

Relatório do Projeto

1. Introdução

Este trabalho prático tem como objetivo principal o desenvolvimento de uma aplicação web destinada à gestão de restaurantes e respetivos pedidos. A plataforma contempla não só a vertente administrativa (back office), permitindo a gestão de funcionários, menus e pratos, como também a vertente de interação com os clientes finais.

A aplicação possibilita que diferentes restaurantes se candidatem para fazer parte da plataforma, sendo posteriormente validados por um administrador da aplicação. Após aprovação, os administradores dos restaurantes passam a ter acesso a um conjunto de funcionalidades que lhes permite gerir os seus dados, menus, colaboradores e pedidos.

Do lado do cliente, a plataforma permite o registo e autenticação, possibilitando a realização de encomendas de restaurantes registados. Os clientes podem escolher o local de levantamento da sua encomenda ou, em alternativa, proceder ao registo presencialmente através de um funcionário do restaurante.

Todo o desenvolvimento do projeto foi feito com auxílio a um repositório remoto no gitlab, para permitir a colaboração em equipa e o controlo de versões (<https://gitlab.estg.ipp.pt/grupo4/trabalhopaw>).

Contas para teste de backend		
email	password	descrição
admin@geral.com	password	Conta de administrador geral
rest@admin.com	12345	Conta de administrador de restaurante
elpresidente@hotmail.com	12345	Conta de funcionário

Contas para teste de frontend		
email	password	descrição
cliente@cliente.com	12345	Conta de cliente registado que foi bloqueado de fazer pedidos por demasiados cancelamentos
cheiodefome@gmail.com	cheiodefome	Conta de cliente

2. Descrição da Milestone 1

A primeira *milestone* centrou-se no desenvolvimento da estrutura base da aplicação, com especial destaque para o desenho do modelo de dados e a definição dos diferentes tipos de utilizadores do sistema. Foram implementadas as funcionalidades de registo e autenticação, bem como os mecanismos de verificação e autorização de acesso às diversas áreas da aplicação, de acordo com o perfil do utilizador.

Foi também desenvolvida a área de *back office*, permitindo que os utilizadores com o perfil de administrador de restaurante submetam pedidos de registo de restaurante, incluindo todas as informações relevantes e categorias associadas. Estes pedidos são posteriormente analisados por administradores da aplicação.

Para além disso, foram criadas funcionalidades de gestão de menus e pratos, possibilitando a associação de categorias a cada prato. Adicionalmente, foi implementada a gestão de funcionários, permitindo que o administrador de restaurante possa gerir a sua equipa diretamente através da plataforma.

3. Descrição da Milestone 2

Na segunda *milestone* focámo-nos na interação do cliente final através de uma aplicação de *front-end*, que interage com a aplicação de *backoffice* anteriormente desenvolvida, através de endpoints de uma **REST API** criada no servidor.

A aplicação de *front-end* foi desenvolvida através da *framework* “**Angular**”.

As operações **CRUD** necessárias para o registo de um cliente novo, alteração de morada, alteração de dados no perfil do cliente, pesquisa de restaurantes, visualizações de restaurantes, e respetivos menus e refeições, cálculo de distâncias, localizações, visualização de comentários de restaurantes, envio de imagens nos comentários, filtragem por categorias de restaurantes, consulta de histórico de pedidos, visualização do pedido corrente, criação de um pedido novo e uma apreciação global sobre os conteúdos nutritivos do carrinho do cliente através de um LLM (Large Language Model), foram todos feitos através destes *endpoints* criados na REST API ou através de APIs externas.

Toda a documentação relativa à REST API foi gerada automaticamente através da ferramenta “**Swagger**”, que pode ser visualizada através da rota acessível através do url <http://localhost:3000/api-docs/>.

Para além da implementação destes novos requisitos procedemos também à eliminação de alguns bugs que permaneceram da primeira *milestone*.

