



Relatório

Análise e Modelação de Software

Aluno/os:

Jonatas de Paula Nº 22562
Vitor Sá Nº 20484

Professor: João Pedro Barbosa da Silva

Licenciatura em Engenharia de Sistemas Informáticos (pós-laboral)

Barcelos, Janeiro, 2023

Resumo

A nossa estratégia é implementar um sistema de transportes onde uma pessoa possa recolher e devolver as bicicletas em locais selecionados na cidade de Barcelos e no campus do IPCA. Essa pessoa poderia ir a um de nossos locais e desbloquear uma bicicleta com um passe ou com um cartão bancário e então poderiam se deslocar pela cidade, para interromper o passeio a pessoa teria de ir até a mesma ou outra de nossas localidades e trancar a bicicleta. Após o bloqueio nosso sistema verificará o horário do passeio, então calculará e sacará o valor referente a esse horário. Para as pessoas que optarem pelo cartão bancário propomos a retenção de um ativo no banco durante o período da viagem, que será reembolsado após a conclusão e pagamento da viagem. Isso é para garantir que os clientes tenham fundos suficientes para desbloquear uma bicicleta.

Palavras-Chave:

Diagrama Classe

Diagrama de Sequência

Diagrama de Uso

Diagrama de Contexto

Lista de Abreviaturas e Siglas

IPCA – Instituto Politécnico do Cavado e do Ave

BiciBox – Armazem da Bicicleta

PN - Processos de Negócios

BPMN - Business Process Model and Notation

CdU – Caso de Uso

PB - Product Backlog

UML - Unified Modeling Language, em português Linguagem de Modelagem Unificada

Índice de Figuras

FIGURA 1 - DIAGRAMA DE CLASSES – VISUAL PARADIGM STANDARD	3
FIGURA 2 - BPMN	4
FIGURA 3 - DIAGRAMA DE CLASSES UML.....	5
FIGURA 4 - PRODUCT BACKLOG.....	10
FIGURA 5 - ROADMAP - VISUAL PARADIGM STANDARD	10
FIGURA 6 - DIAGRAMA DE CLASSES - VISUAL PARADIGM STANDARD	11
FIGURA 7 – CÓDIGO DE INSERÇÃO - VISUAL STUDIO.....	13
FIGURA 8 - CÓDIGO DE GUARDA - VISUAL STUDIO	13
FIGURA 9 - TELA DE CADASTRO E LOGIN	14
FIGURA 10 - DIAGRAMA DE ESTADO - VISUAL PARADIGM STANDARD	16
FIGURA 11 - TESTE REALIZADO - VISUAL STUDIO.....	17
FIGURA 12 - REVISÃO DE VEÍCULO - VISUAL STUDIO	18
FIGURA 13 - TELA DE MANUTENÇÃO	18
FIGURA 14 - DIAGRAMA DE SEQUÊNCIA - VISUAL PARADIGM.....	19
FIGURA 15 - TESTE DE REVISÃO - VISUAL STUDIO	19

Índice

Introdução.....	1
1. Sprint 01 - Visão do produto	2
1.1. Objetivo do Projeto	2
1.1.1. Problemas a resolver.....	2
1.1.2. Objetivos de negócio e benefícios do projeto.....	2
1.2. Enquadramento.....	3
1.2.1. Diagrama de contexto do sistema.....	3
1.2.2. Descrição dos interessados ou stakeholders	3
1.2.3. Descrição das áreas funcionais, incluindo as principais features ou requisitos de alto nível de cada área	3
2. Sprint 02 - Definição do negócio	4
2.1. Processos de Negócios (PN)	4
2.1.1. Descrição dos PN a serem suportados pelo sistema	4
2.1.2. Incluir a modelação dos PN com diagramas BPMN	4
2.2. Modelo de Domínio.....	5
2.2.1. Diagrama de classes UML inicial para capturar as entidades de negócio.....	5
2.2.2. Definição de conceitos, termos e entidades	5
3. Sprint 03 - Definição dos requisitos.....	6
3.1. Modelo de Casos de Uso	6
3.1.1. Diagrama de Casos de Uso por Componente.....	6
3.1.2. Lista de Casos de uso.....	6
3.2. Regras de Negócio.....	8
3.3. Requisitos Não Funcionais.....	8
3.4. Pressupostos e Restrições	8
4. Sprint 04 – Planeamento Inicial do Projeto.....	9
4.1. Arquitetura Técnica.....	9

4.1.1.	Identificação e Justificação dos Componentes	9
4.1.2.	Representação Gráfica da Arquitetura	9
4.2.	Preparação do Product Backlog (PB).....	10
4.2.1.	Lista dos CdU ordenada por prioridade	10
4.2.2.	Elaborar Plano do Projeto (roadmap)	10
4.3.	Desenvolver Diagrama de Classes.....	11
5.	Sprint 05 – Dev Sprint.....	12
5.1.	Selecionar CdU mais prioritários (escolher 1 ou 2)	12
5.1.1.	Escrever as narrativas dos CdU: fluxo principal e fluxos alternativos.....	12
5.1.2.	Elaborar User Stories (US)	12
5.2.	Análise, implementação e teste de cada US	13
5.2.1.	Elaborar mockups para capturar os requisitos de interface	14
5.2.2.	Desenvolver um diagrama de estados: escolher classe mais adequada.....	16
5.2.3.	Implementar e testar as US: não há restrições quanto à tecnologia	17
6.	Sprint 06 – Dev Sprint.....	17
6.1.	Selecionar CdU mais prioritários (escolher 1 ou 2)	17
6.1.1.	Escrever as narrativas dos CdU: fluxo principal e fluxos alternativos.....	17
6.1.2.	Elaborar User Stories (US)	17
6.2.	Análise, implementação e teste de cada US	18
6.2.1.	Elaborar mockups para capturar os requisitos de interface	18
6.2.2.	Desenvolver um diagrama de sequência para os CdU selecionados	19
6.2.3.	Implementar e testar as US: não há restrições quanto à tecnologia	19
7.	Sprint 07 – Dev Sprint.....	20
7.1.	Selecionar CdU mais prioritários (escolher 1 ou 2)	20
7.1.1.	Escrever as narrativas dos CdU: fluxo principal e fluxos alternativos.....	20
7.1.2.	Elaborar User Stories (US)	20
7.2.	Análise, implementação e teste de cada US	21
7.2.1.	Elaborar mockups para capturar os requisitos de interface	22

7.2.2.	Atualizar o Diagrama de Classes.....	22
7.2.3.	Implementar e testar as US: não há restrições quanto à tecnologia	23
8.	Conclusão	27

Introdução

O conceito do projeto nasce, de certa forma, de uma necessidade. A infraestrutura atual carece de capacidade e investimentos recorrentes de manutenção para suportar o impacto que a população cada vez maior traz, sendo os dois principais culpados as estradas obsoletas e sem manutenção e a rede de transporte público. Uma vez que o custo de grandes projetos viários ou de toda uma rede de transportes coletivos supera largamente os nossos próprios custos projetados, estaríamos a trabalhar no nosso próprio interesse e no da população, que também é do interesse do município. Essa poderia ser uma das formas de olharmos para este projeto como um todo, é uma grande oportunidade que pode nos beneficiar tanto como grupo ou entidade quanto individualmente; bem como toda a população de Barcelos ao optar por empreender e resolver os desafios criados pela infraestrutura aquém do esperado.

1. Sprint 01 - Visão do produto

1.1. Objetivo do Projeto

O IPCA pretende oferecer um novo conceito de mobilidade usando as bicicletas do projeto U_Bike Portugal.

Para isso este sistema permitiria que através do uso de um cartão de crédito/debito ou passe especial, o utilizador alugasse uma bicicleta ou outro veículo semelhante numa das biciboxes registando a data e hora em que foi efetuado esse aluguer e, após depositada a bicicleta, um sensor iria ler o código dessa bicicleta e registaria a data e hora da entrega e assim calcular o custo da sua utilização descontando então da conta associada a essa utilização o valor do aluguer.

1.1.1. Problemas a resolver

Este projeto tem 2 grandes problemas: O controlo de danos causados e as medidas a combater os mesmos como coimas ou cauções. Gestão do número de bicicletas e outros meios de deslocação nas biciboxes permitindo sempre que os mesmos sejam estacionados propriamente e que não haja escassez de veículos em nenhum ponto.

