Relatório de Projeto

Nome do projeto: Easy Parking

GitHub do projeto: https://github.com/RenatoJunior98/EasyParking

Link para a plataforma: https://easy--parking.herokuapp.com/

Engenharia Informática 3ºAno

Trabalho realizado por:

Renato Júnior, 50039475

Ricardo Cruz, 50039470

Descrição do projeto

Sistema de monitoração de lugares de parques de estacionamento, sabendo toda a informação disponível dos parques, como nº de lugares total, horários, localização, morada, tabela de preços e ainda a implementação de reviews para que outros utilizadores comparem a avaliação de cada parque. Indica a qualquer momento o número de lugares disponíveis de um dado parque de estacionamento. É também possível pesquisar os parques pelo nome e receber a rota localização do utilizador até ao parque escolhido. Algumas destas funcionalidades requerem que o utilizador tenha uma conta registada e login feito na plataforma.

Enquadramento do projeto

Numa rotina diária bem definida onde as pessoas têm que sair de casa mais cedo para procurar um parque de estacionamento com lugares disponíveis, o nosso projeto vem para quebrar esta norma e indicar de antemão onde se encontram parques com lugares disponíveis e de seguida, a possibilidade de fazer uma reserva.

Pesquisa de mercado

Após uma pesquisa do mercado encontramos três aplicações com funcionalidades semelhantes às que pretendemos ter neste projeto.

APP	Vantagens	Desvantagens
EMEL ePark. Agora mais simples Mais de 100 mil downloads e 2,5☆ (0 a 5, PlayStore)	• Atualizações semanais; • Pagamento mensal dos parques; • Sistema de dísticos; • Mostra na app o local do veículo e do parque.	• Não tem informação dos lugares • disponíveis (apenas total); • Lisboa é a única cidade presente na base de dados; • Impossível reservar ou comprar • bilhetes diários (apenas mensal). • Segundo alguns utilizadores a app crasha frequentemente.
Parclick – Find and Book Parking Spaces Mais de 100 mil downloads e 4,4☆ (0 a 5, PlayStore)	• Interface intuitiva e apelativa; • 1400 parques disponíveis em 7 países e 250 cidades; • Informação dos parques (horários, preços, tipos de veículos permitidos, altura máxima permitida, descrição do parque, morada, como chegar ao parque e reviews de utilizadores; • Opção de reserva e descontos.	• Não tem informação de lugares disponíveis (apenas quando se encontra lotado).
Via Verde Estacionar- Mais de 100 mil downloads e 4,3☆ (0 a 5, PlayStore)	• Facilidade de utilização (acessível para todas as idades); • 1400 parques disponíveis em 7 países e 250 cidades; • Informação dos parques (horários, preços, tipos de veículos permitidos, altura máxima permitida, descrição do parque, morada, como chegar ao parque e reviews de utilizadores; • Opção de reserva e descontos;	• Não tem informação de lugares disponíveis (apenas quando se encontra lotado).

Objetivos do projeto

- Base de dados eficiente e em constante atualização;
- Informação completa dos parques;
- Sistema de localização com informação dos parques e dos utilizadores;
- Ter uma interface intuitiva;
- Aplicação web rápida e eficaz;
- Receber dados sobre parques da EMEL.

Proto-Personas

Andreia

Ocupada

SOBRE

Idade: 25-45
Género: F
Estado civil: Casada
Área de emprego: Marketing
Salário: 1.000/mês
Condições monetárias: Médias

NECESSIDADES

Lugar vago para poder fazer as suas tarefas tanto domésticas como de trabalho

FUNCIONALIDADES UTILIZADAS

A principal funcionalidade utilizada pela a Andreia é a reserva de lugar perto do local onde antes de sair de casa.

MOTIVAÇÕES

Poder fazer as suas tarefas sem se preocupar com a lotação do estacionamento.

"PAIN POINTS"

Gastar tempo à procura de lugar

Rodrigo

Prudente

SOBRE

Idade: 30-55
Género: M
Estado civil: Casado
Área de emprego: Engenharia de software
Salário: 2.200/mês
Condições monetárias: Boas

NECESSIDADES

Lugar vago para estacionar ao pé do trabalho.

FUNCIONALIDADES UTILIZADAS

A principal funcionalidade utilizada pelo Rodrigo é a reserva de lugar perto do emprego antes de sair de casa.

MOTIVAÇÕES

Estacionar o veículo ao pé do trabalho sem ter o risco de o parque estar lotado.

"PAIN POINTS"

Chegar atrasado ao trabalho por falta de estacionamento perto do trabalho.

Casos de Utilização

Nome:
Selecionar Estacionamento
Descrição:
O sistema ativa a interface de seleção de parque que apresenta a lista de parques, o User escolhe um parque (escolhe através da lista ou do mapa apresentado) para ser ativada uma interface com a sua informação (nome, classificação de 1 a 5 estrelas e reviews deixadas por outros utilizadores, tipologia, número de lugares disponíveis, totais comuns e prioritários, horário de funcionamento e localização geográfica) e opções de reserva, indicar direções ou deixar review.
Pré-condições:
Interface de seleção de parques ativa.
Passo a passo:
1- Sistema ativa a interface de seleção de parques; 2- Sistema pesquisa e apresenta a lista de parques disponíveis com informação do nome, lugares disponíveis e preço diário; 3- Sistema faz pedido (ajax) de informação do mapa ao Leaflet API; 4- Leaflet API responde com pacote GEOJSON com informação do mapa; 5- Sistema Apresenta o mapa do pacote GEOJSON; 6- User clica no parque desejado (na lista ou no marker do parque no mapa); 7.1- User clica em "reservar Lugar"; 7.2- User clica em "indicar direções"; 7.3- User clica em "Deixar review".
Pós-condições:
Fica aberta a interface do parque selecionado.

Nome:
Reservar Lugar
Descrição:
Na interface de informação do parque um user autenticado seleciona a data desejada e de seguida a opção fazer reserva o sistema de parques trata de criar dados de pagamento (referência MB) e o sistema apresenta-os ao User, após o sistema de parques confirmar que o pagamento foi feito, o sistema reserva um lugar no estacionamento no dia selecionado pelo user (diminui o número de lugares disponíveis por 1) e apresenta o código de reserva ao User. O User com esse código pode utilizar a reserva.
Pré-condições:
User clica em "Fazer Reserva" na interface de informação do parque; User selecionou um parque com pelo menos 1 lugar disponível; User autenticado;

Passo a passo:

1- User seleciona o dia desejado no mapa apresentado;
2- User clica em "Fazer Reserva";
3- Sistema ativa popup de confirmação de reserva;
4- User clica em "OK";
5- Sistema envia informação de pedido de reserva (id do parque) ao sistema de parques;
6- Sistema de parques envia detalhes de pagamento;
7- Sistema apresenta detalhes de pagamento ao User;
8- Sistema de parques envia confirmação de pagamento;
9- Sistema ativa popup com mensagem de reserva feita com sucesso;

Pós-condições:

Número de lugares disponíveis do estacionamento selecionado é reduzido por um.
Dados da reserva são guardados na base de dados.

Nome:

Indicar direções

Descrição:

User introduz a sua localização atual ao clicar no mapa e o sistema apresenta no mesmo a uma linha que representa as direções da sua localização selecionada até à localização do parque.

Pré-condições:

Interface de informações do parque selecionado ativa.

Passo a passo:

1- Sistema ativa a interface de seleção de parques;
2- Sistema pesquisa e apresenta a lista de parques disponíveis com informação do nome, lugares disponíveis e preço diário;
3- Sistema faz pedido (ajax) de informação do mapa ao Leaflet API;
4- Leaflet API responde com pacote GEOJSON com informação do mapa;
5- Sistema Apresenta o mapa do pacote GEOJSON;
6- User clica no parque desejado (na lista ou no marker do parque no mapa);
7.1- User clica em "reservar Lugar";
7.2- User clica em "indicar direções";
7.3- User clica em "Deixar review".

Pós-condições:

Fica aberta a interface do parque selecionado.