Por fim tratou-se da sincronização em tempo real dos pedidos da parte de *front-end* com o *backend*, com recurso a SSE (Server Sent Events), sem ter a necessidade de utilizar WebSockets. Apesar de WebSockets ser o ideal para aplicações que exigem correpondência em tempo real, no nosso caso determinamos que SSE num endpoint da REST API era o suficiente para a integração da segunda *milestone*.

4. Organização do trabalho

O ponto de partida do projeto foi a definição e estruturação do modelo de dados, conforme os requisitos estabelecidos para a primeira milestone, detalhado na secção 5 deste relatório. Com base neste modelo inicial, desenvolveu-se a lógica principal da aplicação, ajustando progressivamente o modelo de dados conforme os desafios que surgiam durante o processo.

Para facilitar o desenvolvimento e organização do projeto, o grupo definiu claramente as vistas (views) associadas a cada tipo de utilizador. Estas vistas abrangem desde o registo e login até às funcionalidades específicas, consoante as permissões atribuídas a cada papel (role) de utilizador. Com este objetivo, foi criada uma vista geral que clarificava todas as ações possíveis e definia claramente as fronteiras e permissões dos diferentes utilizadores. Este passo permitiu ao grupo debater aspetos essenciais do projeto, como a necessidade de aprovação administrativa das categorias (quer de pratos, quer de restaurantes), bem como validar o fluxo geral da aplicação.

Após esta fase colaborativa inicial, as diversas vistas foram distribuídas pelos membros do grupo para implementação individualizada. Durante este processo, surgiram alguns desafios relacionados com a sobreposição de funcionalidades, já que diferentes elementos se encontravam a trabalhar simultaneamente em partes da lógica semelhantes. Estes conflitos foram ultrapassados através da realização de reuniões regulares de acompanhamento, que permitiram a integração frequente dos desenvolvimentos individuais numa versão atualizada e coerente do projeto.

Outro desafio enfrentado pelo grupo foi a distribuição equitativa das “vistas” entre os membros. Este problema foi mitigado graças ao forte espírito colaborativo da equipa, onde se destacou a entreaajuda constante entre todos os elementos para garantir o sucesso e completude do produto final.

Exemplo de alguns dos protótipos inicialmente feitos pelo grupo:

View Adicionar/Remover categorias de pratos (Admin Geral)

Pedidos de categorias de pratos pendentes:

pedido1: porco chinês	aprovar	rejeitar
pedido2: porco congelado (sobremesa)	aprovar	rejeitar

Adicionar nova categoria:

Nome: ...

Remover categoria:

Dropdown de categorias	categoria1	apagar
------------------------	------------	--------

depois de selecionada, da para carregar no botao de apagar

view Novo Registo

name

mail

password

tipo de utilizador - é um dropdown menu que se for
"Client" mostra os seguintes campos

contacto, nif, etc

address

delivery address

meter um visto que está on por default: "delivery
address is the same as the address"

5. Modelo de dados e utilizadores

Para iniciarmos o modelo de dados precisamos identificar os utilizadores da aplicação, que são o cliente, o administrador de restaurante, o administrador da aplicação, e o funcionário de um restaurante.

Depois foram verificados quais eram as informações necessárias de cada utilizador para formar os schemas que a base de dados tem de respeitar.

Os utilizadores foram divididos em dois schemas o do Staff (administrador, administrador de restaurante e funcionário de restaurante) e o cliente.

No modelo de dados do staff diferencia-se os vários tipos de utilizadores através de 'role'. A cada um desses utilizadores é associado um restaurante na qual pode obter várias permissões dependendo do seu cargo.

Representação do modelo de dados de Staff			
Campo	Tipo	Obrigatório	Observações
email	String	Sim	Tem que ser único e é predefinido para que tenha formatação de um email
password	String	Sim	A password é encriptada antes de guardada
name	String	Sim	
role	String(enum)	Sim	Pode ser Admin, RestAdmin ou RestWorker
restID	objectId	Não	Tem o id do restaurante associado ao utilizador
createdAt	Date	Não	Tem valor default da data de criação
updatedAt	Date	Não	Tem valor default da data em que haja uma modificação nos camps

O cliente que se regista pela aplicação terá de preencher todos os campos abaixo.