1.1.2. Objetivos de negócio e benefícios do projeto

Haverá vários benefícios desde facilidade de deslocação para estudantes do IPCA, meio de deslocação mais verde e mais económico, baixo custo de manutenção e instalação comparado a outros meios de transporte entre outros, porem com o sistema os benefícios são uma forma simples e rápida de gerir não só os alugueres e os dados de todos os clientes e funcionários como também dos veículos e dos alugueres realizados. O sistema também facilita o pagamento dos alugueres e torna mais prática a sua utilização no dia a dia.

1.2. Enquadramento

1.2.1. Diagrama de contexto do sistema

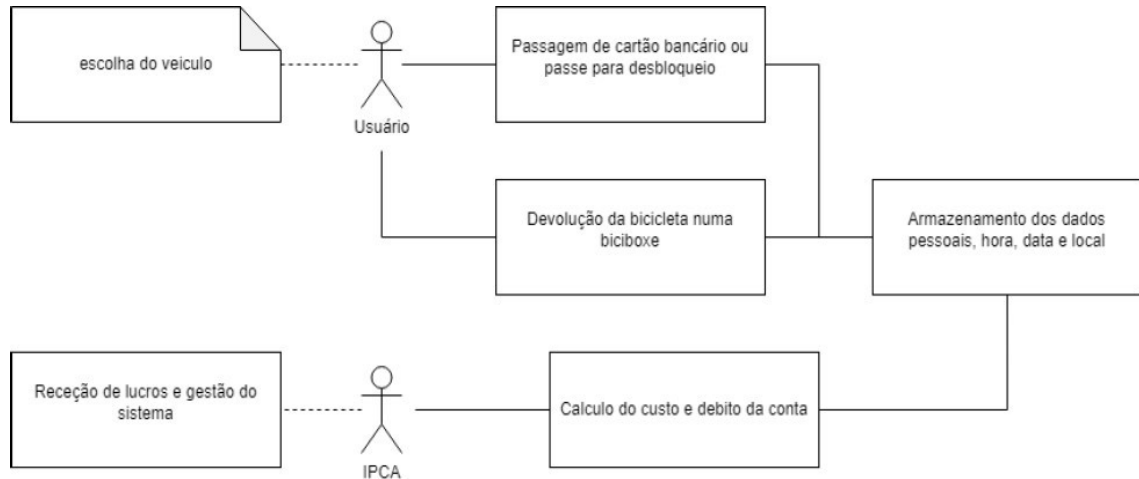


Figura 1 - Diagrama de Contexto – Visual Paradigm Standard

1.2.2. Descrição dos interessados ou stakeholders

Estudantes – Estes serão os utilizadores principais deste sistema e terão um grande impacto no seu sucesso assim como na sua expansão;

Concelho de Barcelos – Com um maior número de alunos a utilizar transportes mais verdes e simples, e o descongestionamento em horas de ponta;

IPCA – Beneficiaria diretamente com este projeto sendo o recetor direto do seu lucro e gestor do mesmo.

1.2.3. Descrição das áreas funcionais, incluindo as principais features ou requisitos de alto nível de cada área

Facilidade no aluguer da bicicleta ou outro meio, sem necessidade para o preenchimento de qualquer documento ou formulário;

Simplicidade na forma do pagamento, sendo este efetuado de forma direta pela conta bancaria ou via subscrição mensal a um passe especial;

Praticidade de utilização, não existindo qualquer necessidade de atentar para a manutenção do veículo escolhido ou com o seu carregamento no caso do mesmo ser elétrico.

2. Sprint 02 - Definição do negócio

2.1. Processos de Negócios (PN)

2.1.1. Descrição dos PN a serem suportados pelo sistema

O cliente dirige-se a uma bicibox, após selecionar a bicicleta pretendida deve passar o seu cartão de crédito ou passe especial para que a seja efetuado o destranque da bicicleta desejada, nesse momento a hora e data de aquisição são guardados e enviados para os SAS para que seja efetuado um melhor controlo da disponibilidade de bicicletas de cada tipo em cada ponto e seja possível se necessário à sua reposição e os dados do cartão são guardados para debito futuro, pós entrega. Após a utilização a bicicleta é então devolvida numa bicibox e é então calculado o custo de utilização com base na data e hora da entrega e é debitado o valor calculado da utilização da bicicleta.

