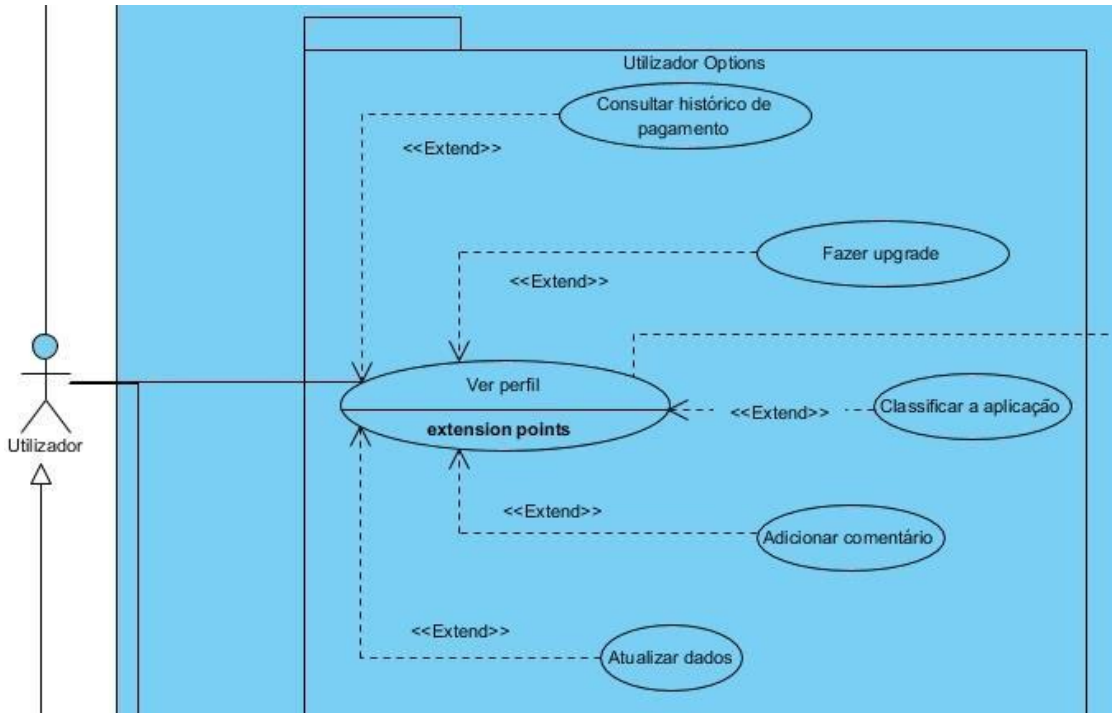

Figura 6 - Caso de utilização "Ver perfil" do diagrama de casos de utilização

4.3.3 Consultar lugares vagos de estacionamento

Tabela 9 – Descrição do caso de utilização “Consultar lugares vagos de estacionamento”

Nome:	**Consultar lugares vagos de estacionamento**
Atores:	Utilizador Ordinário, Utilizador *Premium*, Administrador
Prioridade:	Alto
Finalidade:	Consultar lugares vagos permite visualizar o número de lugares disponíveis em cada um dos diversos parques da cidade;
Requisitos Funcionais:	RFU. 4

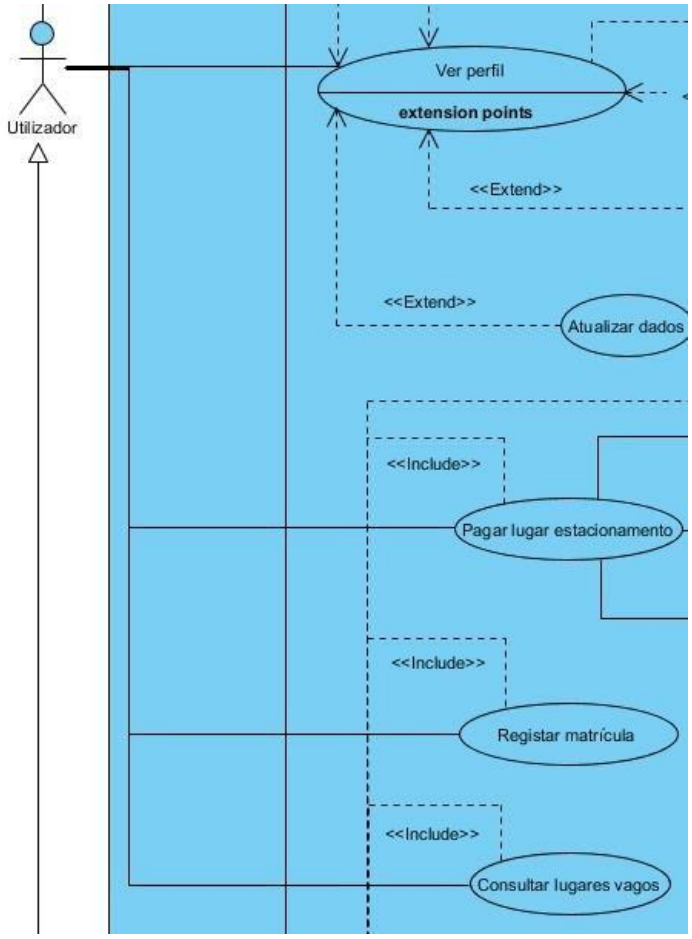

Figura 7 - Caso de utilização "Consultar lugares vagos" do diagrama de casos de utilização

4.3.4 Reservar um lugar de estacionamento

Tabela 10 - Descrição do caso de utilização “Reservar um lugar de estacionamento"

Nome:	**Reservar um lugar de estacionamento**
Atores:	Utilizador *Premium*, Administrador
Prioridade:	Alto
Finalidade:	Reservar um lugar de estacionamento proporciona ao utilizador *premium* e mesmo ao administrador a possibilidade de reservar um lugar, alterando o estado do lugar de livre para ocupado;
Requisitos Funcionais:	RFUP. 1

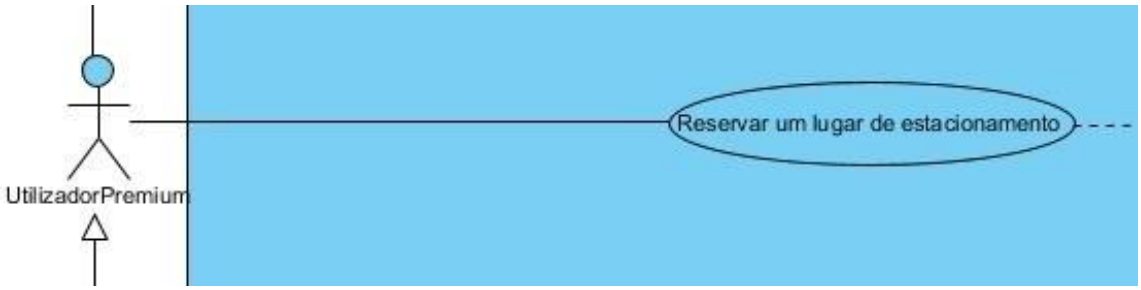

Figura 8 - Caso de utilização "Reservar um lugar de estacionamento" do diagrama de casos de utilização

4.3.5 Atualizar preço

Tabela 11 - Descrição do caso de utilização "Atualizar preço"

Nome:	**Atualizar preço**
Atores:	Administrador
Prioridade:	Alto
Finalidade:	Atualizar preços definindo um preço base;
Requisitos Funcionais:	RFA. 4

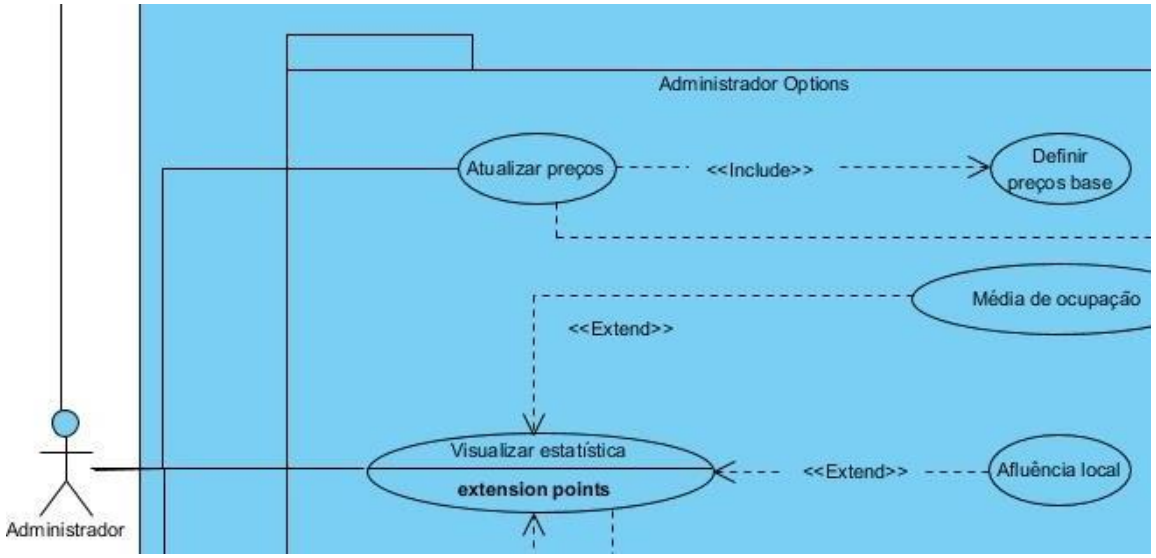

Figura 9 *- Caso de utilização "Atualizar preços" do diagrama de casos de utilização*

4.3.6 Visualizar estatística

Tabela 12 - Descrição do caso de utilização "Visualizar estatística"

Nome:	**Visualizar estatística**
Atores:	Administrador
Prioridade:	Médio
Finalidade:	Visualizar estatística pelo administrador sobre a média de adesão num dado parque de estacionamento, a afluência local (se está com muita ou pouca aderência) ou ainda a consulta do lucro total;
Requisitos Funcionais:	RFA. 2, RFA. 6

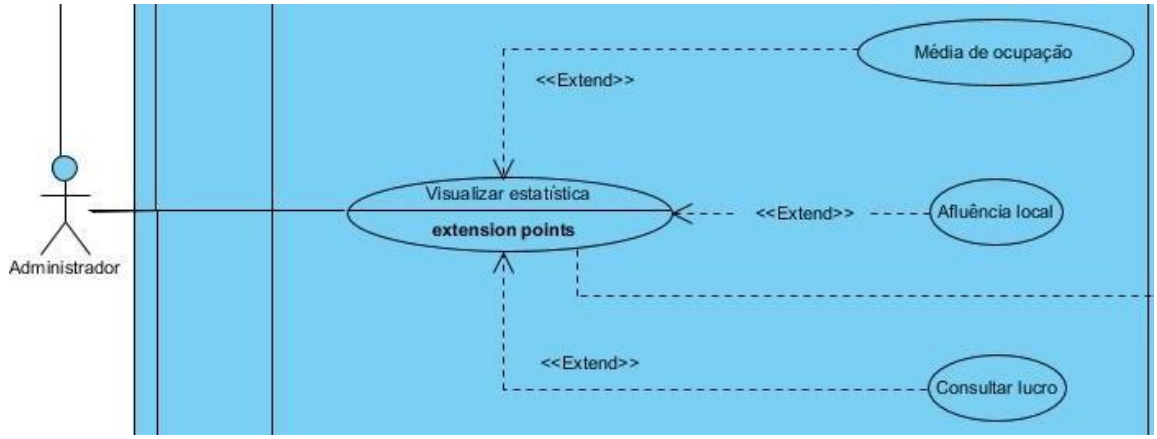

Figura 10 - Caso de utilização "Visualizar estatística" do diagrama de casos de utilização

4.3.7 Consultar utilizadores

Tabela 13 - Descrição do caso de utilização "Consultar utilizadores"

Nome:	**Consultar utilizadores**
Atores:	Administrador
Prioridade:	Alto
Finalidade:	Consultar utilizadores obtendo informações sobre os utilizadores mais influentes e multas, ou ainda também a possibilidade de remover utilizadores;
Requisitos Funcionais:	RFA. 3

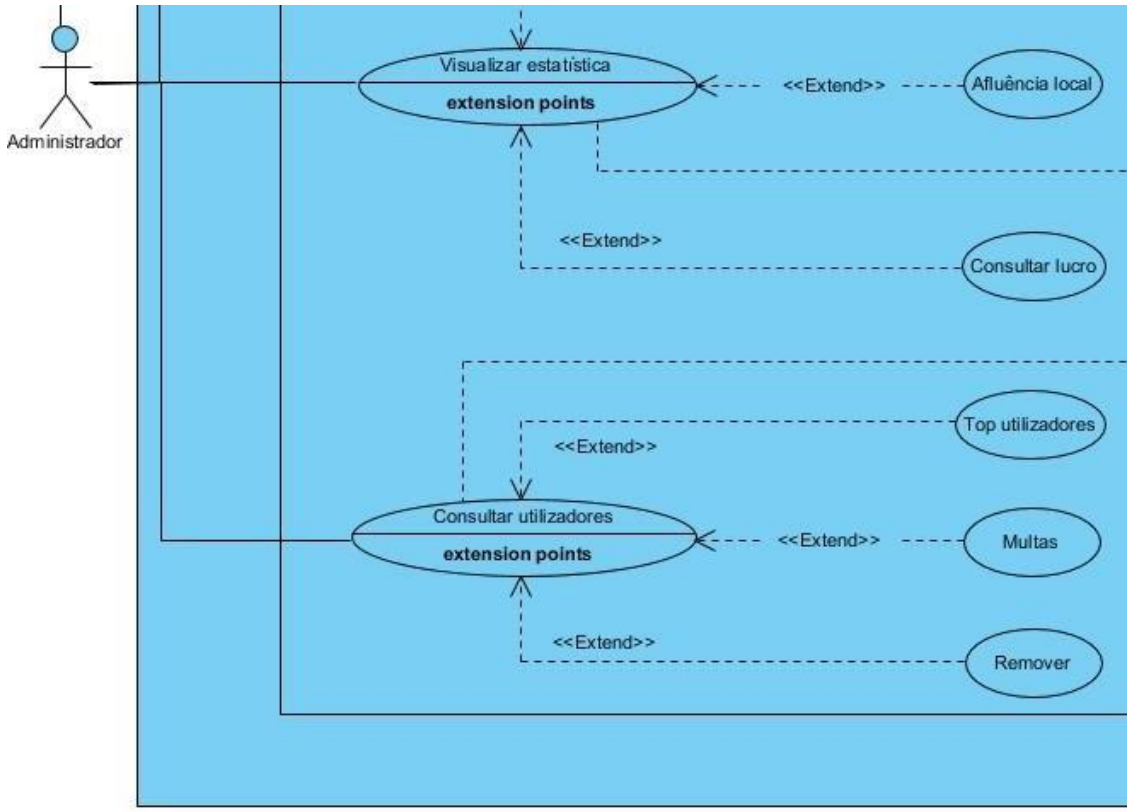

Figura 11 - Caso de utilização "Consultar utilizadores" do diagrama de casos de utilização