Definição final das Personas

Rodrigo - Prudente

"Mais vale prevenir do que remediar"

Idade: 30-55
Emprego: Engenheiro de software
Estado civil: Casado
Localização: Portugal, Porto

Salário: 2.200/mês
Condições monetárias: Boas

Personalidade

Introvertido — Extrovertido
Pensamentos — Sentimentos
Sensação — Intuição
Calculista — Sonhador

Objetivos

- Conseguir subir na empresa e ser coordenadores de projetos na sua área.
- Alcançar o seu sonho de trabalhar na apple ou microsoft.
- Comprar e viver na sua casa de sonho com a sua família.

Frustrações

- Falta de entendimento com o seu chefe.
- Falta de tempo para estar com a família.

Bio

Rodrigo é um engenheiro de software talentoso e com grande potencial. É fluente em várias linguagens de programação como: Java, Phyton e C++.

O maior problema do dia a dia de Rodrigo é o tempo que perde à procura de lugar para estacionar o seu carro. Para que chegar atrasado seja um hábito, Rodrigo decide utilizar o EasyParking para reservar um lugar num parque perto do seu emprego. Assim, pode melhorar a sua relação com o seu chefe e largar um peso das suas costas.

Funcionalidades utilizadas

Qual a principal funcionalidade utilizada pelo Rodrigo

Com o EasyParking, Rodrigo pode reservar um lugar antecipadamente como também escolher um parque barato e seguro para estacionar o carro.

Dispositivos

Smartphone, computador pessoal, computador da empresa, smartwatch.

Motivações

Incentivo
Medo
Poder
Social

Referências e Influências

Maria - Viajante

"Trabalhar durante o ano para ter umas boas férias"

Idade: 25-45
Emprego: Advogada
Estado civil: Solteira
Localização: Portugal, Lisboa
Salário: 1.300/mês

Condições monetárias: Boas

Personality

Introvertida — Extrovertida
Pensamentos — Sentimentos
Sensação — Intuição
Calculista — Sonhadora

Objetivos

- Subir na carreira e ser procuradora geral ou outro cargo importante.
- Trabalhar num caso que lhe impulsione a carreira.
- Casar e criar família.
- Fazer uma viagem à volta do mundo.

Frustrações

- É raro ter férias durante o ano, e quando tem é pouco tempo.
- Não tem muito tempo livre para conhecer outras pessoas e fazer atividades que ela gosta.

Bio

A Maria sempre foi das melhores alunas da sua turma, como tal seguiu por um caminho profissional na área de direto, onde foi bem sucedida. Ultimamente acha que deveria ter mais tempo para fazer a coisa que ela mais gosta: viajar. Tendo poucas férias, a última coisa que ela quer gastar o seu tempo é a estacionar. Por esse motivo a Maria utiliza a aplicação EasyParking para procurar estacionamentos mais convenientes, práticos e ainda poupando tempo.

Funcionalidades utilizadas

Qual a principal funcionalidade utilizada pela Maria

Com o EasyParking, a Maria pode procurar por parques perto dos locais onde quer visitar e ainda não perde tempo à procura de lugar para estacionar.

Dispositivos

Smartphone, smartwatch, computador pessoal, computador da empresa, tablet

Motivações

Incentivo
Medo
Poder
Social

Brands & Influencers

Andreia - "Shopaholic"

"A felicidade mede-se na quantidade de sapatos"

Idade: 25-40
Emprego: Markting
Estado civil: Solteira
Localização: Portugal, Lisboa

Salário: 1.000/mês
Condições monetárias: Médias

Personality

Introvertida	Extrovertida
Pensamentos	Sentimentos
Sensação	Intuição
Calculista	Sonhada

Objetivos

- Ser bem sucedida na sua carreira.
- Casar e criar família.
- Trabalhar num projeto importante e com visibilidade.

Frustrações

- Ainda está num ponto inicial da sua carreira.
- Não consegue controlar o seu vício de compras.

Bio

A Andreia é uma jovem adulta que trabalha pequena empresa local. É inteligente e versátil no trabalho, mas o tem um vício incontrolável: é obcecada em compras principalmente em lojas físicas.

A maior parte do tempo perdido é a encontrar lugar perto das suas lojas favoritas. Para diminuir esse tempo perdido diariamente, a Andreia utiliza a aplicação EasyParking. Assim esse tempo perdido pode-se transformar em tempo útil.

Funcionalidades utilizadas

Qual a principal funcionalidade utilizada pela Andreia

Com o EasyParking, Andreia pode reservar um lugar junto ás suas lojas, para estacionar o seu carro.

Dispositivos

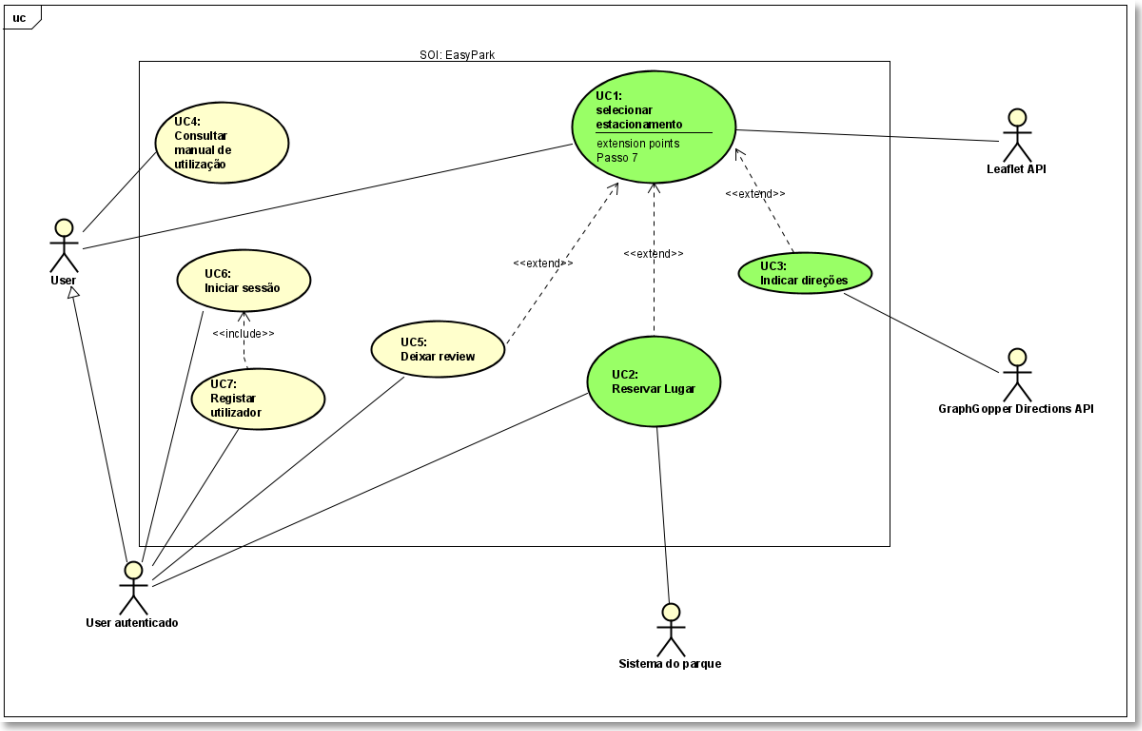
Smartphone, computador pessoal, tablet.