2.1.2. Incluir a modelação dos PN com diagramas BPMN

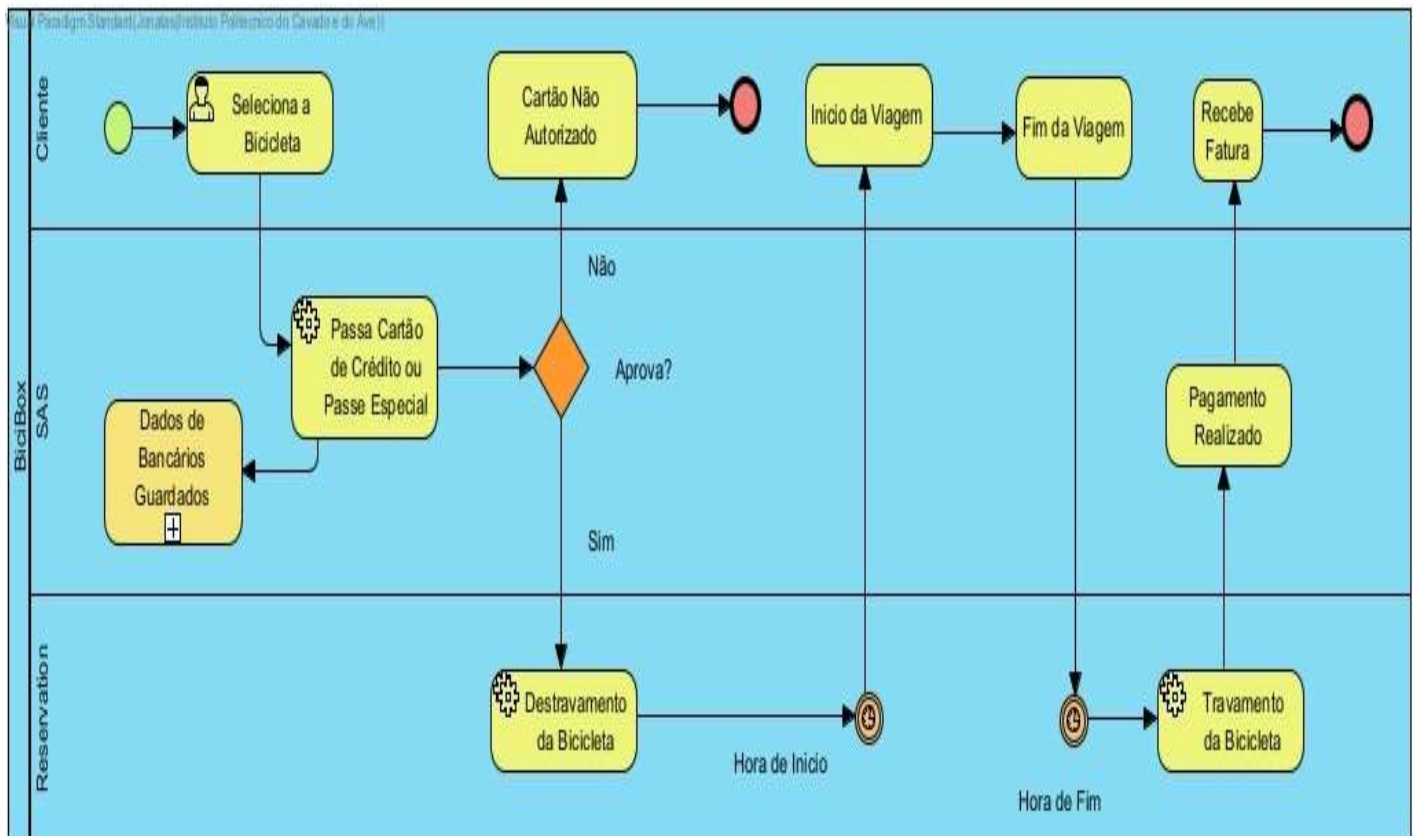


Figura 2 – BPMN – Visual Paradigm Standard

2.2. Modelo de Domínio

2.2.1. Diagrama de classes UML inicial para capturar as entidades de negócio

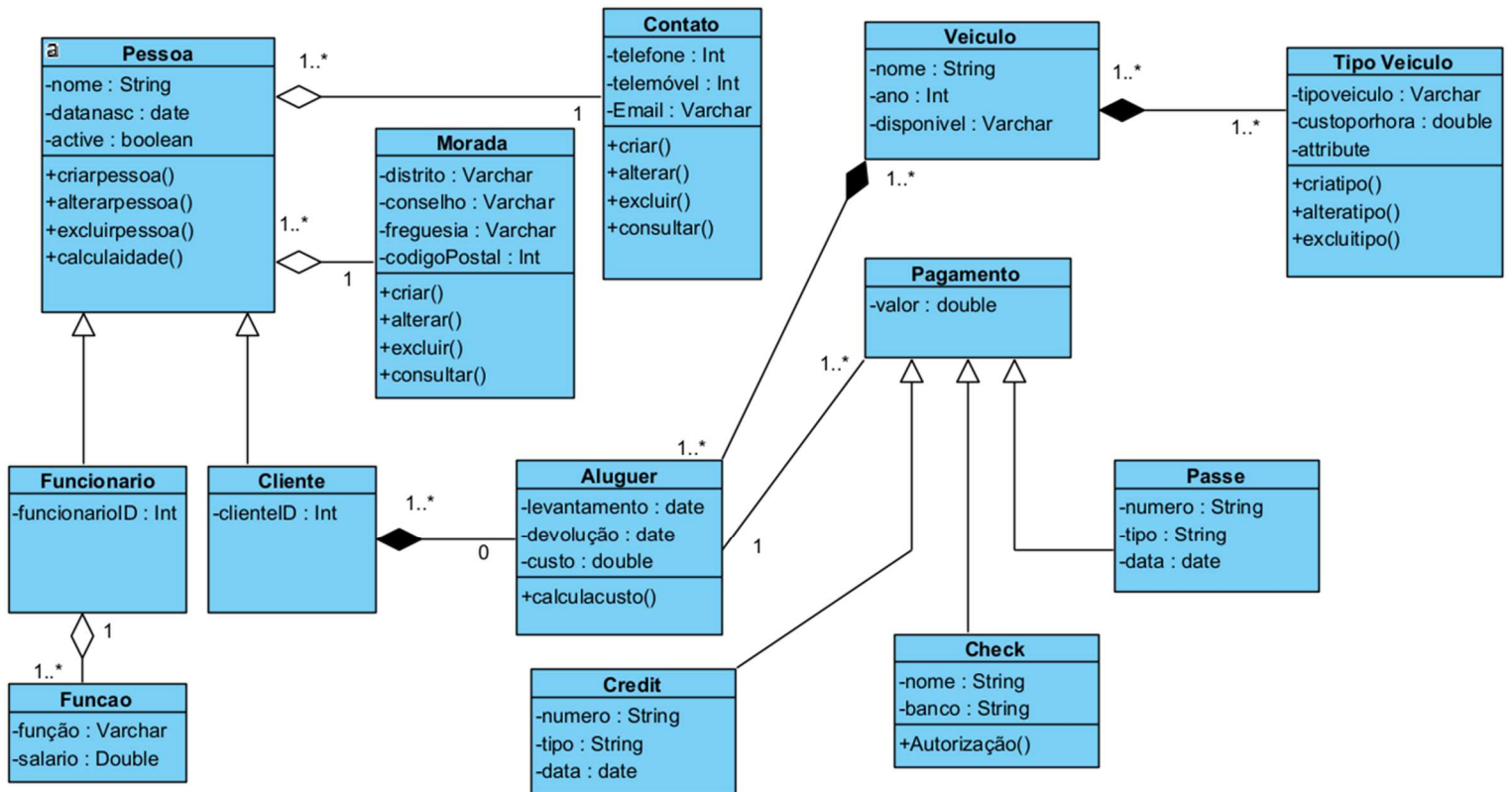


Figura 3 - Diagrama de Classe UML – Visual Paradigm Standard

2.2.2. Definição de conceitos, termos e entidades

Clientes - Alunos, docentes ou outros colaboradores do IPCA que sejam clientes da plataforma BiciBox.

Passe — Transação eletrónica que debita do cartão de crédito ou passe especial e permite o utilizador mover-se com as bicicletas.

BiciBox — São locais pré-definidos ao redor do IPCA onde poderá levantar e devolver as bicicletas.

3. Sprint 03 - Definição dos requisitos

3.1. Modelo de Casos de Uso

3.1.1. Diagrama de Casos de Uso por Componente

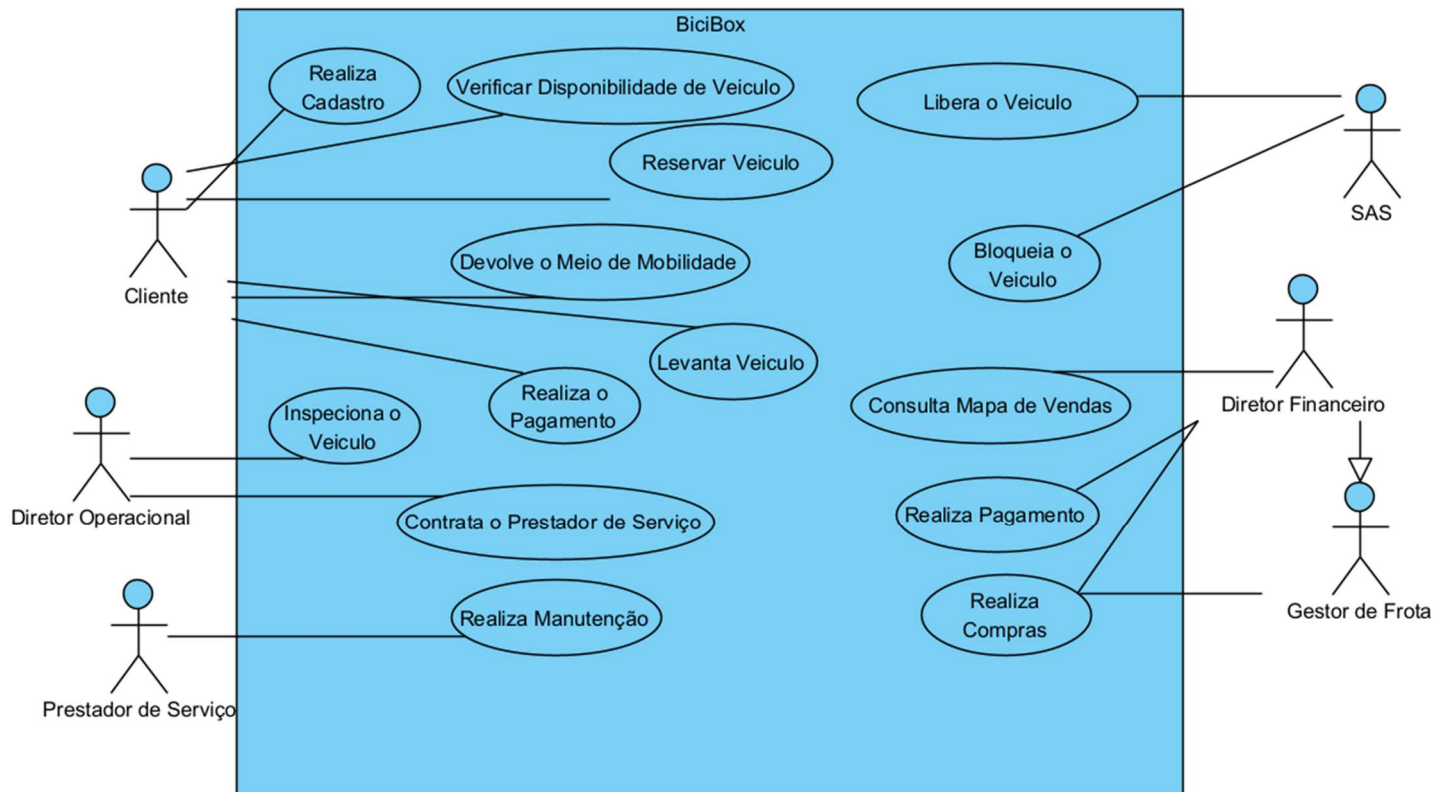


Figura 4 - Diagrama de Uso - Visual Paradigm Standard

3.1.2. Lista de Casos de uso

Ator principal: **Cliente**

Nome do Use Case: **Realiza Cadastro**

Breve descrição: O utilizador deve realizar o cadastro na aplicação para utilizar o meio de mobilidade.