4.3.8 Criar conta

Tabela 14 - Descrição do caso de utilização "Criar conta"

Nome:	**Criar conta**
Atores:	Convidado
Prioridade:	Alto
Finalidade:	Criar conta permitindo ao *guest* (convidado) registar-se na aplicação;
Requisitos Funcionais:	RFC. 2

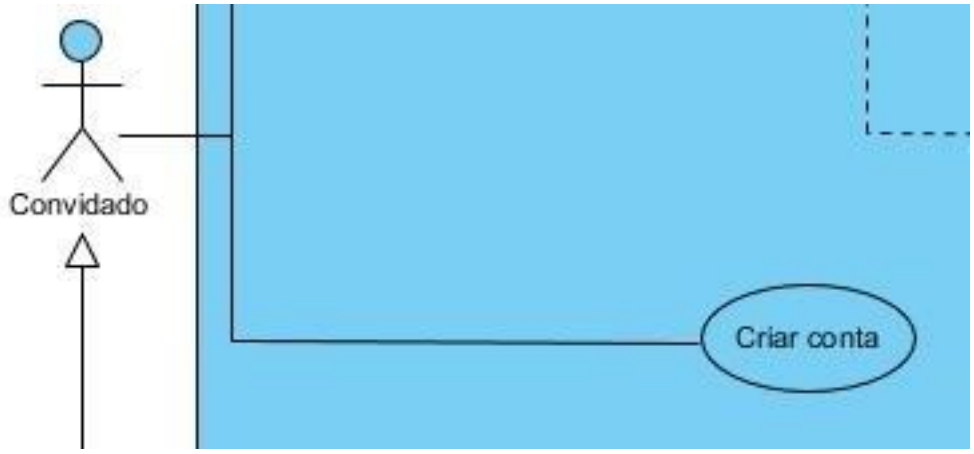

Figura 12 - Caso de utilização "Criar conta" do diagrama de casos de utilização

4.3.9 Pagar lugar de estacionamento

Tabela 15 - Descrição do caso de utilização "Pagar lugar de estacionamento"

Nome:	**Pagar lugar de estacionamento**
Atores:	Utilizador, Utilizador *Premium*, Administrador
Prioridade:	Alto
Finalidade:	Pagar lugar de estacionamento é o ato de pagar um lugar requerendo emissão de fatura
Requisitos Funcionais:	RFU. 6

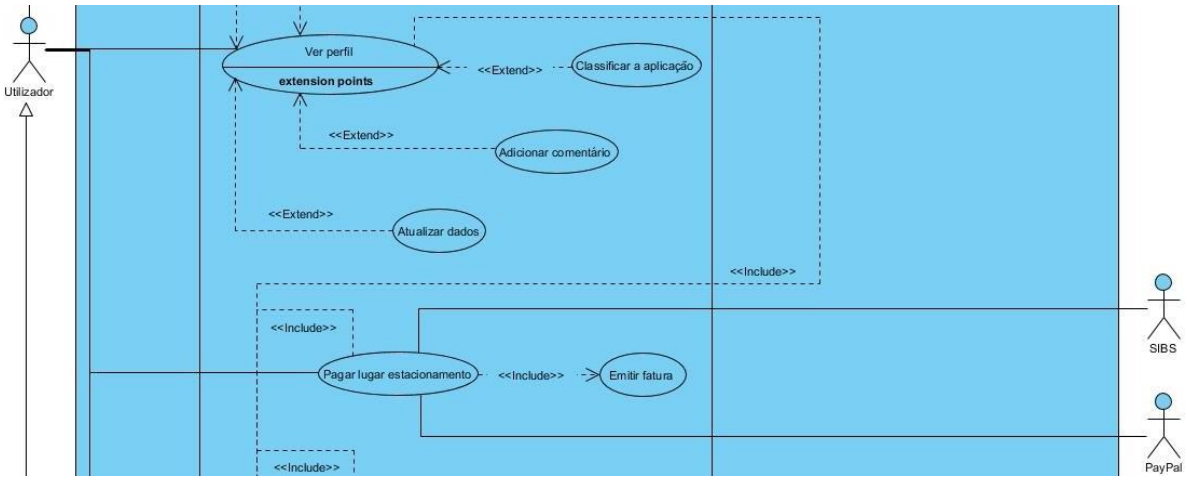

Figura 13 - Caso de utilização "Pagar lugar estacionamento" do diagrama de casos de utilização

4.3.10 Registar matrícula

Tabela 16 - Descrição do caso de utilização "Registar matrícula"

Nome:	**Registar matrícula**
Atores:	Utilizador, Utilizador *Premium*, Administrador
Prioridade:	Médio
Finalidade:	Pode registar mais que uma matrícula que irá estar associada à conta que o denomina
Requisitos Funcionais:	RFU. 10

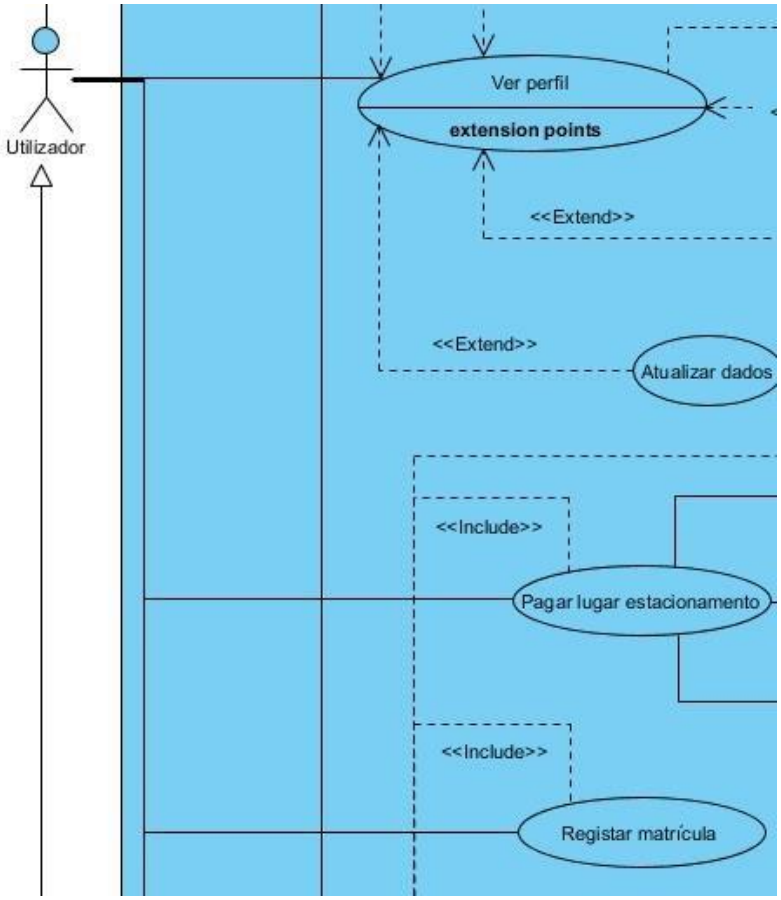

Figura 14 - Caso de utilização "Registar matrícula" do diagrama de casos de utilização

Encontra-se, de seguida, o diagrama de casos de utilização na integral, para cada ator (Convidado – Figura 15, Utilizador Ordinário – Figura 16, Utilizador *Premium* – Figura 17, Administrador – Figura 18) que interage com o sistema, de forma a sustentar o que fora dito anteriormente.