Motivação

Incentivo
Medo
Poder
Social

Brands & Influencers

Diagrama de casos de utilização



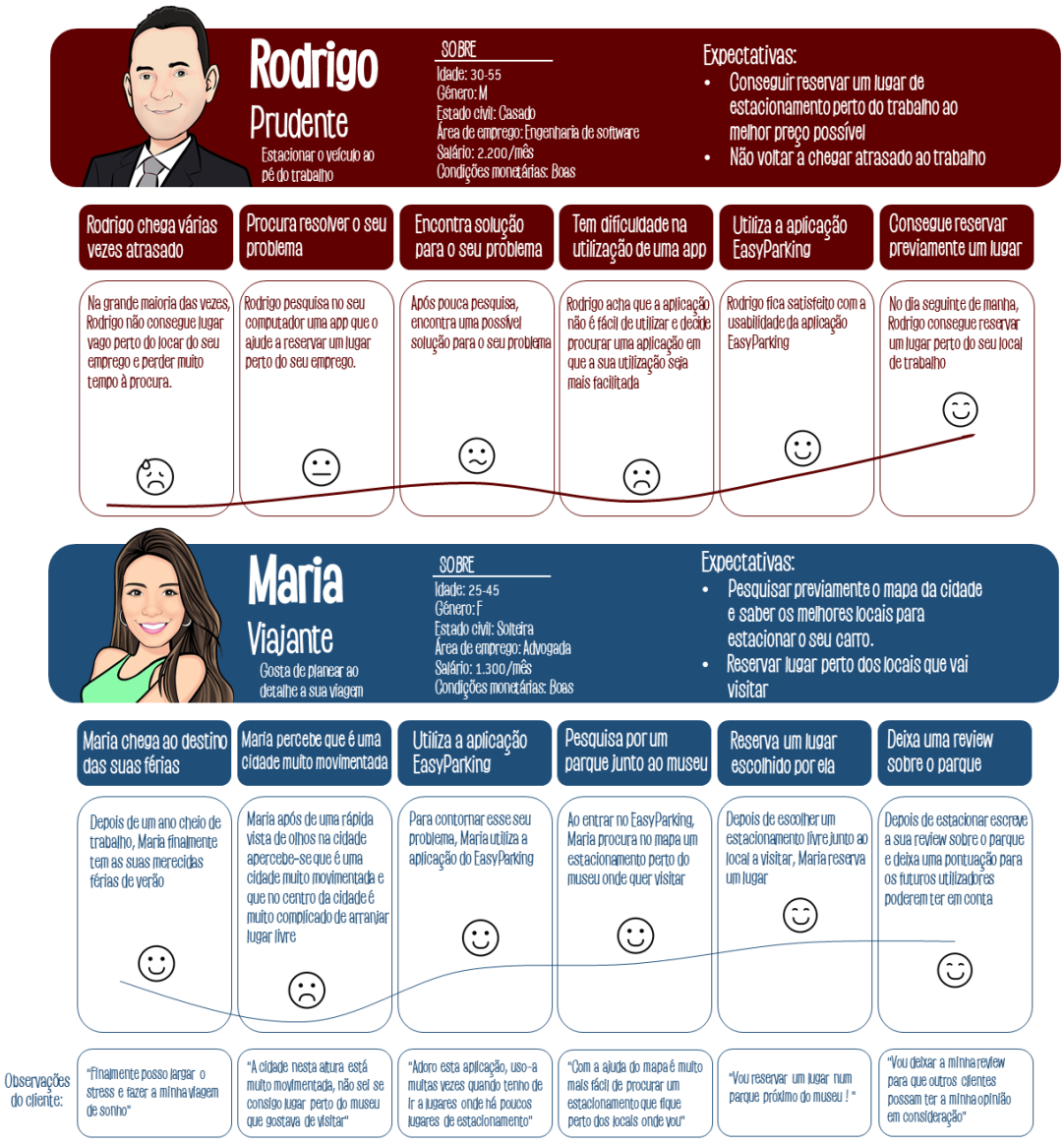
Requisitos funcionais

#	Nome do Requisito	Descrição	Pri.
FR01	Apresentar lista de parques	Apresenta ao User uma lista com os parques com alguma informação disponível (nome, indicação se o parque é coberto ou descoberto, altura máxima permitida, se tem videovigilância, classificação média de 1 a 5 estrelas dada por outros utilizadores).	High
FR02	Apresentar parques no mapa	Apresenta ao User um mapa com a localização dos parques.	Low
FR03	Selecionar parque	User seleciona um parque e o sistema ativa a interface do parque selecionado que apresenta a sua informação disponível (nome, classificação de 1 a 5 estrelas e reviews deixadas por outros utilizadores, tipologia, descrição, restrições dos veículos, número de lugares disponíveis comuns e prioritários, horário de funcionamento e localização geográfica).	High
FR04	Apresentar informação do parque	Sistema apresenta ao User a informação disponível dos parques disponível (nome, classificação de 1 a 5 estrelas e reviews deixadas por outros utilizadores, tipologia, descrição, restrições dos veículos, número de lugares disponíveis comuns e prioritários, horário de funcionamento e localização geográfica).	Medium
FR05	Reservar lugar	Número de lugares disponíveis do estacionamento selecionado é reduzido por um.	High
FR06	Gerar código de reserva	Sistema gera um código de reserva único associado ao parque selecionado e ao user.	High
FR07	Apresentar direções	Sistema apresenta as direções do user até ao parque selecionado.	High
FR08	Guardar dados introduzidos da review	Sistema guarda na BD o comentário e a classificação introduzida pelo utilizador ao parque.	High
FR09	Autenticação de utilizador	Sistema confirma se os dados de login são válidos e autoriza acesso a casos de utilização que não seriam acessíveis sem a autenticação.	High
FR10	Editar dados de utilizador	Utilizador pode alterar os dados pessoais a qualquer momento e essas alterações são guardadas na BD.	High
FR11	Confirmação de pagamento	Sistema recebe a confirmação do pagamento vindo do sistema dos parques.	High
FR12	Autenticar código	Sistema confirma se o código da reserva introduzido pelo utilizador se verifica na base de dados e a reserva correspondente tem o estado "ativa"(esta paga e o código foi introduzido no dia selecionado no ato da reserva)	High
FR13	Receber geolocalização do utilizador	Utilizador seleciona a sua localização para ser mostrado no mapa o trajeto até ao parque	High
FR14	Gestão do estado dos parques	Consoante o horário do parque o sistema indica se este está "Disponível" (aberto e com lugares livres), "Indisponível"(aberto, mas sem lugares livres) ou "Fechado"(fora do horário de funcionamento).	High

Requisitos não funcionais

#	Nome do Requisito	Descrição	Pri.
NFR01	Desenvolvimento multiplataforma	O sistema deverá ser desenvolvido para qualquer browser	Medium
NFR02	Ligação à internet	O utilizador deverá ter uma ligação à internet estável enquanto estiver a utilizar a plataforma.	High
NFR03	Comunicação com DB	O sistema deverá se comunicar com o banco SQL Server.	High
NFR04	Desenvolvimento em Node.js	O sistema deverá ser desenvolvido em Node.js.	High
NFR05	População da BD com dados fidedignos	A base de dados deverá ser preenchida com dados fidedignos.	Medium
NFR06	Possibilidade de adaptação	Sistema configurável para outros modelos de negócio e para outro conjunto de parques.	Medium