Representação do modelo de dados do cliente			
Campo	Tipo	Obrigatório	Observações
email	String	Sim	Tem de ser único e é predefinido para que tenha formatação de um email
password	String	Sim	A password é encriptada antes de guardada
name	String	Sim	
location	locationSchema	Sim	Tem embutido a informação de um schema de localização. Esta localização(morada) é a morada de faturação
deliveryLocation	locationSchema	Sim	Tem embutido a informação de um schema de localização. Esta localização(morada) é a morada de entrega da encomenda
nif	String	Sim	O NIF tem uma validação para ter apenas 9 dígitos
companyName	String	Não	Só é necessário se o NIF começar por 5,6,8 ou 9 pois identifica que é um NIF empresarial
contact	String	Sim	O número deve começar por 2 ou 9 e ter 9 dígitos
role	String	Sim	Tem valor por defeito de 'Client'
createdAt	Date	Não	Tem valor por defeito da data de criação
updatedAt	Date	Não	Tem valor por defeito da data em que haja uma modificação nos campos

O location schema usado em dois campos dos dados do cliente e também para a morada o restaurante é da forma:

Representação do modelo de dados da morada			
Campo	Tipo	Obrigatório	Observações
address.street	String	Sim	
address.postalCode	String	Sim	O código postal tem de ter o formato XXXX-XXX
address.city	String	Sim	
address.country	String	Sim	Tem valor por defeito de 'Portugal'
coordinates.lat	Number	Sim	Latitude da morada inserida
coordinates.lon	Number	Sim	Longitude da morada inserida

Os campos a inserir pelos utilizadores é apenas 'address'. De seguida, caso a morada esteja correta, uma API externa irá gerar as coordenadas do lugar indicado.

Para os restaurantes é usado o modelo de dados:

Representação do modelo de dados de Restaurant			
Campo	Tipo	Obrigatório	Observações
name	String	Sim	
location	locationSchema	Sim	Tem embutido a informação de um schema de localização. Esta localização(morada) é a morada do restaurante
maxOrders	Number	Sim	Tem valor por defeito de 50 encomendas ativas, mas pode ser modificado pelo administrador do restaurante
approvedByAdmin	Boolean	Não	O valor por defeito é 'false' mas quando o administrador da aplicação aceita o pedido do restaurante passa a 'true'
maxDeliveryRange	Number	Sim	Qual o raio máximo de entregas que o restaurante aceita
menu	Array de MenuSchema	Não	Tem embutido todos os menus criados pelo administrador de restaurante
meals	Array de MealSchema	Não	Tem embutido todos os pratos criados pelo administrador de restaurante
categories	Array de objectId	Não	Tem a referência das categorias do restaurante que podem ser no máximo 3
nif	String	Sim	O NIF deve ter 9 dígitos e começar com o dígito 5 (entidades como restaurantes)
companyName	String	Sim	Este nome é único e representa o nome da empresa que pode ser diferente do nome do restaurante
createdAt	Date	Não	Tem valor por defeito da data de criação

O 'name' e 'companyName' geralmente são diferentes então são pedidos ambos. A empresa ao qual o restaurante pertence geralmente nunca tem o mesmo nome que o restaurante.

As categorias do restaurante têm o seu próprio schema e coleção, sendo que nos restaurantes apenas se fica guardado o id do objeto na sua coleção. Já os pratos ('meals') e os menus são embutidos na coleção do restaurante.

Para os menus é usado o modelo de dados:

Representação do modelo de dados de menu			
Campo	Tipo	Obrigatório	Observações
name	String	Sim	
meals	objectId	Não	Tem a referência dos vários pratos que compõem o menu, podendo ter no máximo 10 pratos
isActive	Boolean	Não	Tem valor por defeito 'true' e o administrador de restaurante pode desativar o menu para que não seja visível para o cliente

No menu apenas ficam guardados os id's dos vários pratos ('meals') que compõem o menu.