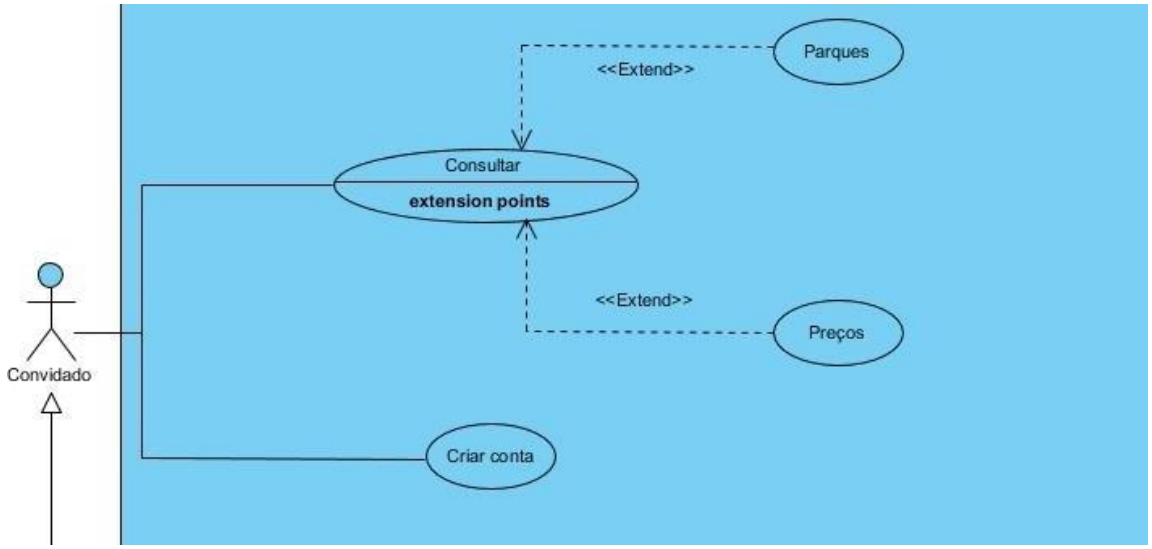

Figura 15 - Casos de utilização do ator Convidado

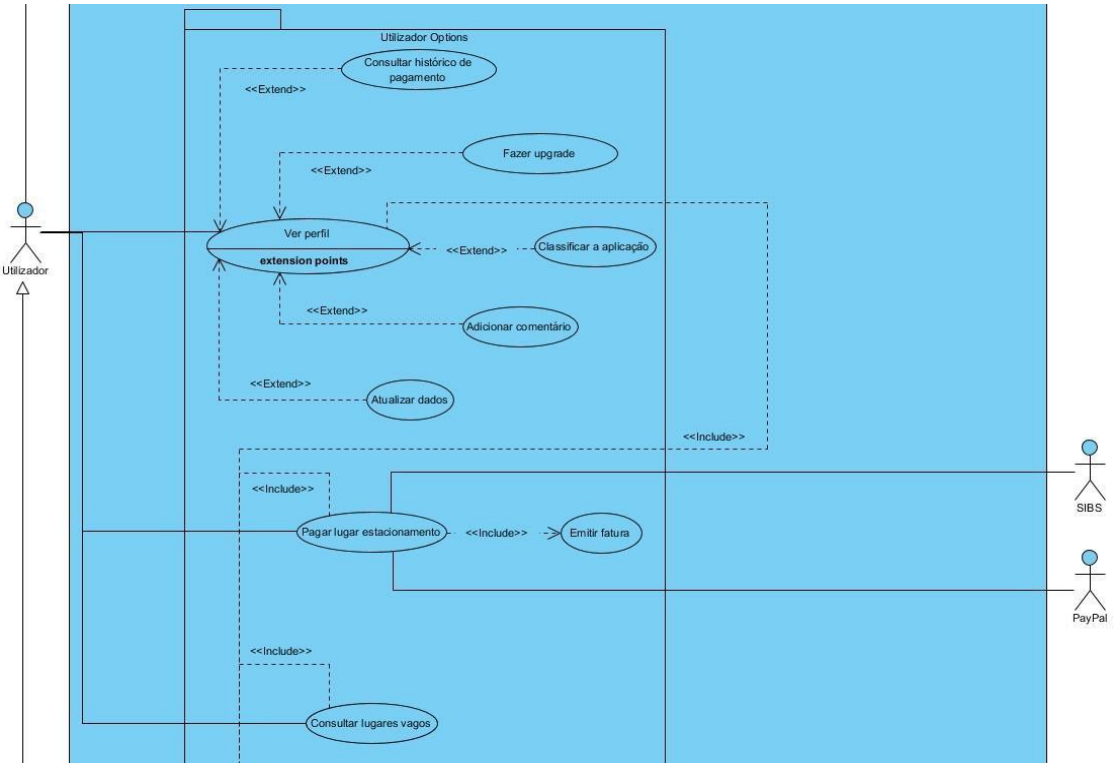

Figura 16 - Casos de utilização dos atores Utilizador Ordinário, SIBS e PayPal

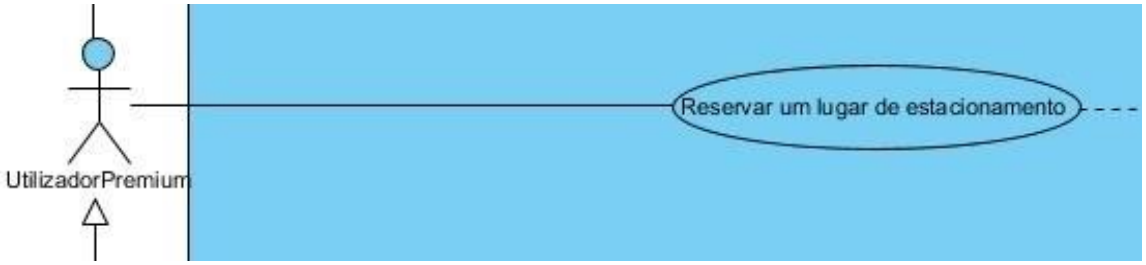

Figura 17 - Casos de utilização do ator Utilizador *Premium*

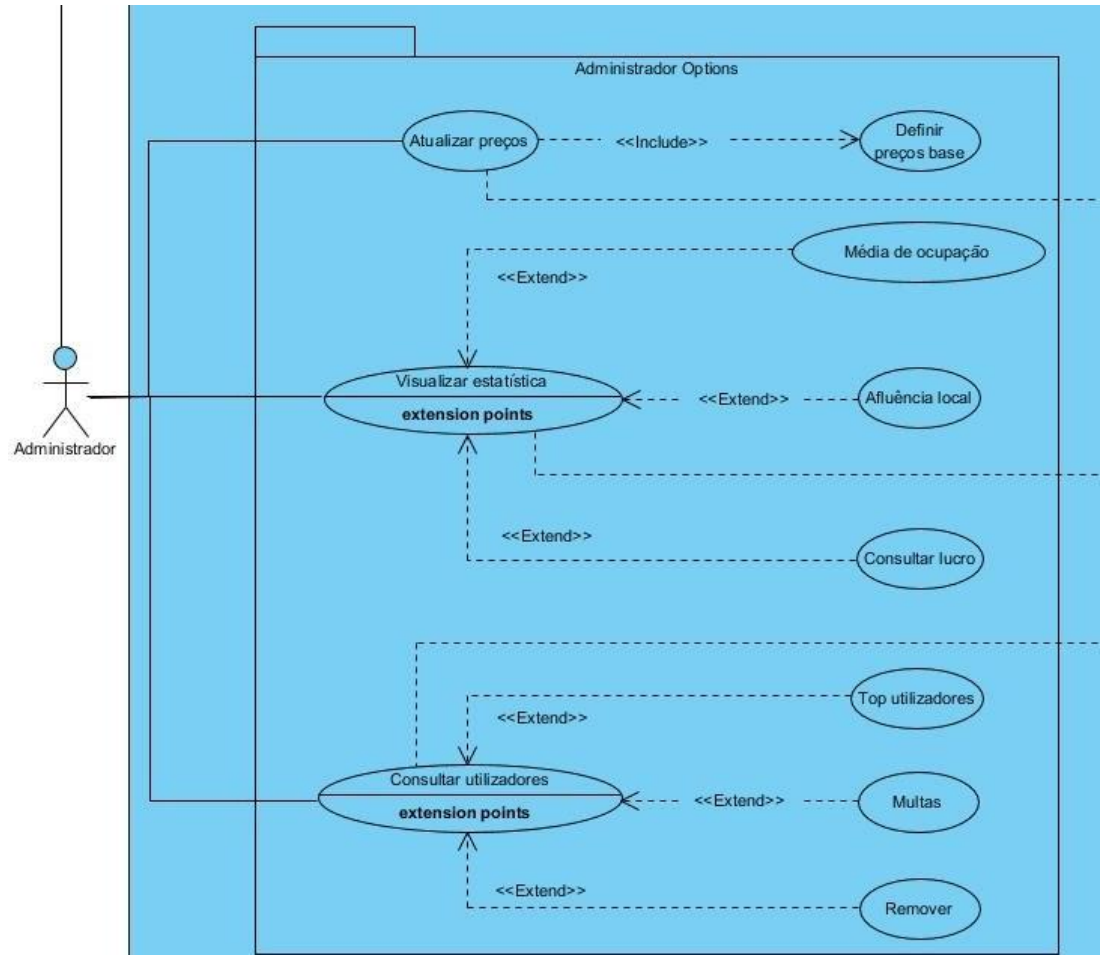

Figura 18 - Casos de utilização do ator Administrador

5 Modelo dinâmico

5.1 Funções de sistema

No tipo de modelação com base num diagrama de sequência, é possível descrever uma sequência particular de funcionamento, em vários ciclos de vida. Imagine-se uma secção de pedidos onde o objetivo é processar uma encomenda de um cliente com a participação de outros sistemas como, por exemplo, Pedido, Linha de Pedidos e Encomenda. Quando o pedido é enviado pelo sistema Secção de Pedidos, é necessário ser tratado pela Linha de Pedidos, que por sua vez cria um sistema temporário (sistema pedido) que trata de verificar se a quantidade pedida existe em stock. Caso exista, cria o sistema Encomenda e processa a encomenda devolvendo uma mensagem de sucesso aos sistemas anteriores. Caso contrário é devolvida uma mensagem de quantidade insuficiente em stock, retornando false. Desta forma, é possível perceber que existe uma linha de vida entre cada mensagem, e que maior parte das mensagens vão depender da reposta de outros sistemas externos, criando assim um ciclo de vida.

Construiu-se um diagrama de sequência para o caso de utilização “pagar lugar de estacionamento” que descreve de uma forma mais detalhada as mensagens que são trocadas na interação com a aplicação e com o método de pagamento escolhido pelo utilizador.

Descrevendo textualmente e como vai ser possível ver pelas figuras abaixo, o ator representado como utilizador abre a aplicação e esta apresenta-lhe um menu exibindo os vários métodos de pagamento, nomeadamente, cartão multibanco ou PayPal. Em seguida, o utilizador escolhe a opção que lhe seja mais conveniente e é criado um sistema, o SIBS, que tem como função verificar o pagamento. Caso o saldo na conta bancária do utilizador seja superior ao valor do estacionamento a pagar, a aplicação emite uma fatura na qual o utilizador confirma o pedido. Caso o valor a pagar pelo estacionamento seja superior ao valor na conta bancária do cliente, a aplicação emite uma multa por falta de pagamento (Figura 19).

Na possibilidade de o utilizador optar por realizar o pagamento por PayPal, o processo é idêntico, isto é, na eventualidade do saldo da conta do utilizador não for suficiente para concretizar o pagamento, é gerada uma multa, caso contrário, é emitida uma fatura na qual o utilizador confirma o pedido (Figura 19).

Para finalizar, o utilizador fecha a aplicação, terminando assim a sua tarefa como se pode ver na figura 20, na mensagem “10. Fecha Aplicação”. Assim que a troca de mensagens termina entre os sistemas SIBS e PayPal, estes são autodestruídos. A figura 21 representa o diagrama na sequência na integra.

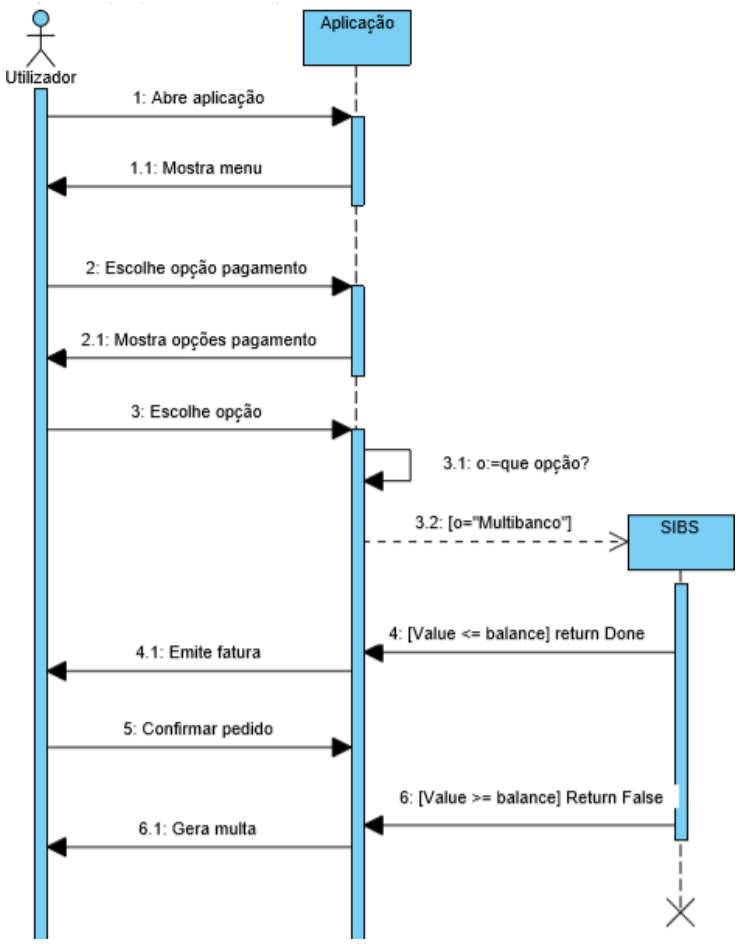

Figura 19 - Sequência de receber fatura ou gerar multa pelo método SIBS

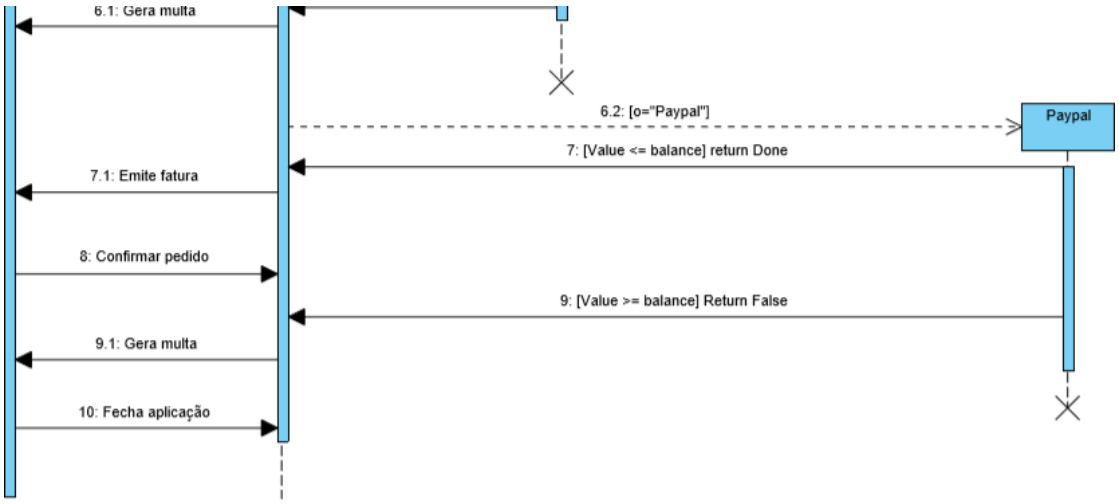

Figura 20 - Sequência de receber fatura ou gerar multa pelo método PayPal

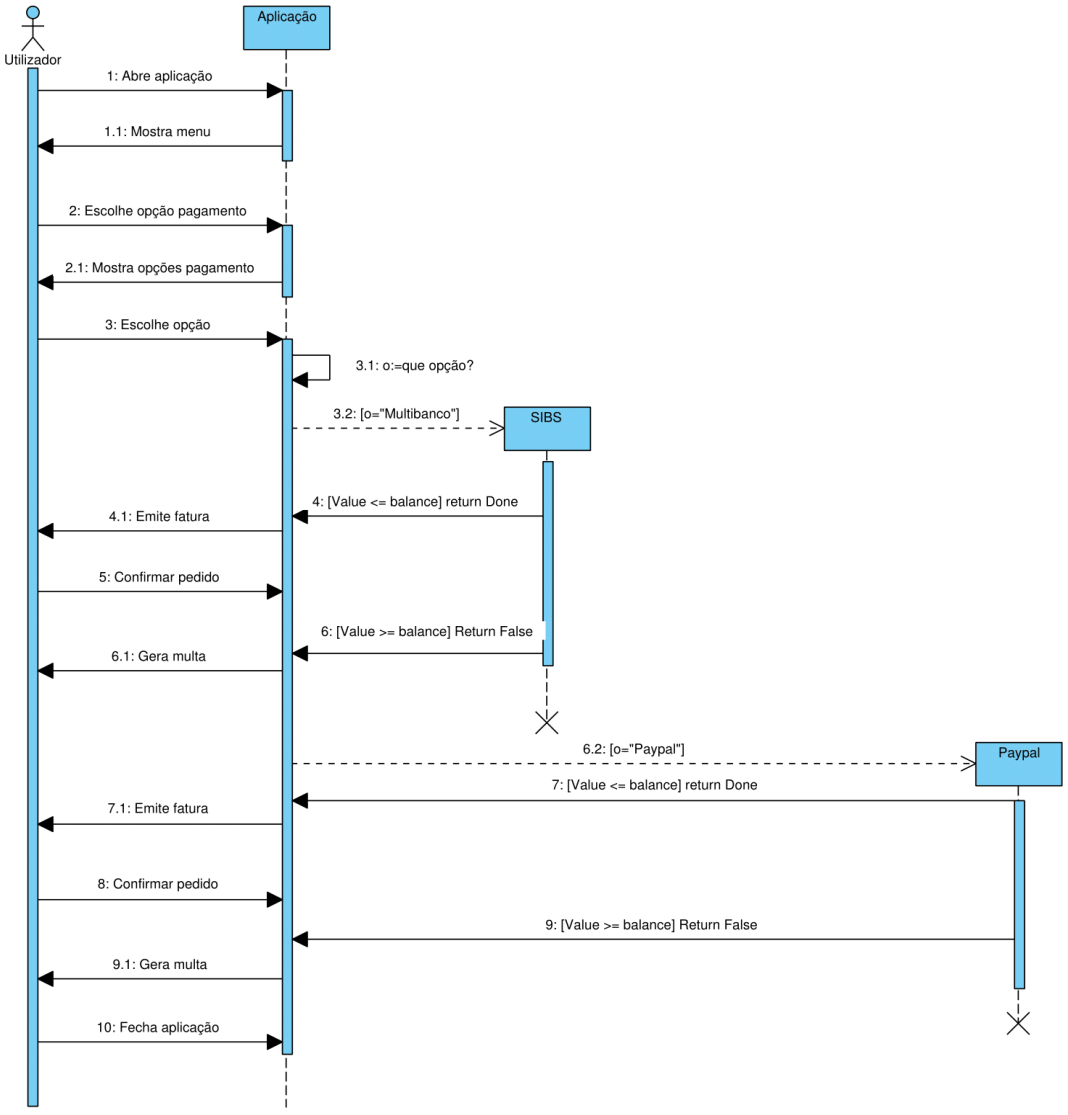

Figura 21 - Diagrama de sequência

6 Modelo estrutural

6.1 Organização da solução

O diagrama de packages é o melhor método quando se está perante um conjunto imenso de diagramas. Assim, por packages há melhor organização devido à possibilidade de agrupar os diagramas por arquivos, sendo mais fácil de consultar (Figura 22). Inclui também o Publish Project do diagrama (Figura 23).