Nome do Use Case: **Verificar Disponibilidade de Veículo**

Breve descrição: Irá verificar se existe veículo a disposição.

Nome do Use Case: **Seleciona o Veículo**

Breve descrição: Irá escolher entre os veículos disponíveis, qual irá utilizar.

Nome do Use Case: **Libera o Meio de Mobilidade**

Breve descrição: Após verificar e escolher irá fazer a liberação do veículo.

Nome do Use Case: **Devolve o Meio de Mobilidade**

Breve descrição: Após utilização Devolve o Meio de Mobilidade diretamente na BiciBox.

Nome do Use Case: **Realiza o Pagamento**

Breve descrição: Após Devolver realizar o pagamento através da plataforma ou dinheiro.

Ator principal: **Diretor Operacional**

Nome do Use Case: **Inspeciona o Veículo**

Breve descrição: Inspeciona o veículo antes e após voltar da manutenção.

Nome do Use Case: **Contrata o Prestador de Serviço**

Breve descrição: Responsável por contratar o prestador de serviço de manutenção.

Ator principal: **Prestador de Serviço**

Nome do Use Case: **Realiza Manutenção**

Breve descrição: Responsável por reparar todos os veículos adjudicados.

Ator principal: **Gestor de Frota**

Ator Secundário: **Diretor Financeiro**

Nome do Use Case: **Realiza Compra**

Breve descrição: Responsável por fazer todas as compras relacionadas com a frota.

Ator principal: **Diretor Financeiro**

Nome do Use Case: **Realiza Pagamento**

Breve descrição: Responsável por fazer todos os pagamentos financeiros.

Nome do Use Case: **Consulta Mapas de Venda**

Breve descrição: Pode consultar o mapa de vendas a qualquer instante.

3.2. Regras de Negócio

Nossa estratégia é implantar um sistema de transporte onde uma pessoa possa pegar e deixar bicicletas em pontos selecionados da cidade de Barcelos e do campus do IPCA. Essa pessoa poderia ir a um de nossos locais e desbloquear uma bicicleta com um passe ou com um cartão bancário e então poderia se deslocar pela cidade. Então para parar o passeio a pessoa teria de ir ao mesmo ou outro de nossos locais e trancar a bicicleta. Após o bloqueio nosso sistema verificará o horário do passeio, então calculará e retirará o valor referente a esse horário. Para as pessoas que optarem por usar o cartão bancário, propomos a retenção de um ativo no banco durante a duração da viagem, que será reembolsado após a conclusão e pagamento da viagem. Isso é para garantir que os clientes tenham fundos suficientes para pagar o desbloqueio de uma bicicleta.

3.3. Requisitos Não Funcionais

- Liberação da licença junto ao município de Barcelos;
- Software que permitiria vários planos (viagem, dia, passe, passe de estudante);
- Sistema de pagamento com várias formas de pagamento (Cartão, Passe Especial);
- Conexão com a bloqueio inteligente para medir com precisão a hora de desbloqueio e bloqueio do veículo em vários locais;
- Uma forma de identificar possíveis danos aos veículos após uma viagem, como furos, componentes partidos ou em falta, quadro rachado ou partido, entre muitos outros, de forma a aplicar uma multa ao utilizador responsável como meio de compensar os custos de reparação.

3.4. Pressupostos e Restrições

Nossa análise inicial nos mostra que serão necessários aproximadamente três meses para entregar um sistema já integrado com as fechaduras inteligentes. Sendo o custo previsto para realização de todas as etapas é no montante de 63.000€ (sessenta e três mil euros), contudo, a falta de adesão do cliente pode ser o maior risco que o projeto enfrenta. O fato de os testes poderem começar durante o inverno pode significar números baixos em termos de adesão. Há também o risco de o município de Barcelos ser contra a maioria das ideias do projeto, os danos das bicicletas também podem ser um problema. Por esse

motivo, é recomendável dedicar algum tempo imaginando uma maneira de verificar a condição da bicicleta.

4. Sprint 04 – Planeamento Inicial do Projeto

4.1. Arquitetura Técnica

4.1.1. Identificação e Justificação dos Componentes

A componente funcional tem diferentes requisitos técnicos. As componentes “desbloqueio de bicicleta” e “entrega de bicicleta” precisam de ser acessíveis de qualquer bicibox, por isso é necessário que as biciboxs estejam conectadas a uma rede para que possam comunicar com o servidor.

O principal requisito para a componente “BiciGo” é que seja de fácil utilização e controlo, rápido e 100% fiável e disponível a toda a hora.

4.1.2. Representação Gráfica da Arquitetura

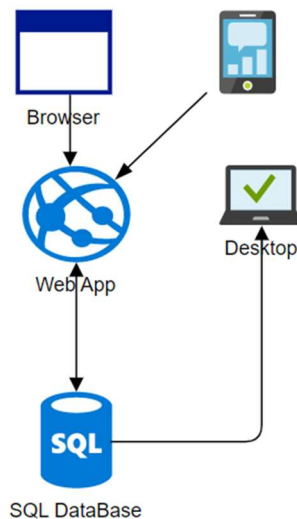


Figura 5 - Arquitetura Técnica

O utilizador realiza as consultas pelo browser e pelo smartphone (app), que também permite a realização dos pagamentos via mbway. Ambos o browser e o smartphone conectam-se com a web app que por fim comunica com a base de dados.

Para manipulação da base de dados e gestão das bicicletas e biciboxes, é utilizada a desktop app, pelos funcionários da SAS.

4.2. Preparação do Product Backlog (PB)

4.2.1. Lista dos CdU ordenada por prioridade

Neste projeto, o product backlog é composto por Use Cases (UC) e Use Case Slices (UCS) ao invés da utilização de user stories.

A priorização da UC é a atribuição de um nível de prioridade a cada item do Product Backlog, ordenando-os por nível de prioridade para que o item de maior prioridade seja desenvolvido primeiro. A técnica MoSCoW, usada abaixo usa 4 níveis de prioridade: Must, Should, Could and Would/Won't.

US Code	UC Name	Priority
UC 1.0	Cadastro Cliente	Must
UC 2.0	Cadastro Colaborador	Must
UC 3.0	Cadastro Veículo	Must
UC 4.0	Cadastro Revisão	Should
UC 5.0	Cadastro Aluguer	Must
UC 6.0	Verificar Disponibilidade de Veículo	Should
UC 7.0	Gerar Metade de Pagamento	Must
UC 8.0	Realiza Compra	Would/Won't

Figura 6 - Product Backlog

4.2.2. Plano do Projeto (roadmap)

Roadmap - Descrição	Jan/2023	Fev/2023	Mar/2023	Abr/2023	Mai/2023
Infra-Estrutura	<ul style="list-style-type: none"> Construção dos Locais de Armazenamentos Aquisição dos Equipamentos Informáticos 	<ul style="list-style-type: none"> Aquisição das Bicletas Aquisição dos Consumíveis 			
Formação Equipa	<ul style="list-style-type: none"> Contratar Administrador 	<ul style="list-style-type: none"> Contratar Diretor 	<ul style="list-style-type: none"> Contratação Colaboradores 		
Desenvolver Software	<ul style="list-style-type: none"> Projeto do Software 	<ul style="list-style-type: none"> Construção Software 	<ul style="list-style-type: none"> Construção Software - Fim Integração do Software 	<ul style="list-style-type: none"> Teste Software Documentação Software 	
Operação da Sociedade	<ul style="list-style-type: none"> Criação da Sociedade Aporte de Capital Criação da Marca 	<ul style="list-style-type: none"> Publicidade da Marca 	<ul style="list-style-type: none"> Contratar Seguro Publicidade da Marca 	<ul style="list-style-type: none"> Publicidade da Marca 	<ul style="list-style-type: none"> Início da Operação