Para os pratos ('meals') é usado o modelo de dados:

Representação do modelo de dados de prato			
Campo	Tipo	Obrigatório	Observações
name	String	Sim	
description	String	Sim	Descrição do prato
sizes.name	String	Sim	Nome da porção (ex.: ½ dose)
sizes.price	Number	Sim	Preço da porção sizes é um array de porções (que tem nome e preço) que tem de ter no mínimo 1 porção
category	Array de objectId	Não	Tem a referência das categorias do prato
images	Array de MealImageSchema	Não	Tem embutido as várias as informações do schema referido

Ter em atenção que as categorias do prato são diferentes das categorias de um restaurante. No prato as referências de categorias apenas dizem respeito ao prato (ex.: sobremesa).

Para as imagens de um prato é usado o modelo de dados:

Representação do modelo de dados de imagens de pratos			
Campo	Tipo	Obrigatório	Observações
imagePath	String	Sim	O path da imagem usada

Para as categorias de um prato:

Representação do modelo de dados da categoria de um prato			
Campo	Tipo	Obrigatório	Observações
name	String	Sim	O nome da categoria de prato é único
approved	Boolean	Não	Tem valor por defeito de 'false', e as categorias de prato tem que ser aprovadas pelo administrador da aplicação

Para as categorias de um restaurante é usado o modelo de dados:

Representação do modelo de dados da categoria de um restaurante			
Campo	Tipo	Obrigatório	Observações
name	String	Sim	O nome da categoria de prato é único
approved	Boolean	Não	Tem valor por defeito 'false', e as categorias de restaurante tem que ser aprovadas pelo administrador da aplicação

Quando um administrador de restaurante faz a requisição para que o seu restaurante esteja disponível na aplicação, é criado um restaurante (com modelo schema de um restaurante) e restaurante para aprovação.

Este pedido de aprovação tem o id do restaurante e a data de requisição. Após ser aprovado pelo administrador da aplicação o pedido é eliminado e o restaurante fica aprovado pelo administrador. Caso não seja aprovado o pedido e o restaurante são eliminados.

Para a aprovação de um restaurante:

Representação do modelo de dados de restaurante para aprovação			
Campo	Tipo	Obrigatório	Observações
restaurant	objectId	Sim	Referência do restaurante que tem de ser aprovado pelo administrador da aplicação. A referência tem de ser única.
requestAt	Date	Não	Tem valor por defeito da data do pedido para aprovação

Para um pedido feito por um cliente:

Representação do modelo de dados de um pedido			
Campo	Tipo	Obrigatório	Observações
costumer.name	String	Sim	Nome do cliente que fez o pedido
costumer.email	String	Sim	Email do cliente que fez o pedido
costumer.nif	String	Sim	NIF do cliente que fez o pedido
costumer.contact	String	Sim	Contacto do cliente que fez o pedido
costumer.companyName	String	não	Nome de empresa do cliente que fez o pedido
costumer.role	String	Sim	O valor por defeito é 'client'
costumer.location	locationSchema	Sim	Morada do cliente que fez o pedido
costumer.deliveryLocation	locationSchema	Sim	Morada de entrega do cliente que fez o pedido
restaurante.name	String	Sim	Nome do restaurante que foi feito o pedido
restaurante.location	locationSchema	Sim	Morada do restaurante que foi feito o pedido
restaurante.categories	Array de String	Sim	Categorias do restaurante que foi feito o pedido
restaurantId	objectId	Sim	Referência para qual restaurante foi feito o pedido
worker.name	String	Sim	Nome do funcionário do restaurante recebeu o pedido
worker.email	String	Sim	Email do funcionário do restaurante recebeu o pedido
meals	Array de MealSchema	Não	Tem embutido os diferentes pratos que foram pedidos
canceledMeal	Boolean	Sim	Por valor por defeito é 'false'
currentStatus	String(enum)	Sim	Pode adquirir os valores 'Ordered', 'Preparing', 'Ready', 'Delivered'. Tem o valor por defeito 'Ordered'
preparationTime	Number	Sim	O tempo de preparação do pedido tem em conta o número de pedidos ativos do restaurante
deliveredAt	Date	Não	Data em que foi entregue o pedido
createdAt	Date	Não	Tem valor por defeito a data que foi criado o pedido