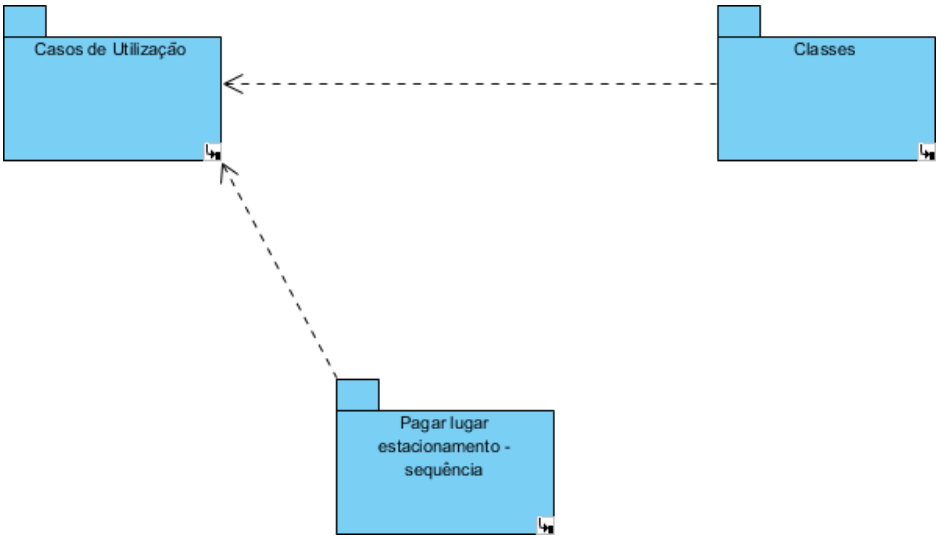

Figura 22 - Diagrama de packages

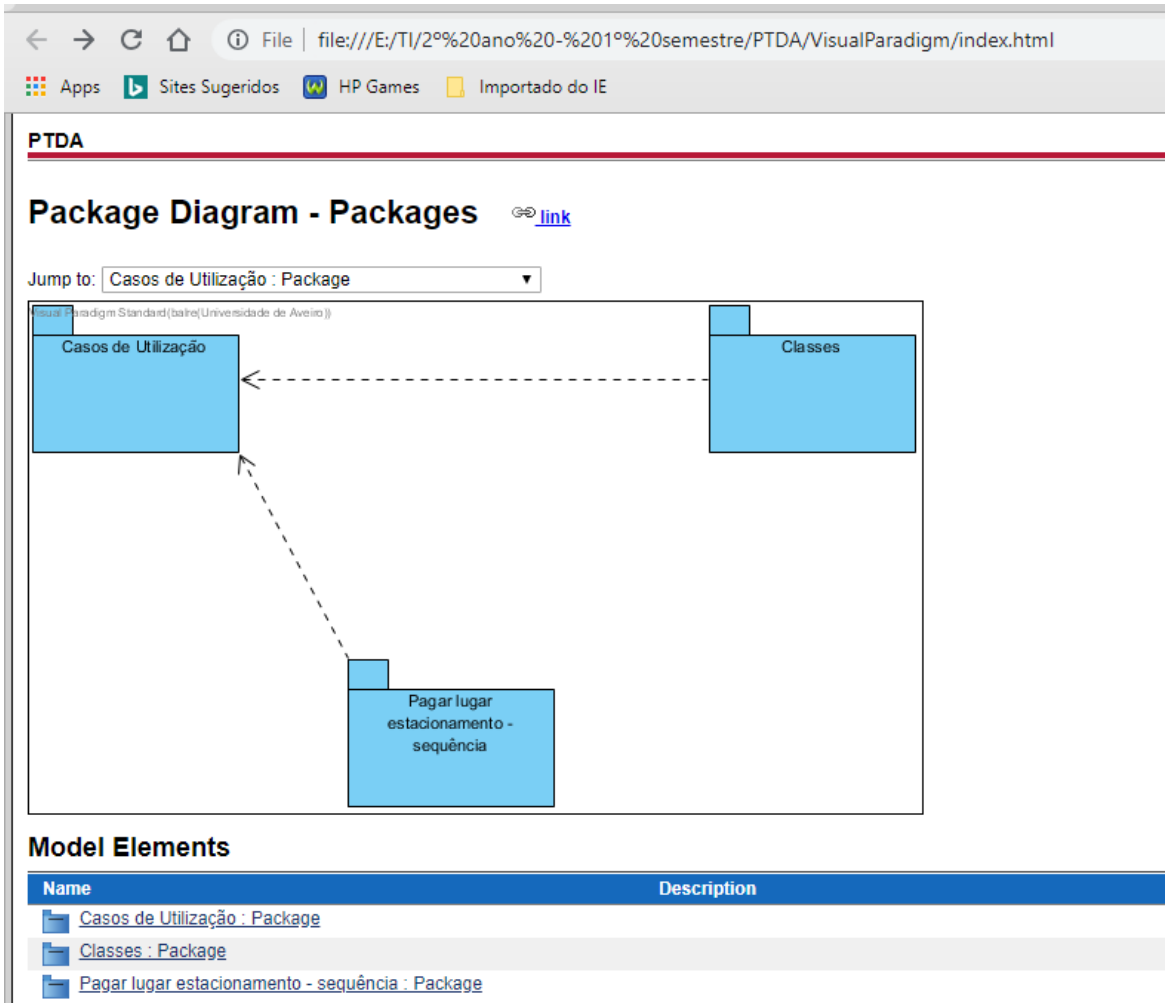

Figura 23 - Publish Project

6.2 Diagrama de classes

Quanto à informação detalhada de cada descritor de um conjunto de objetos que partilham as mesmas propriedades, isto é, de cada classe que está representada pela descrição do sistema, a informação considerada relevante sintetiza-se nos seguintes pontos:

- Os carros em circulação são caracterizados por uma matrícula, marca e modelo;
- Em cada momento, a um veículo pode estar associado, no máximo, uma pessoa e a um único lugar de estacionamento;
- Um estacionamento pode ter vários utilizadores e pode ser consultado por vários convidados;
- Cada zona (zona norte, sul, centro, este, oeste) caracteriza-se por um tipo, um código de identificação e um preço por hora para os utilizadores Ordinários e outro para os utilizadores *Premium* e pode ter vários estacionamentos. Um estacionamento pertence apenas a uma e uma só zona.
- Um utilizador pode realizar várias tarefas na aplicação onde é distinguido na aplicação pelo código de identificação, nome, idade, cartão de cidadão, endereço eletrónico, número de identificação fiscal, morada, data de nascimento, número de cartão para as compras na aplicação e respetivo saldo, código postal, número de telemóvel e palavra-passe;
- Um utilizador Premium pode realizar qualquer funcionalidade desempenhada pelo utilizador ordinário, para além de reservar um lugar de estacionamento, e por sua vez, o administrador pode executar qualquer tarefa incluindo a consulta de informação no sistema;
- Um lugar é definido pelo número e estado (Livre ou Ocupado) e pertence univocamente a um estacionamento, estando associado somente a um veículo;

- A multa regista o motivo da penalização e o código, pertencendo a um único utilizador;

- A ordem de pagamento de um lugar de estacionamento é identificada por um número de ordem, método de pagamento escolhido (Multibanco ou PayPal) e o valor, sendo emitido para um só utilizador.

Segue-se na figura 24 o diagrama de classes tendo em conta a descrição acima. É possível ver pela imagem a associação entre classes, a multiplicidade seguindo o contexto do problema, os *roles* (papéis) que os utilizadores ordinário e *premium* e o administrador terão futuramente pela generalização (herança) e pelas operações.

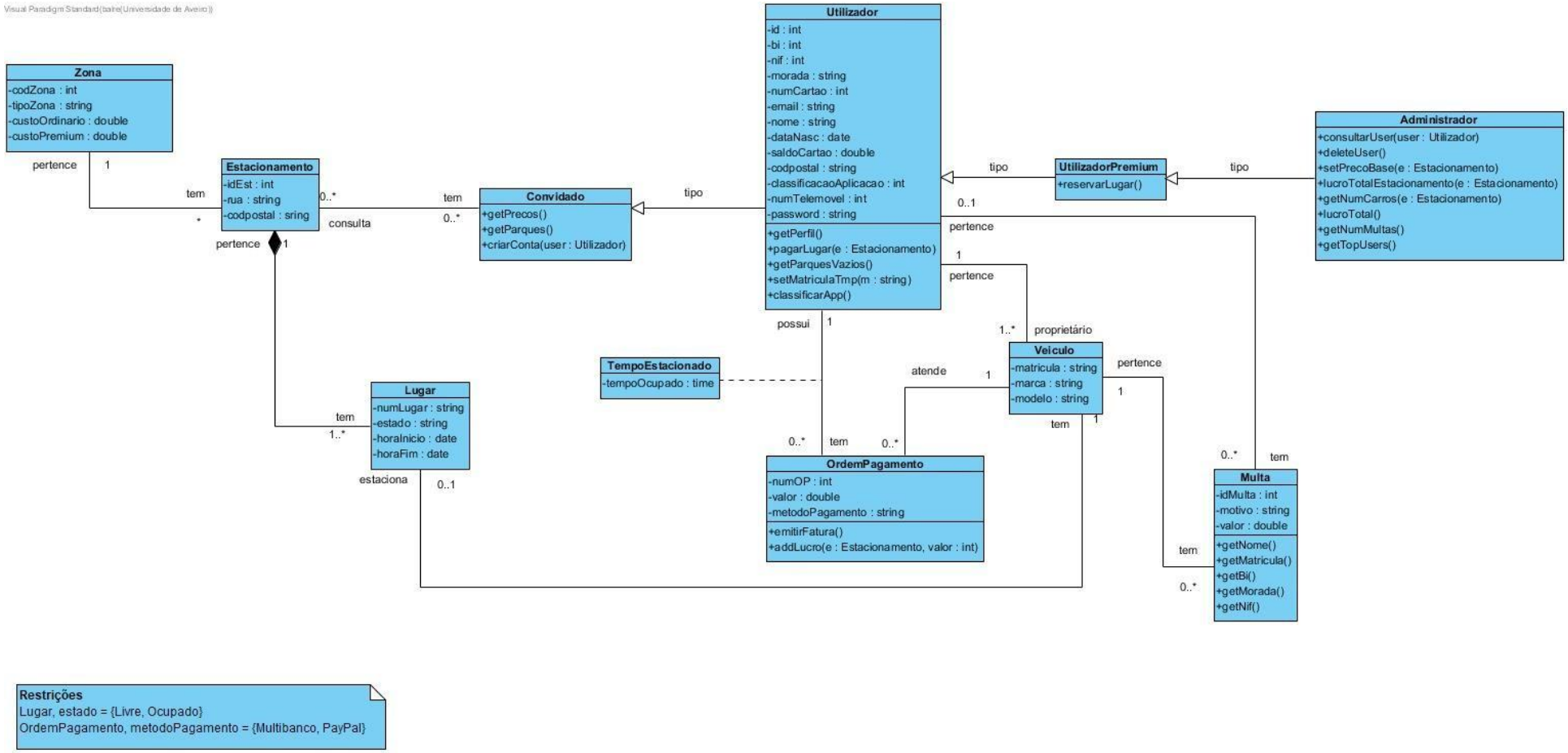

Figura 24 - Diagrama de classes

7 Modelo de dados persistente

7.1 Descrição do problema na ótica da base de dados

Pretende-se criar uma base de dados complementada numa aplicação para gestão do tráfego, referente aos parques de estacionamento de uma dada cidade. O objetivo consiste em adquirir uma ferramenta de gestão para as suas atividades que lhe permita registar o número de pessoas que estaciona num respetivo estacionamento, obtendo o número de lugares ocupados e livres, pela utilização de sensores em cada lugar do parque de estacionamento e, ainda, definir custos variáveis em função do tempo e dos períodos do dia onde se regista uma irregularidade pouco expectável. O sistema a implementar deverá permitir efetuar as seguintes macro operações:

- Permitir registar e visualizar toda a informação de cada utilizador, Ordinário e *Premium*, inclusive ordens de pagamento pela aplicação;
- Multas devido à pessoa não ter dinheiro suficiente no saldo associado ao número de cartão Multibanco ou *PayPal*;
- Sempre que o utilizador procede ao pagamento de um lugar de estacionamento, guardar uma ordem de pagamento com o seguinte: número de ordem, tempo de início e de fim, método de pagamento, valor, identificação do *user* (BI, NIF, ID e matrícula) e estacionamento, zona e lugar respetivo. O processo de pagamento é apenas efetuado na altura de saída do parque de estacionamento onde será necessário ter um procedimento que gera o custo tendo em conta o tempo que a pessoa estacionou o veículo até à saída;
- Permitir registar e visualizar toda a informação atual dos lugares, tais como: número do lugar, estacionamento associado e matrícula do veículo que se encontra estacionado naquele momento

- Segmentar os parques de estacionamento por zonas (zona norte, sul, centro, este e oeste) permite uma melhor consulta por parte dos utilizadores;

- Restringir o acesso aos utilizadores convidados (*guest*) possibilitando apenas a consulta dos parques de estacionamento, preços e o registo de conta;

- Permitir aos utilizadores ordinários a obtenção de uma conta premium pela compra da aplicação ou pelo período de adesão, tendo assim o privilégio de poder reservar um lugar de estacionamento a um preço mais baixo;

- Conceder todas as funcionalidades dos utilizadores ordinário e premium ao administrador, podendo ainda consultar e remover utilizadores, consultar multas por zona ou por um parque em específico, ver estatística (*users* com mais compras na aplicação, lucro total ou número de veículos por zona/parque) e atualizar preços.

Depois de desenvolver o diagrama de classes, é necessário concedê-lo para um modelo de dados relacional. Para isso, existe um conjunto de regras que é necessário aplicar, dependendo do tipo de associação (“Um para Muitos”, “Um para Um” ou “Muitos para Muitos”) e de transposição (composições, agregações, generalizações ou de classes associativas).