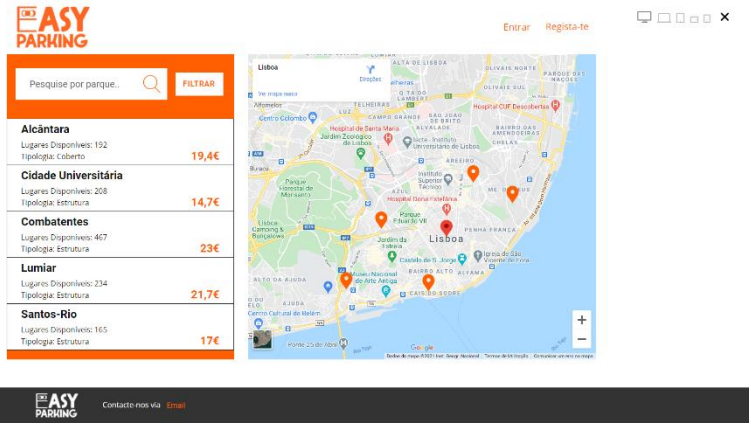
UX Journeys

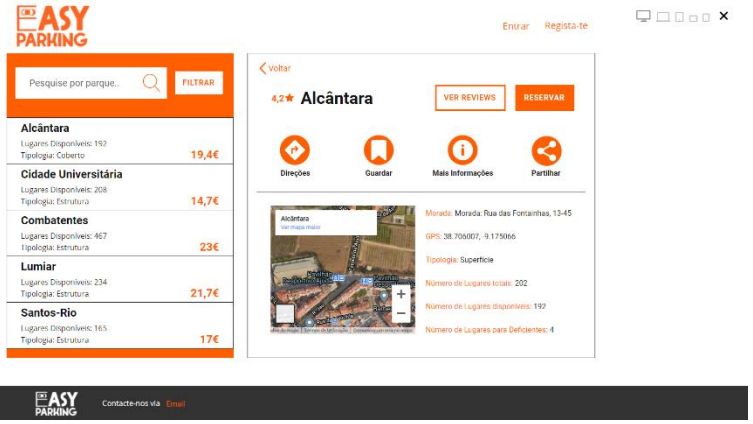


Especificação dos mockups

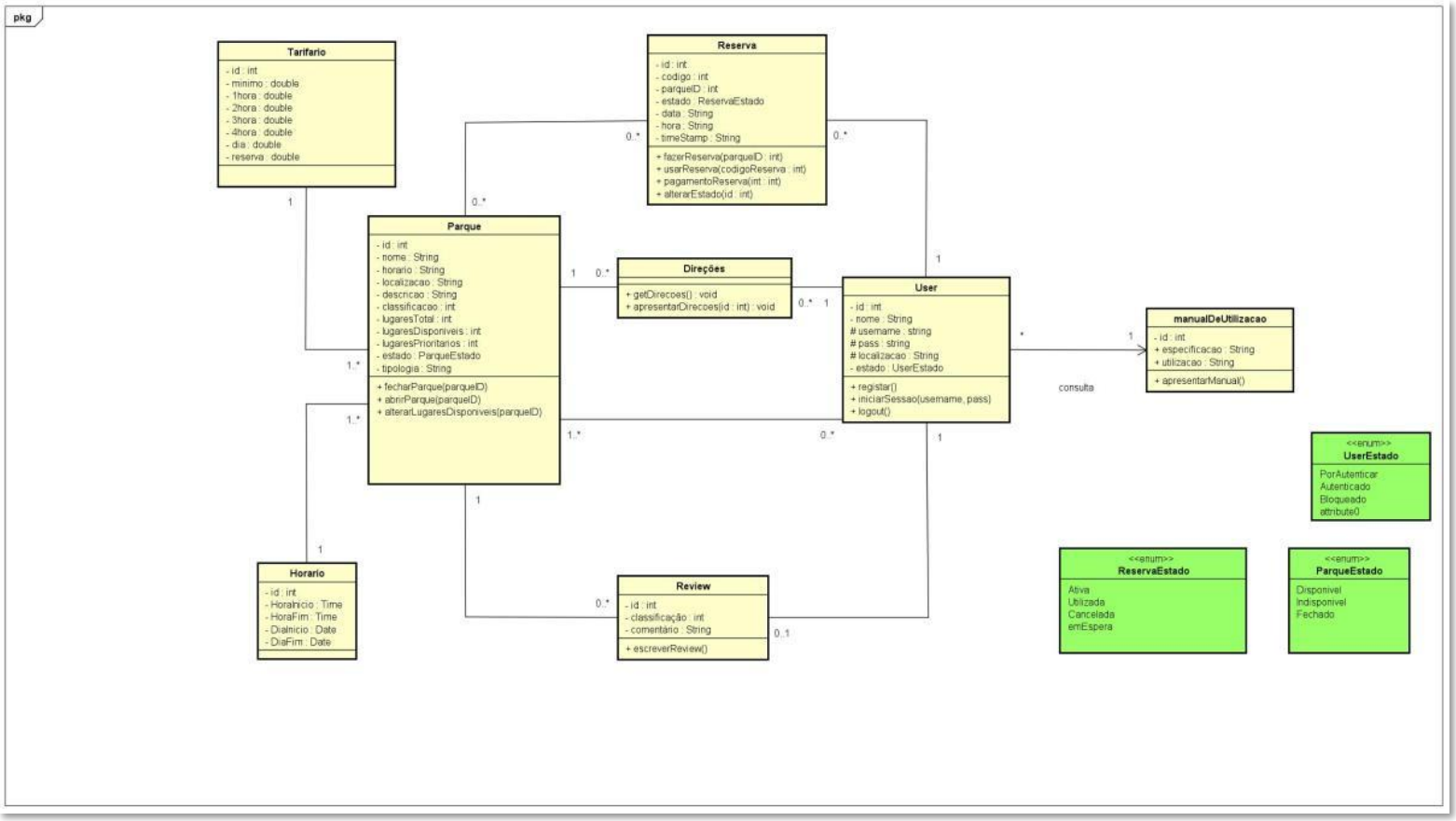
Página Index

Página com uma introdução sobre o site. Contém uma lista com as informações que os users podem ver de cada parque, uma breve introdução de como usar a plataforma e uma descrição "Sobre Nós". É ainda possível clicar no "Regista-te" para criar uma conta e no "Entrar" para fazer log in na sua conta.





Modelo de domínio



Calendário de implementação dos requisitos

Link para o diagrama de Gantt: https://share.clickup.com/g/h/4ard1-29/dcd4ae6a5522898

Software project plan

Começamos por criar mockups e conceitos de como será a melhor forma de interagir com o utilizador, de seguida vamos criar uma base de dados eficiente com informação completa dos parques com querys bem formuladas para que a interação da plataforma seja rápida e eficiente.

O próximo passo será trabalhar no código da aplicação web tendo em mente que o mais importante no início é ter os cenários principais a funcionar sem problemas.

Com esse objetivo atingido, desenvolvemos os casos de uso e funcionalidades de segundo plano. Com todas as funcionalidades descritas nos casos de uso funcionais

já desenvolvidas, trabalhamos na responsividade da aplicação web para dispositivos mobile (que segundo as nossas pesquisas e testes será uma mais valia).

Realizar testes ao longo do desenvolvimento e no fim para resolver possíveis bugs e problemas.

Finalmente com todas estas etapas efetuadas e com os testes realizados vamos poder lançar oficialmente a aplicação online.

- **Interfaces e Usabilidade**

Planear traçar a experiência do utilizador, será um trabalho constante do início ao fim do projeto.

Para definir as personas fazemos um questionário, assim conseguimos perceber melhor o nosso publico alvo e ao fazer as "UX journeys" para entender as necessidades dos nossos possíveis clientes.

Com as tarefas anteriores feitas, falta desenhar um protótipo e o esqueleto (Information Architecture) para seguirmos um guião (apesar de ser um draft) quando formos desenvolver a aplicação web.

No contexto desta unidade curricular a nossa principal preocupação é que a nossa aplicação web proporcione a melhor experiência de utilização possível e que agrade visualmente o cliente.

- **Programação WEB**

Escrever o código para criar e manusear a plataforma que desejamos, usaremos as linguagens HTML, JavaScript e CSS.

Para tal, começamos pela instalação e preparação das ferramentas necessárias, de seguida, começar a trabalhar nos casos de uso formulados em análise de sistemas, começando pelo core.

Com o UC core já desenvolvido, vamos implementar também os outros 2 casos mais importantes e, só no fim, as restantes funcionalidades.

Será feito também um trabalho de correção de erros e melhorias de eficiência desde que começamos a escrever o código até que esteja pronto a ser lançado online oficialmente (posteriormente, se necessário, serão corrigidos erros e bugs).

O trabalho realizado no contexto desta unidade curricular necessita de um trabalho prévio feito no contexto de outras unidades curriculares, assim já temos todo o conceito e planeamento do código antes de começar a escrevê-lo.

Objetivo principal: ter os 3 casos de utilização apresentados na proposta do projeto: Selecionar Estacionamento, Reservar Lugar e Indicar Direções.

- **Análise de Sistemas**

Planear e criar diagramas e conceitos de como será feito o projeto, inicialmente fazemos um diagrama de contexto para perceber as diferentes do projeto, de seguida um diagrama de casos de uso para sabermos como cada caso de uso irá funcionar e interagir com os diferentes atores, sendo estes humanos ou máquinas.

Continuamos com diagrama de modelo de domínio e máquinas de estado para planear melhor os diferentes estados do sistema como são alterados, assim temos em consideração os diferentes eventos e condições de transição de estado. Será feito também um levantamento de requisitos funcionais e não funcionais do sistema para ter uma melhor cobertura e nível de detalhe e assim uma boa caracterização do comprometimento e estrutura do sistema modelado. Por fim a definição do modelo de negócio com a ajuda do BPMN.

- **Sistemas de Informações Geográficos**

Funcionalidades implementadas:

- Mapa navegável com markers na localização de cada parque;
- Markers com popup "clicáveis" para entrar na página de informações do parque selecionado;
- Geocoding;
- Desenho da rota no mapa: Estando na página de informações do parque, o user clica num local do mapa. É colocado um marker nessa localização e ao mesmo tempo é traçado o desenho da rota.