Diagrama do modelo de dados inicial:

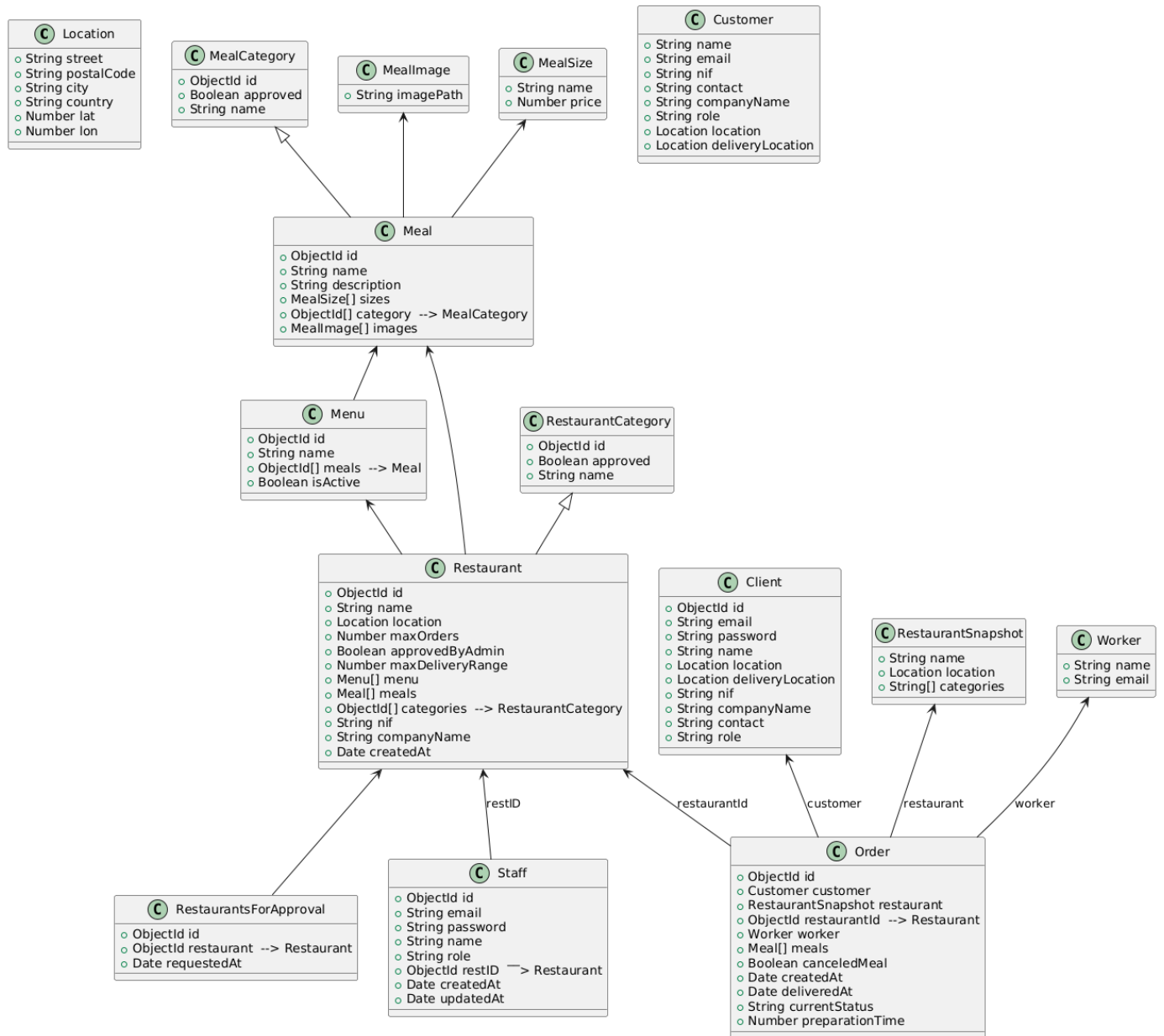


Figura 1: Diagrama de classes para representar o modelo de dados.