O texto seguinte representa a descrição das regras de transposição e a sua aplicação tendo em conta a ligação e multiplicidade existente nas classes do diagrama de classes descrito anteriormente:

Regras de Transposição:

- Regra 1: Chave primária

Todas as tabelas devem possuir uma chave primária. Caso não existam atributos que satisfaçam esta condição, então deve ser criado um atributo para o efeito (p.ex., “id”).

- Regra 2: Origem das tabelas

As tabelas derivam somente das classes do diagrama de classes e das associações de “muitos para muitos”

- Regra 3: Associação de “Um para Um”

Uma das tabelas deverá receber como chave estrangeira a chave primária da outra tabela. Neste caso, recebe a tabela em que faça mais sentido estar o referido atributo.

- Regra 4: Associação de “Um para Muitos”

A tabela em que a informação será repetida é que recebe a chave estrangeira. Ou seja, a parte do “muitos” é que recebe a chave estrangeira.

- Regra 5: Associação de “Muitos para Muitos”

A transição dá origem a uma terceira tabela que representa a associação e cuja chave primária é composta pelas chaves primárias das tabelas associadas.

- Regra 6: Transposição de classes associativas

Utiliza-se uma das regras correspondentes às associações. Neste caso, os atributos da classe associativa são recebidos pela tabela que recebe as chaves.

- Regra 7: Transposição de generalizações

Esta transposição varia consoante a natureza da identidade das subclasses, resumindo-se a 3 soluções:

1. Esmagamento das classes da hierarquia num único esquema relacional correspondente à superclasse
2. Se as subclasses possuem identidade própria independente da superclasse: considerar apenas esquemas correspondentes às subclasses
3. Considerar todas as classes da hierarquia (se as subclasses só têm identidade própria quando associadas à superclasse)

- Regra 8: Transposição de agregações

Esta transposição utiliza as mesmas regras de transposição para associações com a mesma multiplicidade.

- Regra 9: Transposição de composições

A tabela que equivale à classe de composição fica com a chave estrangeira. Esta chave também fará parte da sua chave primária.

Nota: PK – Primary Key **FK** – Foreign Key

Regra 4: Associação de "Um para Muitos" entre as classes Zona e Estacionamento

Zona (codzona (PK), tipozona, custoordinario, custopremium)

Estacionamento (idest (PK), rua, codpostal, codzona (FK))

Regra 9: Transposição de Composições entre as classes Estacionamento e Lugar

Estacionamento (idest (PK), rua, codpostal, codzona (FK))

Lugar (numlugar (PK), estado, horainicio, horafim, idest (PK) (FK))

Regra 5: Associação de “Muitos para Muitos” (entre as classes Estacionamento e Utilizador)

Utilizador (userid (PK), bi (PK), nif (PK), morada, numcartao, email, nome, datanasc, saldocartao, codpostal, classificacaoapp, numtelemovel, password)

Estacionamento (idest (PK), rua, codpostal, codzona (FK))

UtilizadorEstacionamento (userid (PK) (FK), bi (PK) (FK), nif (PK) (FK), idestacionamento (PK) (FK))

Regra 4: Associação de “Um para Muitos” entre as classes Utilizador e Veiculo

Utilizador (userid (PK), bi (PK), nif (PK), morada, numcartao, email, nome, datanasc, saldocartao, codpostal, classificacaoapp, numtelemovel, password, tipo)

Veiculo (matricula (PK), marca, modelo, userid (FK), bi (FK), nif (FK))

Regra 6: Transposição de classes associativas entre as classes Utilizador, TempoEstacionado e OrdemPagamento

Associação "Um para Muitos"

Utilizador (userid (PK), bi (PK), nif (PK), morada, numcartao, email, nome, datanasc, saldocartao, codpostal, classificacaoapp, numtelemovel, password, tipo)

OrdemPagamento (numop (PK), valor, metodopagamento, userid (FK), bi (FK), nif (FK))

TempoEstacionado (numop (PK) (FK), userid (PK) (FK), bi (PK) (FK), nif (PK) (FK), tempoocupado)

Regra 3: Associação de “Um para Um” entre as classes Veiculo e OrdemPagamento

Veiculo (matricula (PK), marca, modelo, userid (FK), bi (FK), nif (FK))

OrdemPagamento (numop (PK), valor, metodopagamento, userid (FK), bi (FK), nif (FK), matricula (FK))

Regra 4: Associação de “Um para Muitos” entre as classes Veiculo e Multa

Veiculo (matricula (PK), marca, modelo, userid (FK), bi (FK), nif (FK))

Multa (idmulta (PK), motivo, matricula (FK))

Regra 4: Associação de “Um para Muitos” entre as classes Multa e Utilizador

Utilizador (userid (PK), bi (PK), nif (PK), morada, numcartao, email, nome, datanasc, saldocartao, codpostal, classificacaoapp, numtelemovel, password, tipo)

Multa (idmulta (PK), motivo, matricula (FK), userid (FK), bi (FK), nif (FK))

Regra 3: Associação de “Um para Um” entre as classes Veiculo e Lugar

Veiculo (matricula (PK), marca, modelo, userid (FK), bi (FK), nif (FK))

Lugar (numlugar (PK), estado, horainicio, horafim, idest (PK) (FK), matricula (FK))

Depois de aplicadas as regras referidas anteriormente, interessa agora modelar cada classe num modelo de dados persistentes para depois se implementar num sistema de gestão de base dados. A figura 25 esquematiza as classes acima com as chaves primárias e chaves estrangeiras definidas por um modelo de dados persistentes. Apresenta-se, de forma integral, o resultado das etapas realizadas pelas regras de transposição:

Zona (codzona (PK), tipozona, custoordinario, custopremium)

Estacionamento (idest (PK), rua, codpostal, codzona (FK))

Lugar (numlugar (PK), estado, horainicio, horafim, idest (PK) (FK), matricula (FK))

Utilizador (userid (PK), bi (PK), nif (PK), morada, numcartao, email, nome, datanasc, saldocartao, codpostal, classificacaoapp, numtelemovel, password, tipo)

Veiculo (matricula (PK), marca, modelo, userid (FK), bi (FK), nif (FK))

Multa (idmulta (PK), motivo, matricula (FK), userid (FK), bi (FK), nif (FK))

UtilizadorEstacionamento (userid (PK) (FK), bi (PK) (FK), nif (PK) (FK), idestacionamento (PK) (FK))

OrdemPagamento (numop (PK), valor, metodopagamento, userid (FK), bi (FK), nif (FK), matricula (FK))

TempoEstacionado (numop (PK) (FK), userid (PK) (FK), bi (PK) (FK), nif (PK) (FK), tempoocupado)

De realçar que as classes Convidado e UtilizadorPremium foram retiradas ao obedecer às regras de transposição devido à inutilidade que iriam ter, na perspetiva do modelo de dados persistentes, visto que no modelo lógico armazenam-se apenas os dados persistentes. Optou-se então por remover a classe Convidado e por utilizar um atributo “tipo” na classe Utilizador para ser mais fácil distinguir entre o utilizador ordinário e o utilizador *premium*. A classe Administrador também foi excluída pelo facto de na base de dados ser possível definir o papel a desempenhar por um utilizador, ou seja, que restrições e privilégios vai ter na base de dados (*Grant permission*), e assim, este já pode alterar, obter e eliminar dados das tabelas.

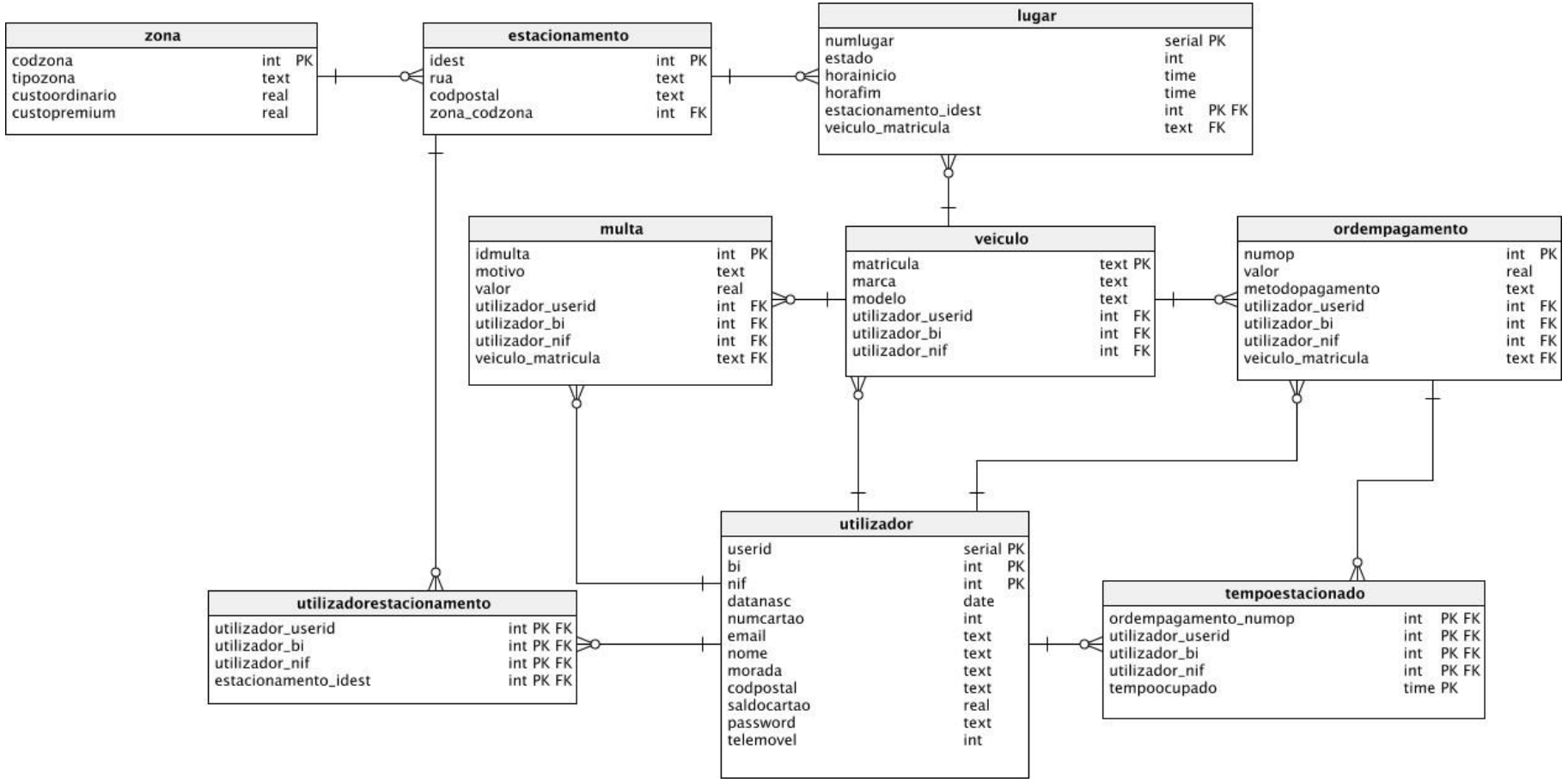

Figura 25 - Modelo de dados persistentes obedecendo às regras de transposição

A descrição seguinte consolida o processo de normalização da base de dados de forma a otimizar e a garantir que as várias estruturas da base de dados são eficientes na representação da informação, prevenindo ainda a perda de informação e a minimização/eliminação da redundância.

Ao obedecer ao conjunto de regras que envolve a normalização e ao tipo de contexto que existe neste sistema a desenvolver, é de realçar que vão existir bastantes associações ou relações entre os objetos (tabelas). Imagine-se que uma pessoa estaciona o veículo num determinado lugar de um estacionamento e, após o período que a pessoa utilizou aquele lugar, pretende sair desse mesmo estacionamento. Para o devido efeito, o sistema irá calcular o custo do tempo que essa pessoa esteve a usufruir do lugar, com a devida informação do tempo inicial e final e da matrícula associada, caso a mesma esteja armazenada na base de dados, obtendo assim o custo final. Desta forma, a base de dados deve, obrigatoriamente, guardar toda a informação do utilizador na qual a matrícula esteve associada e ainda o lugar, estacionamento e zona respetiva para fins estatísticos e de consistência da aplicação. É possível deduzir então que algumas das tabelas das regras de transposição vão sofrer alterações devido à pobre informação que armazenam.

Sendo este um processo evolutivo que apenas floresce e ganha firmeza no decorrer do projeto, inferiu-se que, para a maior parte das tabelas no modelo de dados persistentes, a inclusão de campos e a disposição de alguns dos mesmos vai existir, mesmo após a aplicação das regras da normalização.

O texto seguinte descreve cada uma das regras pertencentes à teoria da normalização:

1ª Forma Normal

- Não deve conter atributos multivalor;
- Valores devem ser do mesmo tipo de dados pedido no atributo respetivo;
- Atributos devem ter um nome único;
- Não há uma ordem específica nos tuplos.

2ª Forma Normal

- Deve estar na 1ª Forma Normal;
- Não deve haver dependência parcial (os atributos que não pertencem a uma chave candidata devem depender da mesma na totalidade e não parcialmente).

3ª Forma Normal

- Deve estar na 2ª Forma Normal;
- Um atributo não-chave não pode depender de outro atributo não chave. Tem de depender de um atributo que seja chave primária ou chave candidata.

Forma normal Boyce-Codd

- Deve estar na 3ª Forma Normal;

- And, for any dependency A → B, A should be a super key.

Um esquema relacional equilibrado fica normalmente entre a 3ª FN e a FNBC. Pelas regras acima, é possível observar que todas as tabelas cumprem a 1ª Forma Normal. Não existem atributos multivalor, ou seja, cada campo vai ter um único valor associado e os nomes são distinguíveis e únicos. A ordem das colunas é ignorada, ainda que seja conveniente ter as chaves primárias como antecessoras e as chaves estrangeiras como sucessoras de todos os restantes campos. Quanto à 2ª Forma Normal, todos os campos que não pertencem à chave candidata dependem da totalidade da mesma e o mesmo acontece na 3ª Forma Normal, onde os atributos não-chave dependem sempre uma chave candidata ou primária.