Criar georreferenciação de conteúdo, começamos por instalar e preparar as ferramentas necessárias (incluindo pedir as chaves de acesso do leaflet e GraphHopper), de seguida começamos a procurar ou criar a informação para os mapas e a organizá-la por camadas (layers) para obtermos uma visualização de informação geográfica através de mapas.

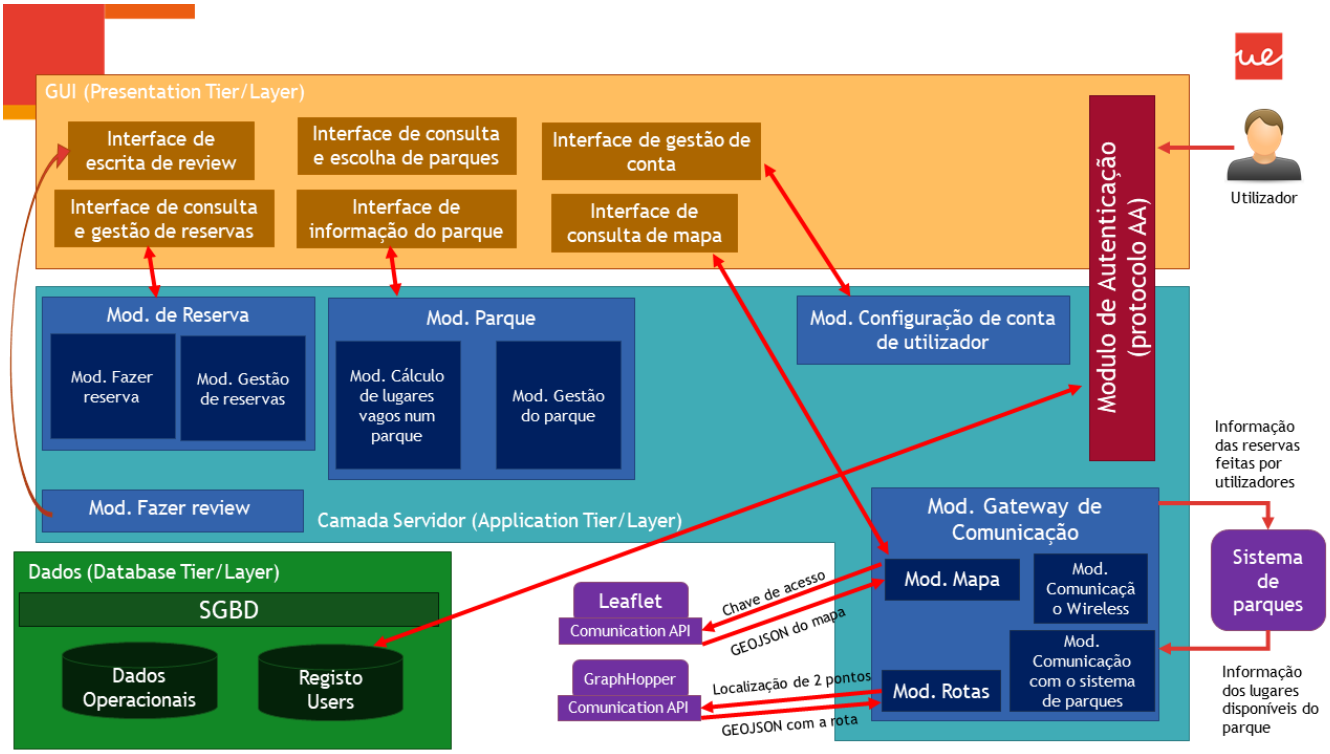
Uma das principais funcionalidades a implementar é o geocoding. Assim, o utilizador pode pesquisar por uma morada, nome de um ponto de interesse ou mesmo por coordenadas e encontrar um parque de estacionamento mais facilmente.

Por fim, implementar a funcionalidade que permite aos utilizadores verem a rota da localização do utilizador até ao parque selecionado. Para esta funcionalidade utilizamos o GraphHopper API. Para receber o GEOJSON que contem o desenho da rota, é necessário enviar pelo menos dois pares de coordenadas (localização do parque e localizações inseridas pelo utilizador).

Ferramentas utilizadas:

- GraphHopper: https://www.graphhopper.com/
- Leaflet: https://leafletjs.com/

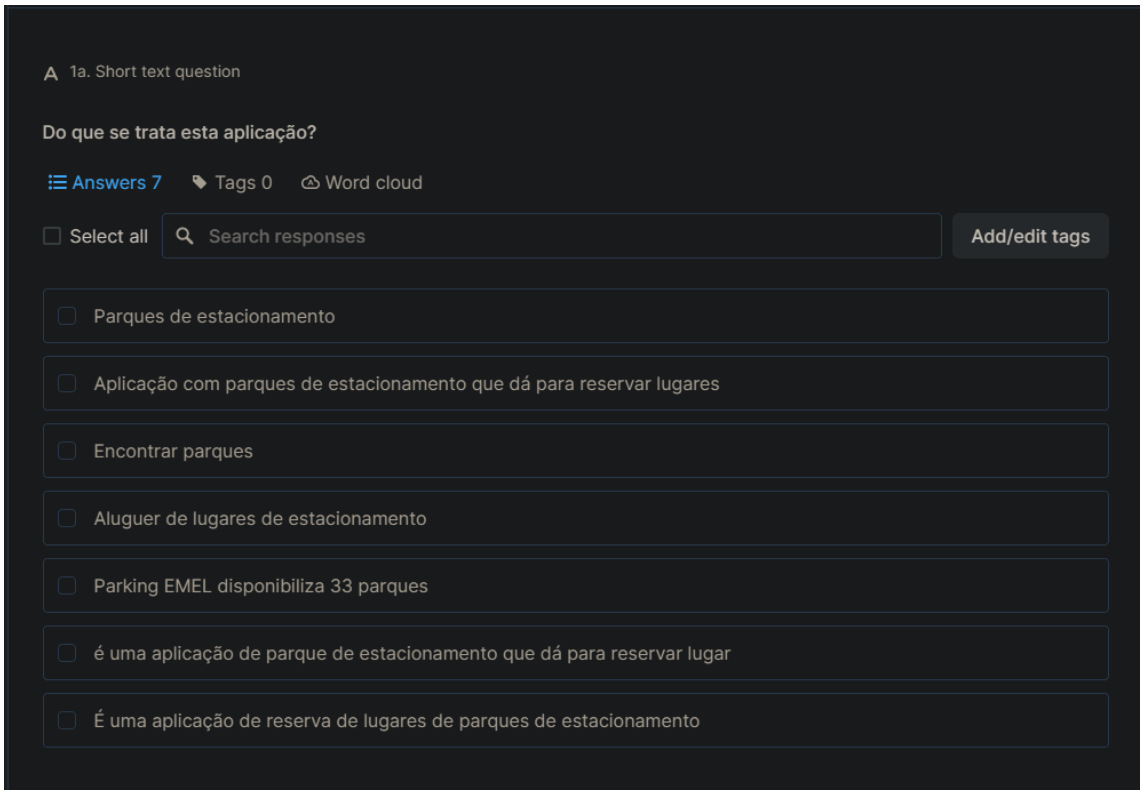
Arquitetura da Solução (Diagrama de blocos)



Testes de usabilidade e UX

Testes de usabilidade dos novos mockups:

Depois de apresentar o mockup da página index, perguntamos se conseguiam dizer do que se tratava esta aplicação.

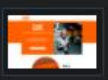


1. First Click

Onde clicaria para ver a lista dos parques

Image | Heat map | Click map

Export heat map as image



Other

12s 100% 7

1a. Linear scale question

Numa escala de 0 a 10, quão difícil foi encontrar o botão?