Figura 7 - Roadmap - Visual Paradigm Standard

4.3. Diagrama de Classes

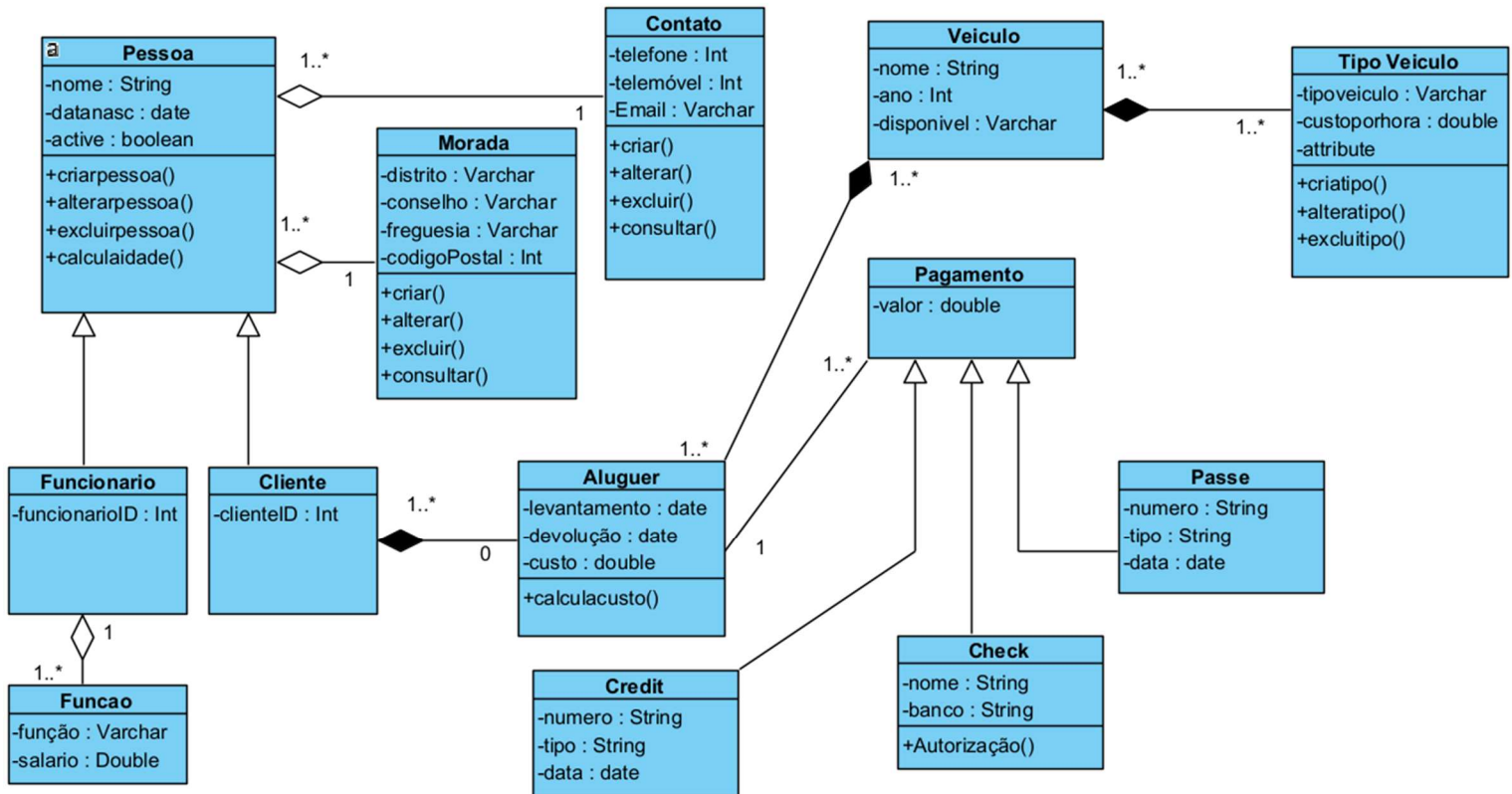


Figura 8 - Diagrama de Classes - Visual Paradigm Standard

5. Sprint 05 – Dev Sprint

5.1. CdU mais prioritários (escolher 1 ou 2)

Ator principal: **Cliente**

Nome do Use Case: **Realiza Cadastro**

Breve descrição: O utilizador deve realizar o cadastro na aplicação para utilizar o meio de mobilidade.

5.1.1. Narrativas dos CdU: fluxo principal e fluxos alternativos

Fluxo Principal	Fluxo Alternativo
FP1 - O cliente acessa a tela de Cadastro via Telemóvel ou Computador	FA1 - Caso o cliente tenha o cadastro realizado anteriormente, a aplicação irá informar e direcionar para alteração de login/senha.
FP2 - Após, deverá preencher as informações pessoais	
FP3 - Após, a aplicação verifica os dados inseridos	
FP3 - Após, verificado a aplicação guarda os dados inseridos	
FP4 - Após, cadastro é finalizado	

Figura 9 - Fluxo Principal e alternativo - Cliente

5.1.2. Elaborar User Stories (US)

Quando tive de fazer o cadastro direcionei-me ao aplicativo do telemóvel e abri a tela de cadastro e, continuei a preencher os dados solicitados, entretanto, fui informado pelo sistema que já tinha um cadastro sendo direcionado para troca de senha.

5.2. Análise, implementação e teste de cada US

Código que procura uma vaga na lista de clientes através do id, quando encontra insere o novo cliente na base de dados com o id livre encontrado anteriormente.

```
public class RulesCliente
{
    /// <summary>
    /// recebe cliente
    /// verifica que o id seja maior que 0 e se nao for altera
    /// altera o id para que nao seja igual a outro id ja existente
    /// chama o metodo InsereClienteBD
    /// </summary>
    /// <param name="c"></param>
    /// <returns></returns>
    3 references
    public static bool InsereCliente(Cliente c)
    {
        if (c.id == 0) c.id = 1;
        while (DL.DadosCliente.ExisteClienteBD(c) == true) c.id++;
        return DL.DadosCliente.InsereClienteBD(c);
    }
}
```

Figura 10 – Código de Inserção - Visual Studio

Código que guarda as informações do cliente no banco de dados.

```
public class DadosCliente
{
    ///criar metodo construtor para criar as listas;
    /// <summary>
    /// listas para clientes e ids
    /// </summary>
    static List<Cliente> clientes = new List<Cliente>();
    static List<int> id = new List<int>();

    /// <summary>
    /// recebe um cliente
    /// verifica que o id nao exista
    /// adiciona o id a lista de ids
    /// adiciona o cliente a lista de clientes
    /// </summary>
    /// <param name="c"></param>
    /// <returns></returns>
    1 reference
    public static bool InsereClienteBD(Cliente c)
    {
        while (id.Equals(c.id))
        {
            c.id++;
        }
        id.Add(c.id);
        clientes.Add(c);
        return true;
    }
}
```

Figura 11 - Código de Guarda - Visual Studio

5.2.1. Mockups para capturar os requisitos de interface



A mockup of a web form for login and registration. The form has a light gray background with a white border. At the top, the word "Login" is written in a large, bold, teal font. Below it, there are two input fields: "Seu e-mail" and "Sua senha". The "Seu e-mail" field contains the text "contato@bicibox.org.br" and the "Sua senha" field contains "1234". Below these fields is a checkbox labeled "Manter-me logado". At the bottom of the form is a large teal button labeled "Logar". Below the button, there is a link "Ainda não tem conta?" followed by a teal button labeled "Cadastre-se".