Tal como fora referido anteriormente, relativamente a existirem alterações nas tabelas, sentiu-se no dever de adicionar mais atributos de forma a ter mais informação quando se faz uma pesquisa do número de multas nas últimas 8 horas, por exemplo, e também para haver uma maior organização nos registos (por exemplo: os campos telemóvel e password ficam numa tabela e os restantes dados do utilizador ficam noutra). As tabelas seguintes conferem essas alterações e demonstram a evolução que houve do primeiro para o segundo modelo de dados persistentes (Figura 26).

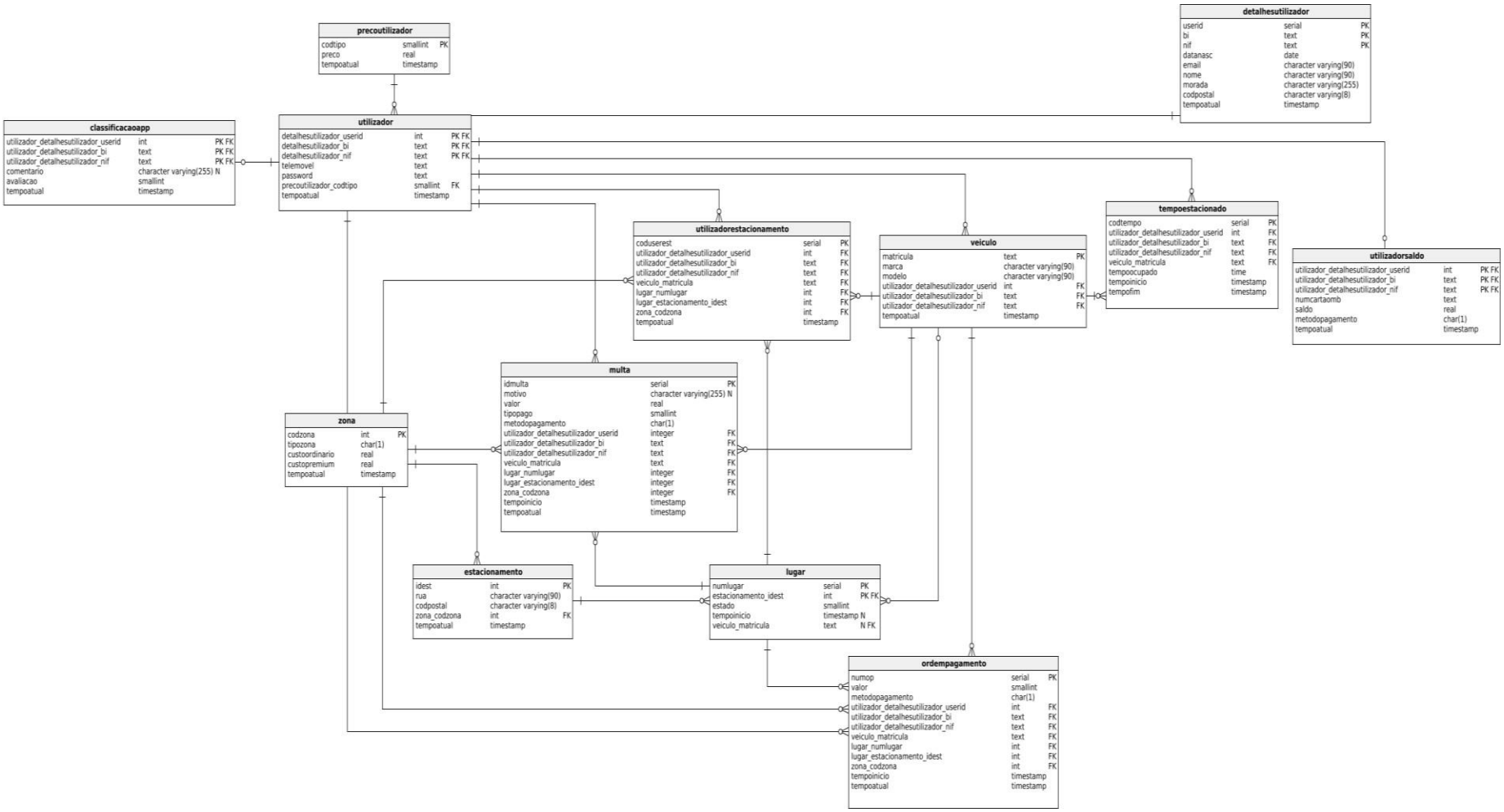

Figura 26 - Modelo de dados persistentes com otimização

A figura 25 demonstra agora que o modelo desenvolvido se aproxima mais da realidade do sistema, onde se regista toda a informação que é necessária sobre o utilizador, zona, estacionamento e lugar de estacionamento, bem como o tempo atual. As tabelas e campos respetivos da figura 26 encontram-se descritos abaixo:

Nota: N – Null

detalhesutilizador (userid (PK), bi (PK), nif (PK), datanasc, email, nome, morada, codpostal, tempoatual)

classificacaoapp (utilizador_detalhesutilizador_userid (PK) (FK), utilizador_detalhesutilizador_bi (PK) (FK), utilizador_detalhesutilizador_nif (PK) (FK), comentario (N), avaliacao, tempoatual)

precoutilizador (codtipo (PK), preco, tempoatual)

utilizador (detalhes_utilizador_userid (PK) (FK), detalhes_utilizador_bi (PK) (FK), detalhes_utilizador_nif (PK) (FK), telemovel, password, precoutilizador_codtipo (FK), tempoatual)

zona (codzona (PK), tipozona, custoordinario, custopremium, tempoatual)

estacionamento (idest (PK), rua, codpostal, zona_codzona (FK), tempoatual)

lugar (numlugar (PK), estacionamento_idest (PK) (FK), estado, tempoinicio **N**, veiculo_matricula (FK) **N**)

veiculo (matricula (PK), marca, modelo, utilizador_detalhesutilizador_userid (FK), utilizador_detalhesutilizador_bi (FK), utilizador_detalhesutilizador_nif (FK), tempoatual)

ordempagamento (numop (PK), valor, metodopagamento, utilizador_detalhesutilizador_userid (FK), utilizador_detalhesutilizador_bi (FK), utilizador_detalhesutilizador_nif (FK), veiculo_matricula (FK), lugar_numlugar (FK), lugar_estacionamento_idest (FK), zona_codzona (FK), tempoinicio, tempoatual)

multa (idmulta (PK), motivo **N**, valor, tipopago, metodopagamento, utilizador_detalhesutilizador_userid (FK), utilizador_detalhesutilizador_bi (FK), utilizador_detalhesutilizador_nif (FK), veiculo_matricula (FK), lugar_numlugar (FK), lugar_estacionamento_idest (FK), zona_codzona (FK), tempoinicio, tempoatual)

utilizadorestacionamento (coduserest (PK), utilizador_detalhesutilizador_userid (FK), utilizador_detalhesutilizador_bi (FK), utilizador_detalhesutilizador_nif (FK), veiculo_matricula (FK), lugar_numlugar (FK), lugar_estacionamento_idest (FK), zona_codzona (FK), tempoatual)

tempoestacionado (codtempo (PK), utilizador_detalhesutilizador_userid (FK), utilizador_detalhesutilizador_bi (FK), utilizador_detalhesutilizador_nif (FK), veiculo_matricula (FK), tempoocupado, tempoinicio, tempofim)

utilizadorsaldo (utilizador_detalhesutilizador_userid (PK) (FK), utilizador_detalhesutilizador_bi (PK) (FK), utilizador_detalhesutilizador_nif (PK) (FK), numcartaomb, saldo, metodopagamento, tempoatual)

Optou-se por retirar as *constraints* das chaves primárias da tabela utilizadorestacionamento pelo facto de que, caso haja dois veículos estacionados em qualquer estacionamento cujas matrículas estão associadas ao mesmo utilizador com conta registada na aplicação, já não será possível ter campos repetidos devido às PK's. Uma vantagem é que, após o utilizador abandonar o lugar respetivo, esse registo é apagado da tabela utilizadorestacionamento, como se verá mais à frente, com a utilização de *triggers*.

8 Estratégia de desenvolvimento

Relativamente ao processo de desenvolvimento de software, adotou-se uma estratégia que consiste em distribuir tarefas pelos elementos do grupo. Define-se os requisitos funcionais e não funcionais que vão ditar o que incorpora o funcionamento do sistema. Obviamente que os requisitos vão estar constantemente numa fase de alteração, isto é, não são completamente definidos à partida. Para contornar essa situação ou evitar que surjam problemas devido à falta de comunicação, tendo em conta que os requisitos não são explicitamente especificados na fase inicial, optou-se por dividir os requisitos por dois subgrupos. Uma parte do grupo comprometia-se a desenvolver a parte do administrador e todas as funcionalidades a que este pode usufruir e ainda a gestão da Base de Dados e a outra responsabilizava-se do utilizador.

Em relação à Base de Dados era necessário implementar o modelo que obedecesse às regras da normalização (incluindo as regras de transposição) e definir os comportamentos que esta terá de possuir quando há uma inserção, atualização ou eliminação de um dado tuplo ou valor de uma coluna. Para o efeito, é necessário conceder validações que tornem a Base de Dados consistente e fiável, caso aconteça algo de errado no processamento de dados e nas interfaces. Se for possível criar mecanismos que acionem quando algo de anormal aconteça ao aceder à Base de Dados, devem ser implementados. Outro aspeto é o tempo de resposta. Se a Base de Dados não é rápida a responder a uma determinada ação (selecionar dados), é inútil o que é feito posteriormente. Para isso, despende de tempo de pesquisa e procura na melhor solução. Desta forma, estipulou-se que um subgrupo do grupo tratava da gestão e manutenção da Base de Dados e das funcionalidades do administrador, e os restantes do utilizador, o que ia envolver a validação e funções da criptografia dos dados (Hash – mistura). Ao fim de um determinado tempo, ambos os subgrupos submetem os excertos do código que está funcional no repositório para depois no final poderem interligar cada um deles. Cada elemento do subgrupo encarrega-se de uma pequena parte da tarefa integral e faz uma pesquisa intensiva de como a realizar, tanto do lado da Base de Dados, como do lado processamento de dados e das interfaces.

Como há a possibilidade de integrar o processamento de inserção, seleção, atualização e eliminação dos dados persistentes das tabelas na própria Base de Dados por meio de funções em SQL, PL/SQL ou PL/pgSQL, triggers e stored procedures, é importante debater na melhor metodologia de implementação da gestão dos dados – se se utiliza as SQL *statements* do lado da Base de Dados ou no próprio Java.

Aplicando esta metodologia, o *feedback* é positivo no decorrer do projeto. Há uma revisão constante do código e não condiciona o código quando há alteração ou se acrescenta requisitos. Qualquer alteração nos requisitos, pode-se facilmente adaptar o código atual.

Tendo em conta o modelo de processo utilizado, o que se adequa e se identifica mais com os parâmetros seguidos no desenvolvimento do projeto é o processo ágil eXtreme Programming. Isto porque consiste no trabalho em equipa e na integração contínua dos elementos do grupo no trabalho. Apesar do desenvolvimento ter consistido de uma forma individual, onde cada elemento do subgrupo desempenhou uma determinada tarefa, o que é importante adquirir domínio total no tema, questões lógicas e técnicas (instruções da linguagem de programação) eram esclarecidas com o resto dos elementos.

Parte-se assim do pressuposto que para se obter um bom resultado é necessário estabelecer uma boa dinâmica do grupo. E isto adquire-se definindo o seguinte, como já fora dito anteriormente: divide-se tarefas pelos elementos do grupo estando estes, por sua vez, agrupados em subgrupos, tendo de haver comunicação, não só pelos subgrupos, mas também em todo o grupo.

9 Arquitetura em 3 camadas

A arquitetura de 3 camadas é um tipo de arquitetura de software composta por três camadas de computação lógica, sendo utilizada maioritariamente em aplicações do tipo cliente-servidor.

Esta arquitetura apresenta muitos benefícios de desenvolvimento pois é modulada a camada de interface do utilizador (user interface), a camada de processamento de dados (business logic) e a camada do servidor, isto é, a base de dados (data storage). A grande vantagem de utilizar esta arquitetura deve-se ao facto de ser possível atualizar e melhorar partes específicas de uma determinada camada de forma independente sem afetar as outras camadas.

9.1 As 3 camadas

A camada Presentation Tier, ou GUI (Graphical User Interface) é a camada que interage diretamente com o cliente a partir de uma interface gráfica.

É nesta camada que é apresentado todo o conteúdo e informação importante ao cliente. É nesta camada que o utilizador pode fazer consultas, requisitos e alteração de dados próprios;

A camada Application Tier, ou business logic é a camada que contém todo o processamento de dados.

É nesta camada que estão armazenadas toda a lógica funcional da aplicação, isto é, é nesta camada que estão contidos todas as funções e regras da aplicação. Esta camada não apresenta nenhuma interface para o utilizador e os seus dados são voláteis, isto é, caso a aplicação seja terminada, os dados contidos serão perdidos e, para isso, é necessária uma terceira camada;

A camada Data Tier ou data storage é a camada composta pela Base de Dados.

É nesta camada que é criada a Base de Dados onde todos os dados são armazenados e onde é possível aceder e alterar a estes através de API's.

No projeto atualmente a ser desenvolvido é possível evidenciar as várias camadas que foram desenvolvidas.

Na camada que interage com o utilizador, o Presentation Tier, é apresentada uma interface onde o utilizador se pode registar, alterar os seus dados pessoais, obter toda a informação sobre os vários estacionamentos presentes na cidade e onde pode atualizar a sua conta de modo a poder ter mais vantagens dentro da aplicação. Esta interface pode ser também utilizada pelo administrador de modo a obter estatísticas sobre os estacionamentos, estatísticas da própria aplicação tal como a sua classificação e tem acesso a rankings.

Na camada de processamento de dados, Application Tier, são desenvolvidas todas as funções e regras da aplicação em java, nomeadamente verificação de dados presentes no registo e no início de sessão, encriptação de dados pessoais dos utilizadores, conexão com a Base de Dados.

Sempre que exista uma seleção na interface, tal como a consulta dos estacionamentos e os seus preços, esta é processada na camada de processamento de dados e posteriormente é esta a camada que faz a query à Base de Dados e após obter a resposta do servidor, esta é apresentada na interface ao utilizador.