Mean: 1.3

Fácil — Difícil

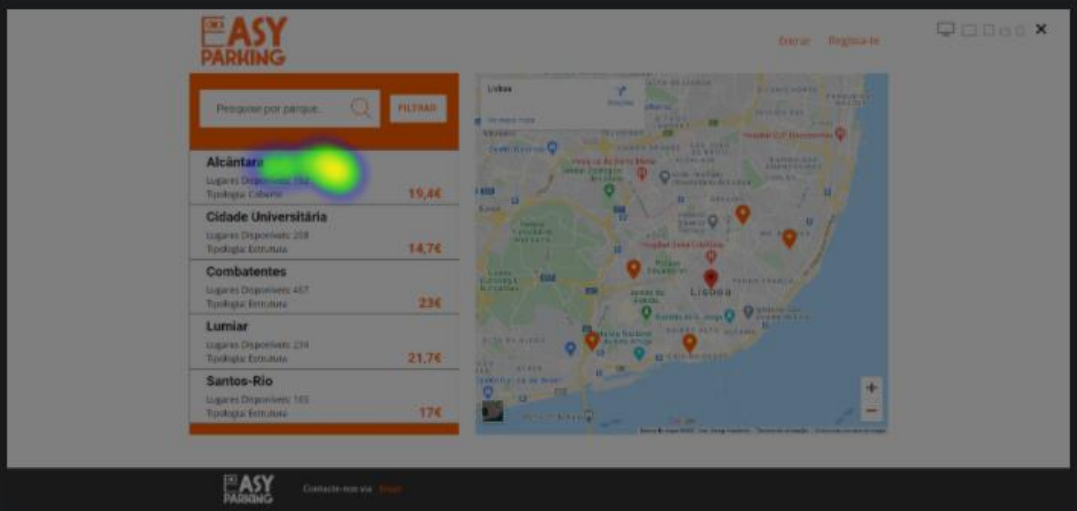
0	1	2	3	4	5	6	7	8	9	10
29%	29%	29%	14%	0%	0%	0%	0%	0%	0%	0%
2	2	2	1	0	0	0	0	0	0	0

1. First Click

Onde clicaria para ver a informação do parque de Alcântara?

Image | Heat map | Click map

Export heat map as image



Other — 5s — 100% — 7

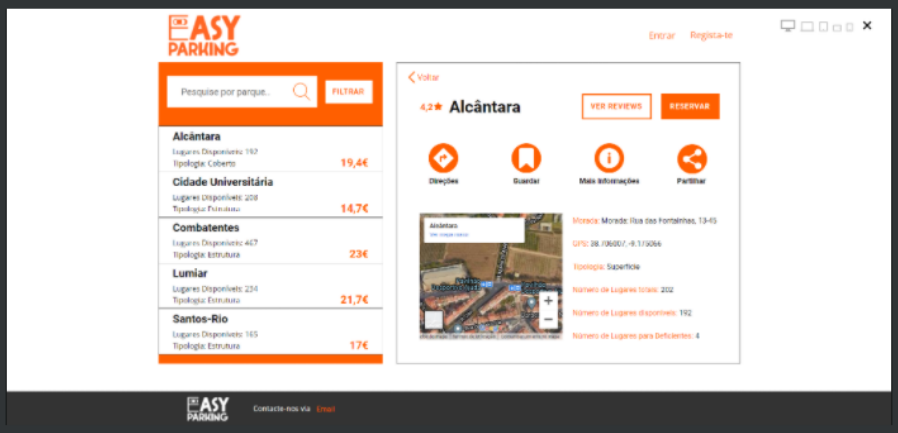
1a. Linear scale question

Numa escala de 0 a 10, quão difícil foi aceder à informação do parque?

Mean: 1.9

Fácil										Difícil
29%	29%	29%	0%	0%	0%	0%	14%	0%	0%	0%
2	2	2	0	0	0	0	1	0	0	0
0	1	2	3	4	5	6	7	8	9	10

1. Design Questions



1a. Short text question

Qual classificação média deste parque

Answers 7 | Tags 0 | Word cloud

Select all | Search responses | Add/edit tags

- 4,2
- 4,2
- 4.2
- 4,2
- 4,2
- 4.2
- 4,2

1b. Short text question

Quantos lugares disponíveis tem este parque no momento

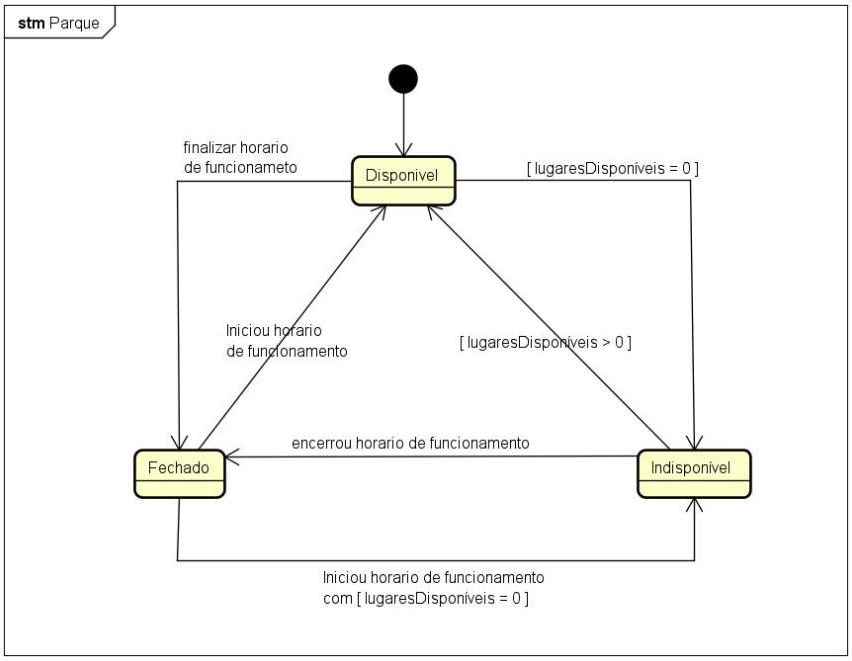
Answers 7 | Tags 0 | Word cloud

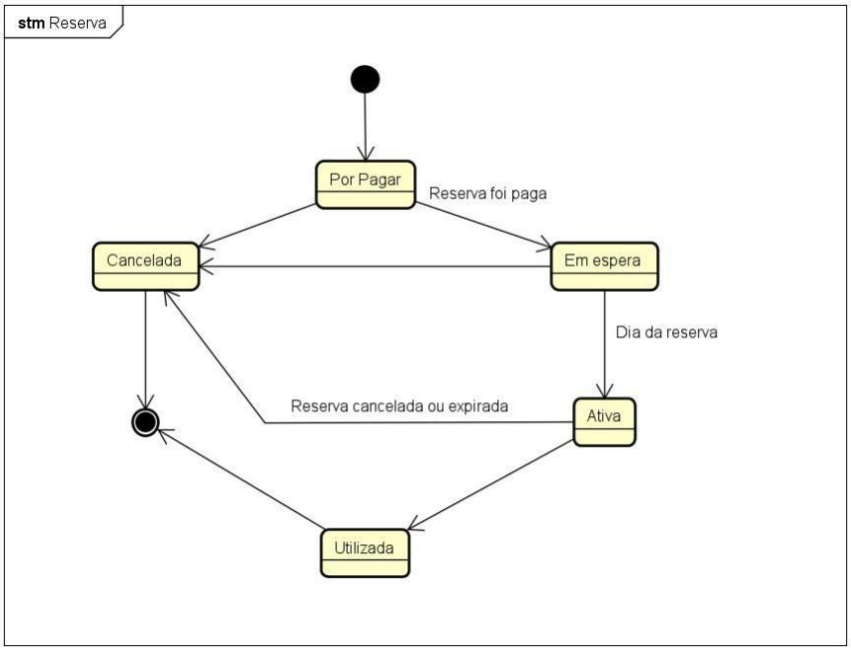
Select all | Search responses | Add/edit tags