Figura 12 - Tela de Cadastro e Login

BICIBOX

[Página inicial](#) [Sobre](#) [Contato](#) [Nossos veículos](#)

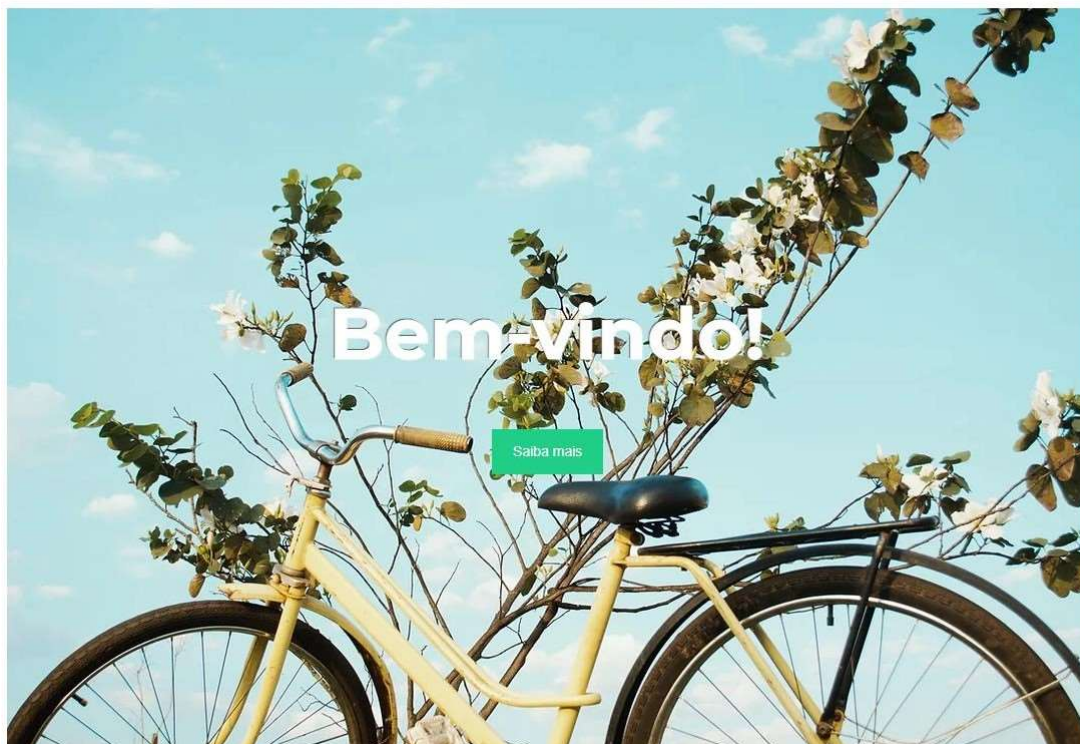


Figura 13 - Website | Página Inicial



Figura 14 – Website | Sobre

Alugar um veículo

Os nossos veículos



Bicicleta Elétrica

€ 1,50/hora

Bicicleta

€ 1,00/hora

Trotinete Elétrica

€ 1,50/hora

Figura 15 - Website | Os nossos veículos

Entre em contato

Campus do IPCA - Lugar do Aldão, 4750-810 Vila Frescainha (São Martinho)

info@bicibox.pt

252 111 111

Nome *

Insira seu nome

Telefone

Insira seu telefone

Assunto

Insira o assunto

Mensagem

Digite sua mensagem aqui

Enviar

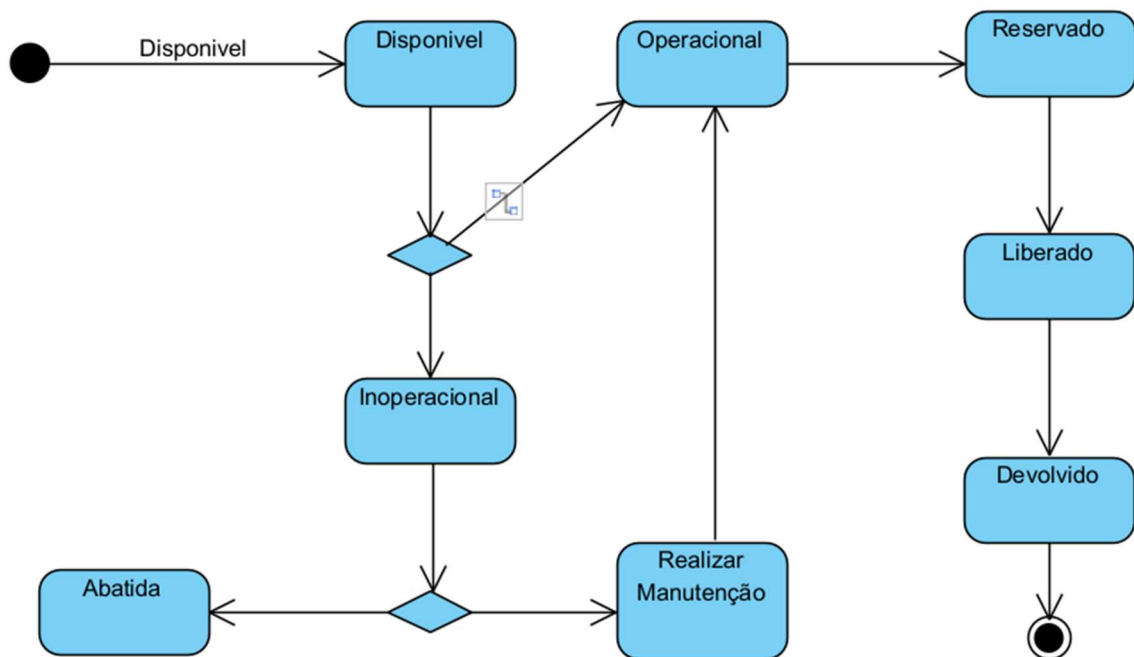
Mapa Satélite

Campus do IPCA - Lugar do Aldão, 4750-810 Vila Frescainha (São Martinho)

Directions

Figura 16 - Website | Contactos

5.2.2. Diagrama de estados: escolher classe mais adequada



5.2.3. Testes às US

```

      Clientes criados e inseridos

Clientes:

  id | Nome
  ---|-----
  1  | vitor
  2  | joao
  3  | andre
    
```

Figura 18 - Teste do cadastro Cliente - Visual Studio - console

6. Sprint 06 – Dev Sprint

6.1. CdU mais prioritários

Ator principal: **Diretor Operacional**

Nome do Use Case: **Inspeciona o Veículo**

Breve descrição: Inspeciona o veículo antes e após voltar da manutenção.

Nome do Use Case: **Contrata o Prestador de Serviço**

Breve descrição: Responsável por contratar o prestador de serviço de manutenção.

6.1.1. Narrativas dos CdU: fluxo principal e fluxos alternativos

Figura 19 - Fluxo Principal e Alternativo - Revisão

Fluxo Principal	Fluxo Alternativo
FP1 - Após inspeção e constatar que o veículo está avariada	FA1 - Caso não exista empresa especializada e, contratada uma oficina não especializada
FP2 - O Veículo é retirado da operação	
FP3 - Contratação da Oficina Especializada e, envio do Veículo	
FP3 - Retorno da oficina é realizado nova inspeção	FA2 - Caso não tenha conserto é realizada nova compra
FP4 - Tudo conforme como contratado o veículo retorna a operação	

6.1.2. User Stories (US)

Inspeciono todos os veículos semanalmente, encontrei um veículo com danos e fiz o envio do mesmo para a empresa contratada por mim para a sua manutenção/reparação. Após o retorno da oficina, realizo novamente a inspeção e estando tudo conforme contratado faço a aprovação e liberação do veículo.

6.2. Análise, implementação e teste de cada US

```
public class DadosRevisao
{
    /// <summary>
    /// cria lista de revisoes e ids
    /// </summary>
    static List<Revisao> revisoes = new List<Revisao>();
    static List<int> id = new List<int>();

    /// <summary>
    /// recebe uma revisao
    /// verifica se existe um id igual na lista de ids, caso exista acresce ao valor da nova revisao
    /// adiciona o id a lista de ids
    /// adiciona a lista de revisoes
    /// </summary>
    /// <param name="a"></param>
    /// <returns></returns>
    1 reference
    public static bool InsereRevisaoBD(Revisao a)
    {
        while (id.Equals(a.id))
        {
            a.id++;
        }
        id.Add(a.id);
        revisoes.Add(a);
        return true;
    }
}
```

Figura 20 – Metodo para registo de Revisão/Reparo de Veículo - Visual

O código acima serve apenas para exemplo, trata-se de um método para a inserção dos dados de uma nova revisão na lista de dados de revisões para mais tarde ser inserido na base de dados em SQL, para isso corre-se um ciclo *while* que vai aumentando o valor do id recebido até encontrar um id que não exista na lista de ids, assim garantimos que não existam “gaps” entre ids de revisões. Uma vez encontrado um id livre o novo id é adicionado a lista de ids para comparações futuras e os dados da revisão recebidos pelo método são guardados numa lista chamada revisões para mais tarde ser guardada na base de dados por outro método.