A camada Data Tier é a camada onde está presente a Base de Dados e onde estão armazenados todos os dados importantes. É nesta camada que são feitas querys pela camada de processamento de dados em resposta ao utilizador.

De forma a respeitar este conceito, definiram-se packages que praticam essas três camadas. Criaram-se packages que têm somente as interfaces, isto é, que permite que o user interaja com o sistema, e ainda o processamento de dados cujas interfaces vão invocar os métodos utilizados na lógica da aplicação.

10 Geração de Dados Aleatórios

Para simular os vários utilizadores que iriam utilizar a aplicação, foi criado um package para gerar dados aleatoriamente.

Para isto foi criada uma classe correspondente a cada uma das tabelas a serem preenchidas na base de dados e, dentro de cada classe foi criado um método de modo a preencher cada uma das tabelas com valores aleatórios.

No final foi criada uma classe main de modo a invocar os métodos das classes previamente feitas preenchendo assim todas as tabelas presentes na base de dados.

11 Criptografia

De maneira a que exista segurança na base de dados com dados mais sensíveis do utilizador, tais como o número de BI e NIF, foi implementado um método que faz a encriptação de dados quando se insere um registo de um novo utilizador.

O Hashing é o processo de criação de uma String a partir de uma mensagem fornecida, utilizando uma função matemática conhecida por *cryptographic hash function.*

Para uma encriptação segura, a função utilizada deve conter quatro propriedades apresentadas seguidamente:

- Não é reversível, isto é, é difícil gerar uma mensagem a partir do seu hash;
- Tem uma elevada entropia, isto é, uma pequena mudança na mensagem produz um hash bastante diferente;
- É resistente a colisões, isto é, duas mensagens diferentes não produzem o mesmo hash;
- A mensagem processada pelo mesmo método de hashing deve produzir sempre no mesmo hash;

Para esta encriptação foi criado um método a que se deu o nome de “HashData” com a utilização de uma função hash denominada de PBKDF2, pois esta vai de encontro com as quatro propriedades necessárias para a encriptação de dados.

Para a autenticação, ao ser feito o início de sessão, é feita também uma leitura da mensagem, uma encriptação da mesma e, caso as duas hash sejam iguais, é confirmada a sua autenticação.

Para o caso das matrículas utilizou-se outro tipo de algoritmo, AES, que trata de encriptar e decriptar esses mesmos dados. Isto foi útil para a funcionalidade gerir matrículas pois requeria que a matrícula fosse omitida mas que seja reversível.

12 Instalação

12.1 Base de Dados

Uma das coisas que é importante que uma Base de Dados possua é um rápido tempo de resposta. Um utilizador rejeita de imediato uma aplicação se despende de mais de 5 segundos para lhe retornar o resultado. Isso deve-se ao facto de a Base de Dados não conter mecanismos que tornem a pesquisa nas tabelas rápida e consistente (exemplo: Índices, relação entre Java e Base de Dados). Desta forma tentou-se criar formas para conceder uma rápida resposta por parte das tabelas, e para isso criaram-se índices.

Imagine-se um livro. Para uma pessoa poder encontrar uma referência ou um capítulo numa página é ridículo ter de folhear todas as páginas para poder achar o pretendido. Para isso, usam-se os índices que indicam a página correspondente aos capítulos e subtítulos. Nos índices da Base de Dados, a analogia é idêntica. Os índices são criados tendo em conta um campo da tabela específico. Um ponto a ter em conta é que isto é aconselhável usar apenas em determinadas situações: tabelas com uma grande quantidade de dados e com operações frequentes, colunas que não contenham muitos valores NULL e que não sejam constantemente manipuladas. Fez-se então um índice para a tabela estacionamento pois era a única que cumpria as regras descritas acima.

No caso da verificação dos dados, utilizaram-se Domains e Check constraints que permitem validar os dados antes da sua inserção nas tabelas. Para o caso criou-se para o código-postal, método de pagamento (m ou M – MultiBanco e p ou P – PayPal), custos e valores entre 0 e 1. Para o caso do código-postal este deve cumprir as normas de um código-postal válido (entre 1000-100 e 9999-999). Nos custos, tais como preços e saldo devem ser maiores ou iguais a 0. Os valores se é 0 ou 1 é para identificar se o utilizador tem conta Premium (1) ou não (0) e se o lugar está ocupado (1) ou livre (0).

Para o tipo de zona, classificação da aplicação e email criaram-se Check Constraints.

Na seleção dos dados onde se pretende obter resultados consoante um determinado estacionamento, por exemplo, realizou-se funções em SQL e PL/pgSQL para se obter um determinado resultado consoante uma variável. Sendo esta uma forma mais simples de se adquirir o que se tenciona, todo o código SQL que exige mais que uma query foi feito na Base de Dados, o que basta apenas invocar no Java.

Outra questão pertinente são os triggers. Estes são operações que acionam automaticamente quando um determinado evento acontece. Isto aplicou-se no seguinte: verificação do número limite de matrículas a registar na aplicação; gerir o processo de pagamento do lugar quando o utilizador sai do lugar; o pagamento da multa; e o chamado initcap (1ª letra maiúscula e restantes letras minúsculas) dos nomes dos utilizadores quando estes se registam ou atualizam.

O trigger da verificação do número de matrículas consiste em verificar se o utilizador ao registar uma matrícula atinge o número máximo (4). Se registar a 5ª matrícula na Base de Dados ocorre uma mensagem de erro.

O trigger do processo de pagamento do lugar consiste no seguinte:

- A pessoa estaciona o seu veículo, e seguidamente, o sensor externo verifica a matrícula. Caso a matrícula não conste na Base de Dados passa a ser um assunto que não é responsabilizado pela app. Caso contrário, identifica a matrícula e regista-a na tabela lugar, registando de forma automática o tempo a partir daquele momento. Adiciona também à tabela utilizadorestacionamento os dados relativos ao utilizador associado à matrícula. Depois de este sair do lugar, o estado do lugar passa para 0 e é calculado o custo consoante o preço da zona respetiva e a relação entre o tempo que a pessoa saiu do lugar e a pessoa estacionou o veículo.

Depois de feito o cálculo, é feita uma verificação com o saldo do utilizador – caso tenha dinheiro suficiente gera um registo na ordempagamento e é-lhe retirado o preço do lugar ao saldo que possui no IBAN. Se não tiver dinheiro suficiente gera uma multa tendo de pagar uma taxa (mais 3€). Em ambas as situações são eliminados os registos no lugar e do utilizadorestacionamento e é registado toda informação respetiva na tabela tempoestacionado.

O trigger do pagamento da multa consiste apenas em verificar se o utilizador possui dinheiro suficiente para pagar o lugar na qual foi gerada multa, isto é, assim que há um UPDATE na tabela, o trigger aciona imediatamente se o saldo é suficiente para poder pagar o lugar.

Finalmente, o trigger que aciona quando há um INSERT ou UPDATE na tabela detalhesutilizador permite alterar a 1ª letra para maiúscula e as restantes letras para minúsculas do campo nome.

Outro assunto importante são as views. Consiste num poderoso mecanismo faz uma representação virtual de tabelas criadas a partir de um SELECT. É geralmente utilizada para devolver dados desnormalizados evitando que o utilizador que proceda constantemente à relação entre mais que uma tabela (JOINS). Sendo isto bastante útil, criaram-se diversas views que impede que o user esteja sempre a consultar tabelas com JOINS. Tendo em conta o contexto do problema, aplicou-se este conceito na obtenção do seguinte: afluência local (afluência do número de lugares ocupados por estacionamento), média ocupacional por zona, número total de estacionamentos por zona, informação detalhada das classificações da app incluindo endereço eletrónico e contacto, custo de cada lugar que corresponde a cada estacionamento, informação da posse das matrículas para cada user, top utilizadores com mais contribuições na aplicação, detalhes do histórico de pagamento e ainda todos detalhes pessoais de cada user com endereço eletrónico e contacto. Facilita assim que se façam JOINS cada vez que se pretende devolver resultados em mais que uma tabela e necessita apenas que se invoque a tabela por meio de uma query simples.

Quanto à instalação, utilizou-se o PostgreSQL 10 e o pgAdmin III para interagir com a Base de Dados e o Netbeans 8.2 para a lógica de negócio e interfaces.

13 Manual de Utilização

13.1 Registo

Ao iniciar a aplicação, é apresentado algumas opções na qual o utilizador tem de escolher consoante as suas necessidades.

Ao se clicar no botão de “Registar” é apresentada uma janela com campos a serem preenchidos com dados pessoais do utilizador, do seu veículo e dados necessário para a sua autenticação na aplicação.

Enquanto o utilizador faz o Registo é feita uma verificação em cada campo à exceção dos Campos Nome e Morada que apenas não podem ser apresentados como campos vazios.

Nos restantes campos é feita uma verificação dos dados que o utilizador introduz, sendo feita pelo número e tipo de caracteres inseridos. Caso os dados estejam incorretos, estes aparecem com as letras ou números a cor vermelha.

Após a introdução de dados autênticos nos respetivos campos é ainda necessário dar a indicação de que o utilizador é maior de idade e caso queira, pode ainda fazer o upgrade da sua conta para Conta Premium.

Aquando o término do seu registo, assim que o utilizador clica no botão “OK”, os seus dados são guardados na Base de Dados.

Caso o utilizador opte por cancelar o registo, todos os dados inseridos nos respetivos campos são apagados.

13.2 LOGIN

Na janela “Iniciar Sessão” são apresentados dois campos para a autenticação do utilizador. No primeiro campo é necessário a inserção do E-mail ou número de telemóvel e no segundo campo a sua Palavra-passe. Quando o utilizador clica no botão de “Iniciar sessão” ou pressiona a tecla Enter é feita uma verificação do seu e-mail ou número de telemóvel como também da sua Palavra-passe. Caso os dados apresentados estejam contidos na base de dados o utilizador é reencaminhado para a janela principal, caso contrário é apresentada uma mensagem de erro que indica que o utilizador ou a palavra-passe inseridos estão incorretos.

13.3 Utilizador

13.3.1 Página Inicial

13.3.1.1 Ver Parques

A opção ‘Ver Parques’ presente na página principal dá, ao utilizador, a possibilidade de visualizar o número de estacionamentos livres, o número de lugares ocupados e o preço destes por zona.

Para a visualização dos estacionamentos livres e do seu preço, primeiramente é apresentada uma janela ao utilizador onde este terá de escolher a zona desejável onde queira estacionar. Só após a indicação da zona pretendida é que o utilizador é redirecionado para uma janela onde estão presentes todas as informações dos vários estacionamentos.

Sempre que o utilizador termine a sua pesquisa, o botão ‘Voltar’ redireciona-o para a janela ‘Zonas’. Deste modo, é mais fácil para o utilizador visualizar todas as zonas existentes, prevenindo também erros caso haja um lapso por parte do utilizador e este clique numa zona não pretendida.

13.3.1.2 Obter Conta Premium

Esta opção é descrita na alínea 13.3.2.2.

13.3.1.3 Sobre

Nesta opção o utilizador é encaminhado para uma janela onde é descrita uma breve explicação dos objetivos e das tarefas oferecidas pela aplicação.

13.3.1.4 Terminar Sessão

A opção ‘Terminar Sessão’, tal como o nome indica, permite ao utilizador sair da sua sessão, sendo redirecionado para a janela de ‘Iniciar Sessão’.

13.3.2 Ver Perfil

Após a autenticação de conta, é possível ver os dados do utilizador ao clicar no botão “Ver Perfil” onde o utilizador é direcionado para uma interface onde são apresentadas várias opções.

13.3.2.1 Ver Dados

Após a autenticação, é possível ver os dados do utilizador ao clicar no botão “Ver Perfil” onde o utilizador é direcionado para uma interface onde são apresentadas várias opções.

No caso de o utilizador escolher a opção “Ver Dados”, são apresentados todos os dados fornecidos pelo utilizador aquando o seu registo como também a possibilidade de alterar os seus dados caso haja necessidade para tal. Por questões de segurança, alguns dados mais sensíveis, tal como o número de conta, número de telemóvel e palavra-passe são apresentados por ‘*’ e dados tais como número de BI e NIF não são apresentados.

Caso o utilizador queira alterar algum dos registos, é necessário primeiramente clicar no botão de “Editar” apresentado à frente de cada campo para que estes deixem de estar bloqueados a alterações e, no caso de algum erro ainda é dada a opção de anular as alterações feitas.

Assim que o utilizador termina as alterações dos seus dados é necessário clicar no botão “Atualizar” para que os registos sejam devidamente alterados na Base de dados.

São apresentados também os botões de “Voltar” onde o utilizador é redirecionado para a interface Ver Perfil como também o botão “Terminar Sessão” para o término da sua sessão.

13.3.2.2 Adesão Conta Premium

Ao clicar no botão “Obter Conta Premium” presente na interface é exibida uma janela onde o utilizador tem a opção de fazer o upgrade da sua conta para Premium ou voltar para a página inicial, como também o custo do upgrade de conta.

Caso o utilizador ordinário decida ser um utilizador premium, após a confirmação de upgrade, é-lhe retirado o valor apresentado do seu saldo e reencaminhado para a página inicial.

13.3.2.3 Gestão de Matrículas

Aquando o utilizador acede à interface “Gerir matrículas”, este pode modificar os dados relativos à matrícula que inseriu no registo da sua conta. A modificação das informações não pode ser efetuada enquanto o utilizador não clicar no botão “Modificar”. Assim que o fizer, os espaços definidos para esta funcionalidade tornam se editáveis, sendo possível então modificar os dados.

Também é possível o utilizador poder introduzir um número máximo de quatro matrículas. Para tal, o utilizador necessita clicar no botão “Adicionar”, tornando os campos editáveis, sendo assim possível preencher os dados necessários. Terminando de preencher os dados o utilizador clica no botão “Atualizar” no canto inferior direito da interface, guardando assim a(s) matrícula(s) na base de dados. Assim que os dados forem guardados, o utilizador pode querer eliminar mais tarde a(s) matrícula(s) adicionada(s), clicando no botão “Eliminar” disponível.

13.3.2.4 Histórico de Pagamentos

Nesta opção é apresentada uma tabela com todos os pagamentos feitos pelo utilizador na utilização da aplicação para o pagamento de estacionamentos.