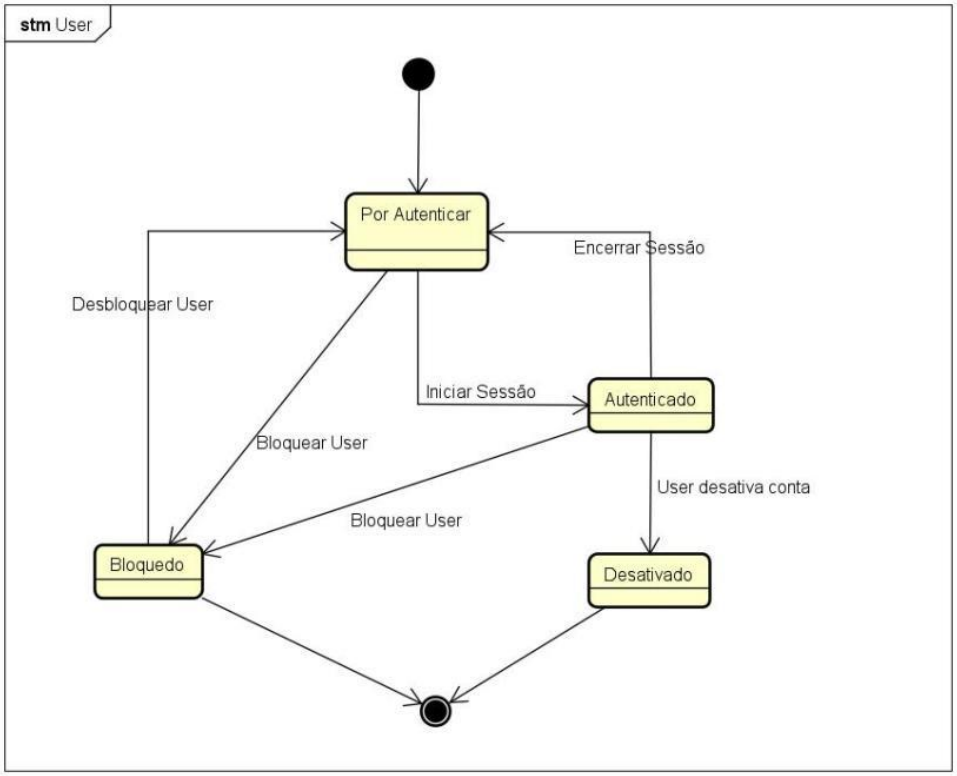
- 192
- 192
- 192
- 192
- 192
- 192
- 192

Mockups and early web interfaces: guia de estilos e padrões de design

Diagrama máquinas de estados



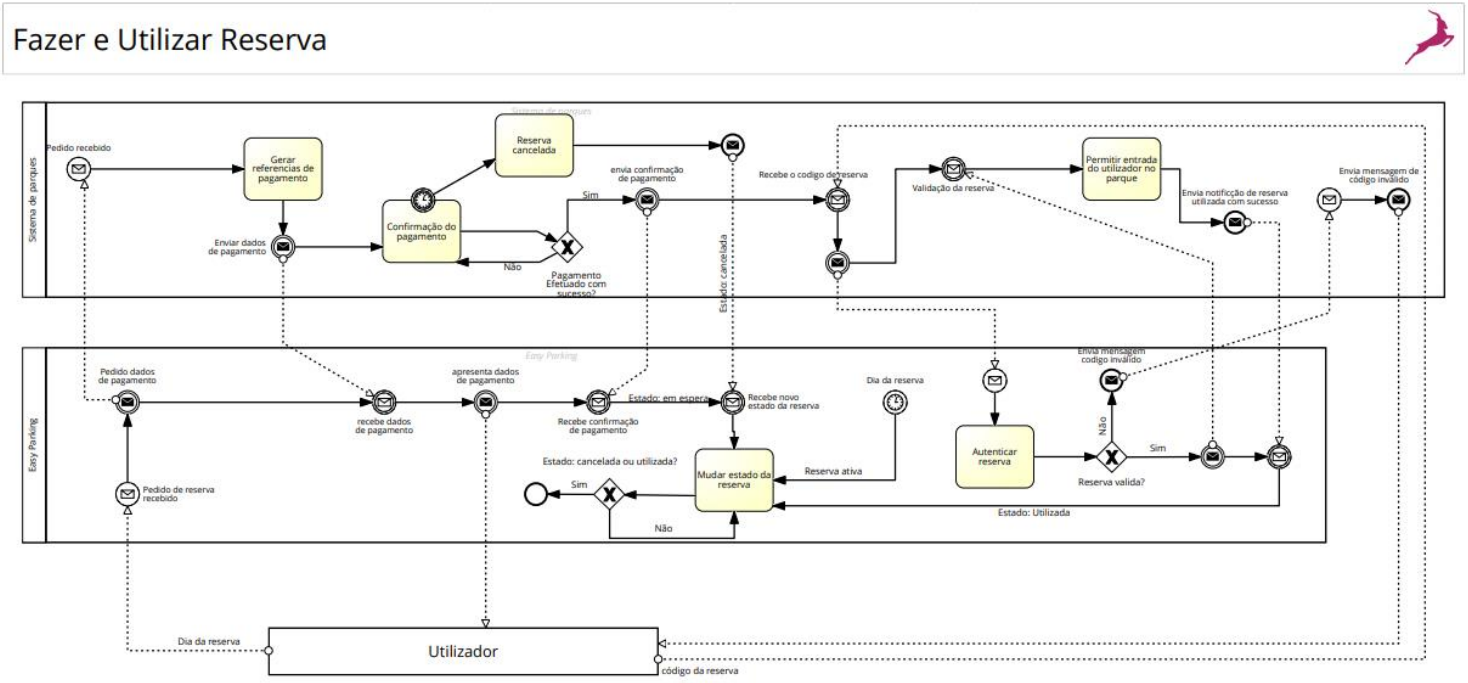




Descrição dos processos de interação (BPMN)

Ator seleciona na interface de informação do parque desejado o dia da reserva, de seguida clica na opção de fazer reserva, assim o SOI faz um pedido dos dados de pagamento ao Sistema de parques, o SOI apresenta-os ao User, a reserva fica guardada na base de dados no estado “Por pagar”, quando o sistema de parques indicar que o pagamento foi realizado o SOI muda o estado da reserva para “Em espera”, caso o timer do sistema de parques seja acionado, este envia notificação ao SOI e a reserva é cancelada, quando chegar o dia da reserva (selecionado pelo User) o SOI muda o seu estado para “Ativa”, o User assim pode dirigir-se ao parque e introduzir o código da reserva ao entrar, O sistema de parques indica o código que foi introduzido, SOI autentica o código e envia notificação de autenticação realizada com sucesso e assim o sistema de parques autoriza a entrada do utilizador no parque, de seguida o SOI muda o estado da reserva para “Utilizada”, caso contrário o SOI envia notificação de código invalido e o sistema de parques apresenta-a ao utilizador.

Processo 1: Fazer e Utilizar Reserva



Documentação do serviço REST

Documentação REST do projecto Easy Parking

Recurso parques (/api/parques)

Listar todos os parques Devolve a lista de todos os parques com a sua informação Permite obter todos os parques ou filtrados pelo nome/parte nome ou apenas de um parque quando dado um parqueID
/api/parques?Nome=:text&parqueID=:int **(get)**
Parâmetros: **nome** (opcional): Os parques mostrados têm de conter o texto neste parâmetro no nome do parque; **parqueID** (opcional): O parque mostrado tem de ter o id igual ao número inteiro texto neste parâmetro.
Sucesso (200): [{ "LugaresPrioritarios": 4, "Tipologia": "Superfície", "ParqueID": 3, "Latitude": 38.706007, "Longitude": -9.175066,Descricao": "Morada: Rua das Fontainhas, 13-45", "Nome": "Alcântara", "LugaresTotal": 202, "precoDiario": 10, "LugaresDisponiveis": 186 },

{ “LugaresPrioritarios": 7, "Tipologia": "Estrutura", "ParqueID": 9, "Latitude": 38.74784, "Longitude": -9.177575,"Descricao": "Morada: Rua João Chagas, nº 3 - Urb. A.Moinhos", "Nome": "Alto dos Moinhos", "LugaresTotal": 279, "precoDiario": 19.4, "LugaresDisponiveis": 269 },
…]

Erro:
500: Erro de Servidor

Exemplo:

```
        let Nome = document.getElementById("search").value;
          let parques = await $.ajax({
            url: "/api/parques?Nome=" + Nome,
            method: "get",
            dataType: "json"
        });
```

Mudar lugares disponíveis
Verifica se os lugares disponíveis do parque (com o id nos dados recebidos) somado com o valor (1 ou -1) nos dados recebidos é maior ou igual a 0 e menor ou igual aos lugares totais do parque, de seguida altera os lugares disponíveis para o resultado dessa soma.