6.2.1. Mockups para capturar os requisitos de interface



Figura 21 - Tela de Manutenção

6.2.2. Diagrama de sequência para osCdU

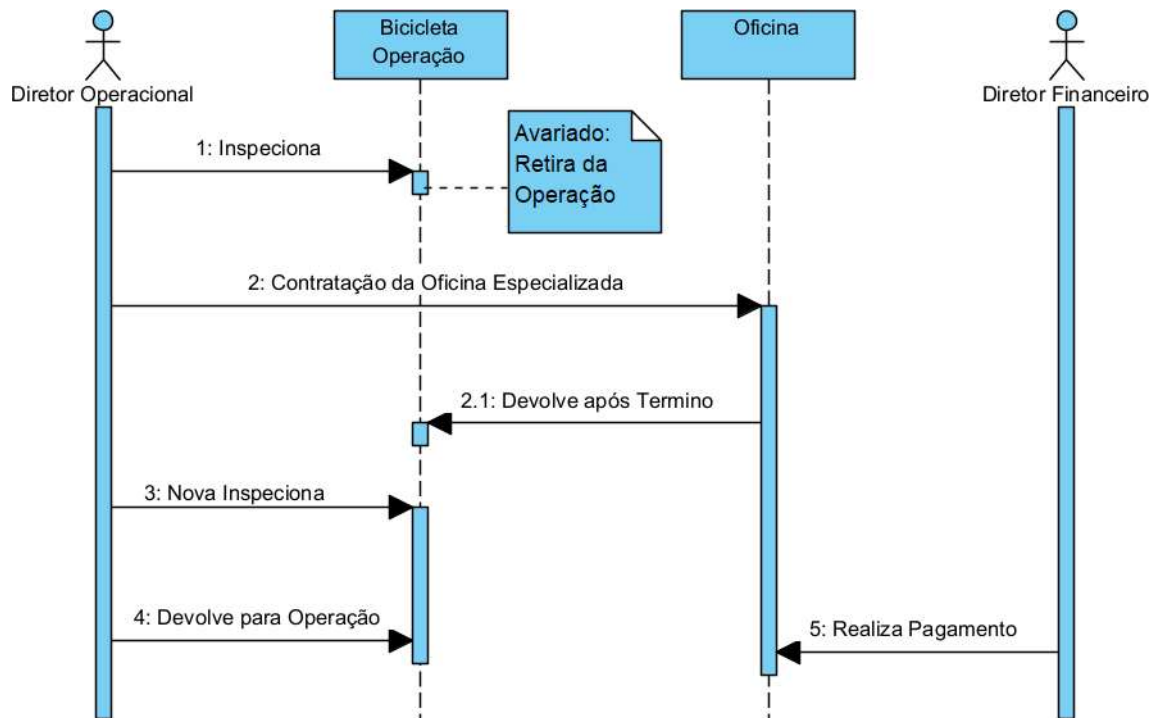


Figura 22 - Diagrama de Sequência - Visual Paradigm

6.2.3. Implementação e testagem das US

Revisoes:

id	vei	fun	data levantamento	data retorno	entregue	custo
1	1	1	07/01/2023 13:27:01	07/01/2023 13:27:01	True	34,39

Figura 23 - Teste de Revisão - Visual Studio - console

7. Sprint 07 – Dev Sprint

7.1. CdU mais prioritários (escolher 1 ou 2)

Ator principal: **Diretor Financeiro**

Nome do Use Case: **Realiza Pagamento**

Breve descrição: Responsável por fazer todos os pagamentos financeiros.

Nome do Use Case: **Consulta Mapas de Venda**

Breve descrição: Pode consultar o mapa de vendas a qualquer instante.

7.1.1. Narrativas dos CdU: fluxo principal e fluxos alternativos

Figura 24 - Fluxo Principal e Alternativos - Pagamento e Mapa de Venda

Fluxo Principal	Fluxo Alternativo
FP1 - Recebe pedido de compra	FA1 - Caso não exista recurso reporta ao socios
FP2 - Recebe Autorização para pagamento	
FP3 - Realizada Pagamento	
FP3 - Consulta Mapa de Vendas	

7.1.2. User Stories (US)

A ordem de compra é enviada diariamente, organizo por prioridades e, após receber autorização realiza-se o pagamento. Tenho que consultar o mapa de venda para fazer previsões do que resta pagar e valores a receber, caso não tenha recurso em caixa, devo solicitar/reportar aos sócios.

7.2. Análise, implementação e teste de cada US

```

/// <summary>
/// recebe um tipo de veiculo e um aluguer
/// cria 2 variaveis ct e cb do tipo double
/// atribui um custo base (cb) com base no tipo do veiculo
/// calcula diferença entre as datas
/// calcula custo por hora e atribui o valor minimo automaticamente
/// chama a função InsereCustoBD
/// </summary>
/// <param name="tipo"></param>
/// <param name="a"></param>
/// <returns></returns>
1 reference
public static bool CalculaCusto(TIPO tipo, Aluguer a)
{
    double ct = 0, cb = 0; //custo total e custo base
    switch (tipo) {
        case (TIPO.Bicicleta_Normal): cb = 1;
            break;
        case (TIPO.Bicicleta_Eletrica): cb = 1.5;
            break;
        case (TIPO.Trotinete_Eletrica): cb = 1.5;
            break;
        default: return false;
    }

    TimeSpan dif = a.dataRet.Subtract(a.dataLev);
    if (dif.Minutes > 0)
        ct = (dif.TotalHours + 1) * cb + cb;
    else ct = dif.TotalHours * cb + cb;

    DL.DadosAluguer.InsereCustoBD(ct, a);

    return true;
}