Ao clicar na opção ‘Voltar’ o utilizador é redirecionado para a janela ‘Ver Perfil’.

13.3.2.5 Classificar a Aplicação

Nesta janela é pedido ao utilizador para classificar a aplicação de acordo com a escala de 0-5 como também um comentário, não sendo este um campo obrigatório. Após a classificação da aplicação e da escrita, ou não, de um comentário é necessário submeter a sua avaliação.

O utilizador terá também a opção de retroceder para a página ‘Ver Perfil’ com o botão ‘Voltar’ e a opção ‘Terminar Sessão’ que o reencaminhará para a página ‘Iniciar Sessão’.

13.3.2.6 Remoção de Conta

Caso o utilizador queira eliminar a sua conta da aplicação, é-lhe dado uma opção, ‘Remover Conta’ presente na página ‘Ver Perfil’. Após o clique, é apresentado uma mensagem de confirmação da remoção de conta, como também a opção ‘Cancel’ de modo a evitar qualquer lapso.

13.3.3 Administrador

Na realização do administrador teve-se em conta este ter a possibilidade de se comportar como qualquer outro utilizador da aplicação, tendo ainda acesso reservado a funcionalidades como a atualização dos preços, como visualizar estatísticas e ainda poder consultar utilizadores.

13.3.3.1 Menu

O menu apresentado ao Administrador tem como finalidade possibilitar todas as ações que este pretende, tais como, atualizar preços, Visualizar estatísticas e Consultar dados dos utilizadores. Tem também a possibilidade de terminar a sessão.

13.3.3.2 Atualizar Preço

Abordando as funcionalidades da atualização dos preços, o administrador tem a opção de atualizar o preço do estacionamento consoante a zona ou atualizar o valor necessário para o upgrade para utilizador premium.

Para o atualizar o preço dos estacionamentos numa determinada zona, o administrador apenas tem de especificar qual a zona a ser atualizada assim como o novo preço base, tendo por fim que confirmar a ação.

O atualizar valor do upgrade permite ao administrador definir um preço para o upgrade de um utilizador ordinário para um utilizador premium.

13.3.3.3 Estatísticas

Desenvolvendo a estrutura da funcionalidade de visualizar estatísticas, nesta opção o administrador poderá visualizar a média de ocupação, a afluência local, os lucros obtidos e a classificação dada à aplicação.

13.3.3.3.1 Média Ocupação

Para a média de ocupação o administrador poderá, através da tabela, visualizar a média por zona. Nesta tarefa foi utilizado uma vista da base de dados para armazenar os devidos valores.

13.3.3.3.2 Afluência Local

Na afluência local o administrador poderá ter acesso a todas as afluências de cada estacionamento na respetiva zona, caso pretenda visualizar apenas a afluência dos estacionamentos numa determinada zona, apenas tem de ver detalhes e inserir a zona que pretende ter acesso. Para tornar isto mais simples foi utilizado uma vista da base de dados para facilitar o acesso aos dados que se pretendiam.

13.3.3.3.3 Consultar Lucros

Na consulta dos lucros o utilizador poderá visualizar os lucros obtidos com o pagamento do lugar, com as multas e com cobrança da atualização de conta. Para o administrador puder consultar a informação apenas necessita de pressionar o botão de visualizar.

13.3.3.3.4 Classificação da App

Por fim, caso o administrador queira visualizar a classificação atribuída à aplicação, poderá através da tabela, visualizar a classificação referida pelos utilizadores. Caso pretenda visualizar apenas uma determinada classificação através do botão “ver detalhes” onde poderá obter todos os utilizadores que classificaram a aplicação.

13.3.3.4 Consultar Utilizador

Por fim, o administrador, de forma reservada, pode ainda consultar os utilizadores. Aqui o administrador poderá ter acesso ao top dos utilizadores, às multas e pode ainda remover utilizadores.

13.3.3.4.1 Top Utilizadores

Para o administrador poder visualizar o top utilizadores, que consiste nos utilizadores que já realizaram pagamentos com a aplicação com igual valor ou superior ao introduzido pelo administrador, sendo que a informação é vista através da tabela. Para tornar este processo mais fácil e eficaz, foi realizado uma vista na base de dados, onde é armazenado os dados do utilizador e o total de pagamentos executados em valor.

13.3.3.4.2 Multas

Na opção Multas, o administrador tem a possibilidade de ver uma tabela com as multas dos utilizadores, como o seu motivo e o valor da mesma.

13.3.3.4.3 Remover Utilizador

No caso de o administrador pretender remover um utilizador poderá fazê-lo introduzindo o email do mesmo e confirmando a sua ação.

14 Simulação de Estacionamentos

De modo a ser possível testar a aplicação, foi criado um package com várias classes que permite utilizadores aleatórios estacionarem em lugares aleatórios, assim como as suas saídas dos parques.

Para isso, foram criadas duas principais classes, a classe ‘Lugar’ e a classe ‘Matricula’.

Dentro da classe ‘Lugar’ foi criado o método ‘lugares’ onde são selecionados todos os lugares de estacionamento presentes na base de dados, tendo sido estes armazenados num *ArrayList* para futura utilização destes.

Dentro da classe ‘Matricula’ foi criado o método ‘matriculas’ permitindo assim obter todas as matrículas existentes na base de dados, tendo sido posteriormente armazenadas num *ArrayList*.

De modo a simular os estacionamentos, foi criada uma nova classe, a classe ‘Estacionar’. Dentro desta classe, foram criados alguns métodos, sendo estes os seguintes: ‘update estado’ que altera o estado do lugar livre para ocupado; ‘insertmatricula’ que regista a entrada do carro no lugar; ‘checklugar’ que faz a verificação dos lugares livres; e por último o método ‘verestacionamento’ que permite ao carro procurar em todos os lugares, um lugar livre onde possa estacionar e, caso encontre algum lugar estaciona, caso contrário, procura no estacionamento seguinte.

No método main invocou-se o método ‘verestacionamento’ de modo a simular entradas de carros.

Por último foi criada a classe ‘Sairlugar’ onde, dentro da classe, se encontram os seguintes métodos: o método ‘numlugar’ onde é escolhido aleatoriamente um lugar que já esteja ocupado; o método ‘updatesaida’ que atualiza o lugar anteriormente escolhido de ocupado para livre. No método main foi criado um ciclo de modo que, a cada dois segundo, procura um lugar que esteja ocupado e altera o seu estado para livre simulando a saída de um carro.

15 Análise Crítica de Resultados

Analisando o trabalho na integra, grande parte dos requisitos foram cumpridos, porém, o mesmo não aconteceu a todos. Pelo esquema seguinte, é possível observar os requisitos que cumpridos e postos em prática e os que não foram cumpridos (Tabela 17).

Nota:

RFC – Convidado **RFU –** Utilizador Ordinário **RFUP –** Utilizador Premium

RFA – Administrador **RNF –** Não Funcionais

Tabela 17- Requisitos cumpridos e não cumpridos

Requisitos Funcionais	Cumpridos	Não cumpridos
RFC. 1	✓	
RFC. 2	✓	
RFU. 1	✓	
RFU. 2	✓	
RFU. 3	✓	
RFU. 4	✓	
RFU. 5	✓	
RFU. 6	✓	
RFU. 7		✗
RFU. 9	✓	
RFU. 10	✓	
RFUP. 1		✗
RFUP. 2	✓	
RFA. 1	✓	
RFA. 2	✓	
RFA. 3	✓	
RFA. 4	✓	

Requisitos Funcionais	Cumpridos	Não cumpridos
RFA. 5	✓	
RFA. 6	✓	
RFA. 7	✓	
RNF. 1		✗
RNF. 2	✓	
RNF. 3		✗
RNF. 4		✗
RNF. 5	✓	

Pela tabela acima, é possível observar que há um conjunto de requisitos que não foram cumpridos (Esclarecer dúvidas, obter conta Premium por tempo de adesão, variar o custo dos lugares por zona pela média ocupacional ou por tempo) e outros que foram cumpridos.

Relativamente ao primeiro requisito não-cumprido, o grau de dificuldade não era elevado no que respeita à sua execução; no entanto, como consistia num requisito de baixa prioridade, foi um pouco excluído no decorrer do projeto. Quanto à obtenção da conta Premium pelo tempo que o utilizador já está constado na aplicação, tornou-se num problema que despenderia bastante tempo a encontrar uma solução, tendo em conta que havia requisitos com maior relevância, pelo que se ponderou em tentar implementar no final caso ainda restasse tempo. Em relação à variação do custo dos lugares pela média ocupacional, foi bastante difícil encontrar uma solução para definir os preços consoante a afluência por zona. Tentou-se criar um trigger que aciona sempre que a média ocupacional numa determinada zona ultrapassasse uma determinada escala, porém, não funcionou devido à inadaptação dos triggers neste sentido, isto é, os triggers só respondem a determinados eventos quando há um INSERT ou um UPDATE numa tabela, pelo que era um caminho a percorrer que não iria ter saída. Discutiu-se, entretanto, que a única solução plausível seria desenvolver esta variação por meio de um método em Java, contudo, era bastante complicado ter o método a ser executado constantemente e deixou-se de parte, dando prioridade a outros requisitos. Na variação dos preços num determinado período do dia ou dia semana, criou-se uma função em PL/pgSQL que aumenta e diminui o preço das zonas consoante uma hora do dia mas nunca chegou a ser posta em prática.

16 Conclusão

Em suma, com a realização deste projeto, conseguiu-se desenvolver uma proposta tendo em conta o tema escolhido, correspondendo às expectativas que foram atribuídas ao longo do projeto. A julgar pelos testes realizados no decorrer do trabalho, concluiu-se que: um utilizador (*guest*) que não tenha uma conta associada tem a possibilidade de ver os parques de estacionamento, agrupados por zonas, com a inclusão dos respetivos preços; é possível registar uma conta inserindo os dados pessoais e as suas credenciais; tendo uma conta associada, o user pode iniciar sessão, na qual adquire acesso aos parques de estacionamento com o número total de lugares livres e ocupados, alterar o seu perfil, aderir a uma conta premium, gerir matrículas, classificar a aplicação e ainda remover a própria conta.

Ao longo do trabalho surgiram algumas dificuldades, nomeadamente no processo de gestão da Base de Dados. Foi um desafio encontrar a melhor solução para se obter uma base de dados rápida e consistente, estabelecendo um equilíbrio o máximo possível entre o código em SQL e em Java. Optou-se, portanto, por manter as queries maiores agregadas em funções SQL e/ou PL/pgSQL e assim era necessitava-se apenas de as invocar pelo Java. Outro aspeto foi em criar os triggers na qual requereu algum tempo para adquirir o conhecimento lógico e em gerir os pagamentos dos lugares. Foi necessário ter uma abordagem constante à Base de Dados para inferir nos triggers que eram precisos implementar, mediante a descrição do problema.

Além disso foi bastante difícil definir os preços consoante a hora do dia e a média ocupacional. No debater desta questão, tentou-se utilizar triggers em views que devolvem a média ocupacional por estacionamento e acioná-los caso a média aumente ou diminui numa determinada escala; no entanto, os triggers atuam apenas quando há um INSERT ou UPDATE na view, o que não se adapta à questão do problema, visto que o trigger deve acionar sempre que há mudança de estado da média. Dito isto, a variação dos preços em função do tráfego e do período do dia não foi conseguida. Outros dos aspetos que não foram conseguidos são os seguintes: obtenção de uma conta premium por tempo de adesão, esclarecimento de dúvidas e ainda o reservar lugar. Ambicionou-se ainda permitir que o administrador tenha acesso às estatísticas por meio de gráficos em Excel, isto é, este escolhe ver os utilizadores com mais contribuição, em termos financeiros, na aplicação, por exemplo, e é-lhe gerado um gráfico com os 5 primeiros users e ainda o utilizador e administrador obtêm os resultados consoante o tempo (última semana, há 48h…).

No âmbito teórico e prático, o projeto foi concebido com maior domínio na área

da programação, engenharia de software e nos sistemas de base de dados, isto porque foi necessário desenvolver código que processasse os dados da Base de Dados, obedecer a um tipo de processo de desenvolvimento (eng. software) e gerir o sistema de Base de Dados.

Foi deveras enriquecedor ter oportunidade de desenvolver uma aplicação com esta envergadura, devido à dimensão e as imensas opções que poderiam ser implementadas. Diante do tema escolhido, era possível enveredar por diversos caminhos e explorar muitas funcionalidades que podiam ser postas em prática.

Este foi um projeto que ajudou o grupo a compreender o que é necessário para poder ser feito uma aplicação que necessita de atenção constante. Desta forma, o grupo teve de interagir entre si para poder coordenar as tarefas para trazer “ao de cima” algo como foi criado neste projeto. Apesar de existir algumas funcionalidades por implementar e das dificuldades indicadas acima, o grupo esforçou-se por completar as tarefas propostas.

17 Bibliografia

Millington, S. (1 de Janeiro de 2019). *Baeldung*. Obtido de Baeldung: https://www.baeldung.com/java-password-hashing?fbclid=IwAR0z7BNrGvdkkIr1zDcDy5j0qdH6b2mEpwPZaoXGyXwYknphK5Gd4Fv0dyA

Oracle . (s.d.). *Java™ Platform, Standard Edition 7 API Specification*. Obtido de Java™ Platform, Standard Edition 7: https://docs.oracle.com/javase/7/docs/api/?fbclid=IwAR0YuLCyIXXYamy2VOB5njwq8Q0C2DtBcH5w2Z3fTsXhLuH_GeH0-_qpp2w

POSTGRESQL. (s.d.). *PostgreSQL JDBC*. Obtido de PostgreSQL JDBC : http://www.postgresqltutorial.com/postgresql-jdbc/

Refsnes Data. (s.d.). *w3schools*. Obtido de w3schools: https://www.w3schools.com/sql/default.asp

Stack Overflow. (s.d.). *Stack Overflow*. Obtido de Stack Overflow: https://stackoverflow.com/

the PostgreSQL Global Development Group. (8 de Novembro de 2018). *PostgreSQL.org*. Obtido de PostgreSQL: https://www.postgresql.org/docs/

Tutorials Point. (s.d.). *tutorialspoint*. Obtido de tutorialspoint: https://www.tutorialspoint.com/sql/