/api/parques/updateLugares **(put)**

Dados:

```
{  "parqueID": 1, "valor" : 1 }
```

Sucesso (200):

```
{
    "fieldCount": 0,
    "affectedRows": 1,
    "insertId": 0,
    "serverStatus": 2,
    "warningCount": 0,
    "message": "(Rows matched: 1  Changed: 1  Warnings: 0",
    "protocol41": true,
    "changedRows": 1
}
```

Erros:
500: Erro de servidor

Exemplo:

```
      let parqueObj = {
        parqueID: sessionStorage.getItem("parqueID"),
        valor: valor
      }
      let result = await $.ajax({
        url: "/api/parques/updateLugares/",
        method: "put",
        dataType: "json",
        data: JSON.stringify(parqueObj),
```

```
        contentType: "application/json"
      });
```

Recurso Reservas (/api/reservas)

Obter todos as reservas de um utilizador
Devolve a lista de reservas do utilizador

/api/reservas/reservasUser?userID= int **(get)**

Parâmetros:
userID (obrigatório): id do utilizador a quem pertence as reservas

Sucesso (200):

```
[
{ "Estado": "Ativa", "Nome": "Alcântara", "Descricao": "Morada: Rua das Fontainhas,
13-45", "Codigo": "DI1YA0573", "DataHora": "25/01/2021 às 03:41", "DiaReserva":
"27/01/2021" },
{ "Estado": "Ativa", "Nome": "Alto dos Moinhos", "Descricao": "Morada: Rua João
Chagas, nº 3 - Urb. A.Moinhos", "Codigo": "0ZA73EFZT", "DataHora": "25/01/2021 às
04:49", "DiaReserva": "28/01/2021" },
... ]
```

Erros:
500: Erro do servidor

Exemplo:

```
 let reservas = await $.ajax({
      url: "/api/reservas/reservasUser?userID= " +
sessionStorage.getItem("userID"),
      method: "get",
      dataType: "json",
    });
```

Fazer uma nova reserva
Recebe o dia da reserva e os id 's do utilizador e do parque num objeto, chama a função getCodigo que gera um codigo random e verifica se já existe na base de dados

/api/reservas/newBooking **(post)**

Dados:

```
{
    “diaReserva”: “24/02/2020”,
    “parqueID”: 1,
    “userID”: 5  }
```

Sucesso (200):

```
{
    "CustomerId": 2, "FirstName": "Leonie", "LastName": "Köhler", "Company": null,
    "Address": "Theodor-Heuss-Straße 34", "City": "Stuttgart", "State": null,
    "Country": "Germany", "PostalCode": "70174", "Phone": "+49 0711 2842222",
    "Fax": null, "Email": "leonekohler@surfeu.de", "SupportRepId": 5
}
```

Erros:
500: Erro de servidor

Exemplo:

```
  let login = {
            firstname: document.getElementById("fname").value,
            lastname: document.getElementById("lname").value,
            email: document.getElementById("email").value,
        }
        let customer = await $.ajax({
            url: "/api/customers/login",
            method: "post",
            data: JSON.stringify(login),
            dataType: "json",
            contentType: "application/json"
        });
        sessionStorage.setItem("customerId",customer.CustomerId);
        window.location = "profile.html";
```

Verificar reservas não utilizadas ou em espera
Verifica reservas não utilizadas (com diaReserva igual ao dia anterior) ou em espera com diaReserva igual ao dia atual(hoje) e muda estados e os lugares disponíveis do parque a que pertence

/api/reservas//newState **(put)**

Sucesso (200):

```
{
   "fieldCount": 0, "affectedRows": 1, "insertId": 0, "serverStatus": 2,
"warningCount": 0, "message": "(Rows matched: 1  Changed: 1  Warnings: 0",
"protocol41": true, "changedRows": 1
}
```

Erros:
500: Erro de servidor

Exemplo:

```
let reservas = await $.ajax({
            url: "/api/reservas/newState",
            method: "put",
            dataType: "json"
        });
```

Verificar código da reserva para ser utilizada e muda o seu estado
Verifica o código nos dados, o estado da reserva(ativa) e o dia, de seguida faz update ao estado dessa reserva para utilizada e devolve true, caso os dados não forem válidos devolve false.
Exemplo não implementado pois é chamado pelo parque para validar entrada de um utilizador com reserva no parque

/api/reservas//newState **(put)**

Sucesso (200):
```
{
msg: "Codigo valido"
}
```

Erros:
500: Erro de servidor
404: Não existe uma reserva que corresponda a esses dados
```
{    "msg": "Código invalido ou utilizador encontra-se no parque errado"  }
```

Exemplo:

Recurso Review (/api/reviews)

Fazer uma nova review
Recebe os dados da review num objeto e os introduz na base de dados

/api/reviews/ **(put)**

Dados:
```
{
“parqueID”: 1, “userID”: 5, “classificacao”: 4, “comentario”: “Otimo parque”
}
```

Sucesso (200):
```
{
affectedRows: 1
changedRows: 0
fieldCount: 0
insertId: 34
message: ""
protocol41: true
serverStatus: 2
warningCount: 0
}
```

Erros:
500: Erro de servidor

Exemplo:

```
  let review = {
            parqueID: 1,
            userID: 5,
            classificacao: 4,
            comentario: “Otimo parque
        }
         let result = await $.ajax({
            url: "/api/reviews",
            method: "post",
            dataType: "json",
            data: JSON.stringify(review),
            contentType: "application/json"
        });
         window.location = "infoParque.html";
```

Obter todos as reviews de um parque
Devolve a lista de reviews do parque, calcula a classificação média do parque e inroduz no objeto review

/api/reviews/reviewsParque/:parqueID **(get)**

Dados:
parqueID (obrigatório): id do parque identificado nas reviews

Sucesso (200):

```
{
"reviews":[{"Nome":"Ruben
Viana","Classificacao":4,"Comentario":"Barato"},{"Nome":"Ruben
Viana","Classificacao":4,"Comentario":"Otimo parque"}],
"classificacaoMedia":"4.0"
}
```

Erros:
500: Erro do servidor

Exemplo:

```
 let reservas = await $.ajax({
      url: "/api/reservas/reservasUser?userID= " +
sessionStorage.getItem("userID"),
      method: "get",
      dataType: "json",
    });
```

Recurso User (/api/users)

Fazer a autenticação de um utilizador
Recebe o username e password do utilizador e caso exista um utilizador com essa informação devolve o nome e id do utilizador.

/api/users/LoginDados/ **(get)**

Dados:
```
{
    "username": "RViana",
    "lastname": "Rv123",
 }
```

Sucesso (200):
```
[
{"userID":5,"nome":"Ruben Viana"}
]
```

Erros:
500: Erro de servidor

Exemplo:
```
let user = {
        username: "RViana",
        pass: "Rv123"
    }
         let dados = await $.ajax({
                url: "/api/users/LoginDados/",
                method: "get",
                dataType: "json",
                data: user,
                contentType: "application/json"

            });
       if (dados[0] != null) {
                await swal("Sessão Iniciada com sucesso!", "");
                sessionStorage.setItem("nome", dados[0].nome);
                window.location = "index.html";
            }
```

Fazer um novo utilizador
Recebe os dados do utilizador num objeto e os introduz na base de dados

/api/users/newUSer **(put)**

Dados:
```
{ "nome": "Ruben Viana", "username": "RViana", "password": "Rv123", }
```

Sucesso (200):
```
{ affectedRows: 1
changedRows: 0
fieldCount: 0
insertId: 34
message: ""
protocol41: true
serverStatus: 2
warningCount: 0 }
```

Erros:
500: Erro de servidor

Exemplo:
```
  let user = {
            username: newUsername,
            pass: password,
            nome: nome,}
         let result = await $.ajax({
                url: "/api/users/newUSer",
                method: "post",
                dataType: "json",
                data: JSON.stringify(user),
                contentType: "application/json"
            });
 if (result == null) {
                swal("Username ja se encontra utilizado, por favor
tente novamente com um username diferente!", "");}
            swal("User registado")
```