Modelo de dados complementar da 2ª Milestone

Na aplicação de front-end foi necessário acrescentar novos modelos de dados, para dar resposta aos requisitos enunciados. Fez-se também uma pequena alteração ao modelo de dados de restaurantes, para que cada restaurante tivesse uma imagem representativa do seu logotipo.

Foi introduzido o modelo de dados de comentários com o suporte para as suas respetivas imagens.

Representação do modelo de dados de um comentário [ImageSchema]			
Campo	Tipo	Obrigatório	Observações
imagePath	String	Sim	String que corresponde ao nome do ficheiro da imagem

Representação do modelo de dados de um comentário [CommentSchema]			
Campo	Tipo	Obrigatório	Observações
clientId	ObjectId	Sim	Identificador único do <u>cliente</u> , neste caso é o ObjectId do documento do respetivo cliente
restaurantId	ObjectId	Sim	Identificador único do <u>restaurante</u> , neste caso é o ObjectId do documento do um restaurante ao qual o comentário foi feito
text	String	Sim	O comentário do cliente
images	ImageSchema	Não	É um conjunto de imagens que o cliente escolhe para fazer upload juntamente com o seu comentário. As imagens são facultativas.
createdAt	Date	Sim	Quando o comentário é criado, a data atual é adicionada ao documento

Foi também introduzido um modelo de dados de blacklist, para poder anotar nesta coleção todos os clientes que cancelaram pelo menos 5 pedidos num restaurante num período inferior a 30 dias.

Representação do modelo de dados da Blacklist [BlacklistSchema]			
Campo	Tipo	Obrigatório	Observações
clientId	ObjectId	Sim	Identificador único do cliente
reason	String	Sim	Razão pelo qual este cliente entrou na blacklist
blackListedAt	Date	Sim	Data em que foi anotada a transgressão do cliente
expiresAt	Date	Sim	Data em que expira a transgressão

3. Tecnologias Utilizadas

Durante a primeira *milestone* foram utilizadas diversas tecnologias que suportam a implementação da estrutura base da aplicação, tanto a nível de servidor como de interface com o utilizador. O desenvolvimento do *backend* foi realizado com recurso ao Node.js, um ambiente de execução JavaScript no lado do servidor, que permite lidar com múltiplos pedidos de forma eficiente e não bloqueante. Sobre esta base, utilizou-se a *framework* Express.js, que facilitou a criação e organização de rotas, controladores e *middleware*, promovendo uma arquitetura modular e clara para a aplicação.

A gestão da base de dados foi feita com MongoDB, uma base de dados NoSQL orientada a documentos, que se revelou adequada para a flexibilidade exigida pela aplicação. Para modelar os dados e definir os esquemas das entidades, foi utilizada a biblioteca Mongoose, que permite criar modelos com validações e relações entre documentos, como é o caso das associações entre restaurantes e as respetivas categorias, ou entre um administrador de restaurante e o restaurante que gere.

A interface foi construída utilizando EJS (Embedded JavaScript), um motor de templates que permite gerar HTML dinâmico no lado do servidor, integrando variáveis e estruturas de controlo diretamente no ficheiro da vista. Esta abordagem simplifica a criação de páginas dinâmicas e reutilizáveis, que se adaptam ao utilizador e aos dados da base de dados. Para manipulação do conteúdo exibido nas páginas e melhorar a interação com o utilizador, foi explorado também o DOM (Document Object Model), que permite aceder e modificar a estrutura do documento HTML de forma dinâmica. Através do DOM, é possível alterar conteúdos, validar formulários e adaptar o comportamento da interface com base em eventos ou dados introduzidos.

Relativamente à segurança e autenticação, foi implementado um sistema baseado em JSON Web Tokens (JWT), que são guardados em cookies no navegador do utilizador. As passwords são encriptadas com o algoritmo bcrypt antes de serem armazenadas na base de dados, garantindo a confidencialidade dos dados sensíveis.