```

7.2.1. Mockups para capturar os requisitos de interface

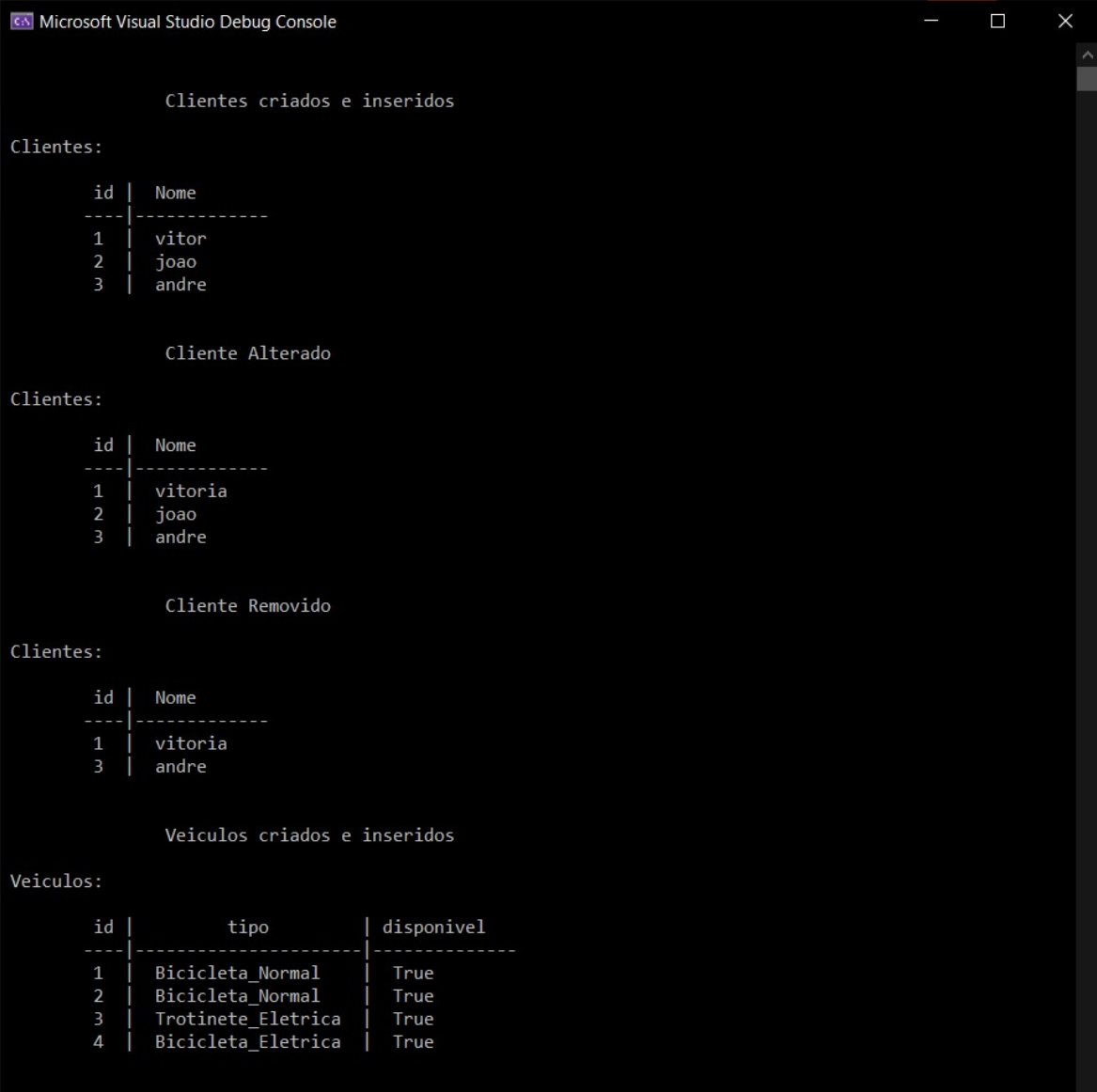


Figura 25 - Mockup pagamento

7.2.2. Diagrama de Classes

Com o desenvolvimento deste projeto achamos, após uma análise, não ser necessário fazer quaisquer atualizações ao diagrama de classes desenvolvido anteriormente, que pode ser visto no ponto [4.3](#).

7.2.3. Implementar e testar as US: não há restrições quanto à tecnologia



```
Microsoft Visual Studio Debug Console

    Clientes criados e inseridos
Clientes:

  id | Nome
  ---|-----
  1  | vitor
  2  | joao
  3  | andre

    Cliente Alterado
Clientes:

  id | Nome
  ---|-----
  1  | vitoria
  2  | joao
  3  | andre

    Cliente Removido
Clientes:

  id | Nome
  ---|-----
  1  | vitoria
  3  | andre

    Veiculos criados e inseridos
Veiculos:

  id |      tipo      | disponivel
  ---|-----|-----
  1  | Bicicleta_Normal | True
  2  | Bicicleta_Normal | True
  3  | Trotinete_Eletrica | True
  4  | Bicicleta_Eletrica | True
```

Figura 26 - teste em consola pt.1

```

Microsoft Visual Studio Debug Console

Veiculo Alterado

Veiculos:

  id |      tipo      | disponivel
----|-----|-----
  1 | Bicicleta_Normal | True
  2 | Bicicleta_Eletrica | True
  3 | Trotinete_Eletrica | True
  4 | Bicicleta_Eletrica | True

Veiculo Removido

Veiculos:

  id |      tipo      | disponivel
----|-----|-----
  1 | Bicicleta_Normal | True
  3 | Trotinete_Eletrica | True
  4 | Bicicleta_Eletrica | True

Funcionarios criados e inseridos

Funcionarios:

  id | Nome
----|-----
  1 | Joaquim
  2 | Antonio

Funcionario alterado

Funcionarios:

  id | Nome
----|-----
  1 | José
  2 | Antonio
    
```

Figura 27 - testes em consola pt.2


```
Microsoft Visual Studio Debug Console

Funcionarios removido

Funcionarios:

  id | Nome
  ---|-----
  1  | José

tenta levantar um veiculo com um id de cliente nao existente

ERRO: CLIENTE NAO EXISTE

tenta levantar um veiculo que nao existe

ERRO: VEICULO NAO EXISTE

Alugueres Criados e inseridos

Alugueres:

  id | vei | cli | data levantamento | data retorno | entregue | custo
  ---|---|---|-----|-----|-----|-----
  1  | 1  | 1  | 08/01/2023 13:50:14 | ----- | False   | 0
  2  | 3  | 3  | 08/01/2023 13:50:14 | ----- | False   | 0
  3  | 4  | 1  | 08/01/2023 13:50:14 | ----- | False   | 0

Veiculos:

  id | tipo | disponivel
  ---|---|-----
  1  | Bicicleta_Normal | False
  3  | Trotinete_Eletrica | False
  4  | Bicicleta_Eletrica | False

tenta devolver um veiculo que nao existe

ERRO: VEICULO NAO EXISTE
```

Figura 28 - testes em consola pt.3

```

Microsoft Visual Studio Debug Console

Alugueres:

  id | vei | cli | data levantamento | data retorno | entregue | custo
-----|-----|-----|-----|-----|-----|-----
  1 | 1 | 1 | 08/01/2023 13:50:14 | 08/01/2023 13:50:14 | True | 0
  2 | 3 | 3 | 08/01/2023 13:50:14 | 08/01/2023 13:50:14 | True | 0
  3 | 4 | 1 | 08/01/2023 13:50:14 | ----- | False | 0

Veiculos:

  id | tipo | disponivel
-----|-----|-----
  1 | Bicicleta_Normal | True
  3 | Trotinete_Eletrica | True
  4 | Bicicleta_Eletrica | False

tenta levantar veiculo que nao existe para revisao

ERRO: VEICULO NAO EXISTE

tenta levantar veiculo para revisao com id funcionario que nao existe

ERRO: FUNCIONARIO NAO EXISTE

Revisao Iniciada

Revisoes:

  id | vei | fun | data levantamento | data retorno | entregue | custo
-----|-----|-----|-----|-----|-----|-----
  1 | 1 | 1 | 08/01/2023 13:50:14 | ----- | False | 0

Veiculos:

  id | tipo | disponivel
-----|-----|-----
  1 | Bicicleta_Normal | False
  3 | Trotinete_Eletrica | True
  4 | Bicicleta_Eletrica | False

tenta retornar veiculo que nao existe

ESTE VEICULO NAO EXISTE

```

Figura 29 - testes em consola pt.4

```

Microsoft Visual Studio Debug Console

Revisao Terminada

Revisoes:

  id | vei | fun | data levantamento | data retorno | entregue | custo
-----|-----|-----|-----|-----|-----|-----
  1 | 1 | 1 | 08/01/2023 13:50:14 | 08/01/2023 13:50:14 | True | 34,39

Veiculos:

  id | tipo | disponivel
-----|-----|-----
  1 | Bicicleta_Normal | True
  3 | Trotinete_Eletrica | True
  4 | Bicicleta_Eletrica | False

```

Figura 30 - testes em consola pt.5

Conclusão

A elaboração e estrutura de um software não baseia se apenas no implementar códigos de uma linguagem específica, a codificação é um passo importantíssimo, entretanto, não é o único, pois existem outros aspetos indispensáveis, como exemplo o levantamento de requisitos, diagrama de caso de uso, diagrama BPMN, diagrama de classe, diagrama de sequencia, diagrama de estado, são alguns dos modelos que nos ajudam no desenvolvimento de um solução, ou seja, uma solução sem os modelos acima identificados é comparado a um comboio desgovernado com a máxima velocidade, uma catástrofe iminente, deste modo, com o objetivo de incentivar a criação e o desenvolvimento pessoal do individuo, o instituto politécnico IPCA, no curso de Engenharia de sistemas informáticos, pretende criar solucionar através da linguagem C# para uma solução de mobilidade.

Com a Supervisão do Professor João Pedro Barbosa da Silva, os alunos deverão elaborar um trabalho de Análise e Modelação de Software dividido em sete sprint a fim de solucionar a questão acima identificada.

Concluimos que o trabalho realizado foi grande importância pois obtivemos excelentes avanços no aprendizado de forma dinâmica e funcional.

Bibliografia

<https://bitbucket.org/ams-ws22/canteen22/wiki/SP04.ProjectPlan.md>

https://pt.wikipedia.org/wiki/Business_Process_Model_and_Notation

<https://pt.wikipedia.org/wiki/UML>

<https://www.visual-paradigm.com/>

<https://visualstudio.microsoft.com/>