Adicionalmente, a aplicação recorreu a APIs externas para melhorar a fiabilidade dos dados introduzidos. No caso da morada, foi utilizada uma API de geocodificação que converte endereços textuais em coordenadas geográficas (latitude e longitude), essenciais para funcionalidades como o cálculo de distâncias ou a definição de um raio máximo de entrega. Para garantir a validade dos números de telemóvel inseridos pelos utilizadores, recorreu-se à API disponibilizada pela plataforma apilayer, que permite validar se um número é real, ativo e pertence ao país correto. Dado que a aplicação está inicialmente direcionada para o mercado português, foi adicionada automaticamente a indicação nacional (+351) antes do envio do número para validação.

Na segunda milestone, como já foi referido anteriormente, a framework utilizada foi Angular. A lógica de implementação deste projeto de front-end, com recurso a esta framework, assentou numa arquitetura de serviços e componentes. A linguagem de programação desta *framework* é TypeScript.

Os componentes do projeto em Angular geriram e apresentaram a lógica e a interação do utilizador com a aplicação, enquanto que os serviços trataram de injeção de dependências através de pedidos à API REST, tratamento de dados, e da comunicação entre componentes. Os componentes não comunicam entre si, apenas entre serviços.

Nos serviços, através da biblioteca RxJS que está integrada com a Angular, utilizamos o padrão de software de Observer/Listener. Este padrão facilitou a comunicação entre componentes. Os componentes que estão subscritos aos observables implementados nos serviços, são notificados automaticamente sempre que os dados mudam.

Outras duas ferramentas do Angular que foram importantes na implementação do projeto de front-end, foram as Guards e os Interceptors. As *guards* serviram para proteger rotas, garantindo que apenas utilizadores autenticados acedem a determinadas páginas da aplicação, através do token que o utilizador recebe ao conseguir fazer o login com sucesso. Os *Interceptors* serviram para garantir que todos os pedidos a endpoints protegidos, são autenticados sem que seja repetido o código em cada serviço.

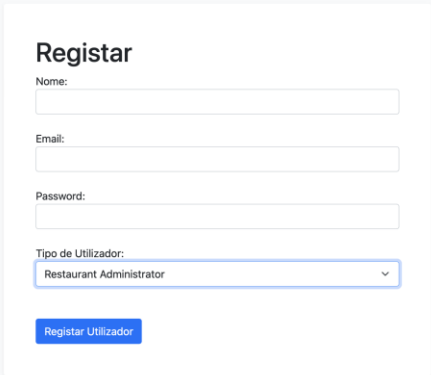
No projeto de front-end foram também utilizadas as tecnologias de HttpClientModule para comunicação com o backend através da API REST. Foi utilizado Server Sent Events (SSE) para estabelecer uma comunicação em tempo real do servidor com a aplicação de front-end. Para o teste dos endpoints da API REST desenvolvida utilizamos o Postman.

O local storage foi utilizado para a persistência do carrinho. Para além disso foi utilizado o bootstrap para conseguirmos ter um layout mais moderno e apelativo, através de navbar, cards e botões.

5. Funcionalidades Implementadas na 1ª Milestone

5.1. Administrador de Restaurante

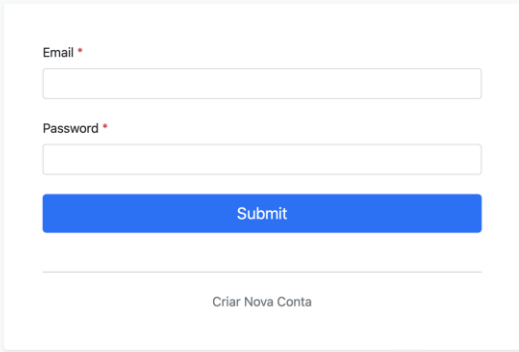
Qualquer administrador de restaurante tem, em primeiro lugar, de se registar na plataforma para poder submeter o pedido de registo do seu restaurante. Para tal, deve preencher um formulário com os dados necessários, conforme ilustrado na Figura 5. Entre os dados obrigatórios está a seleção do tipo de utilizador, sendo as opções "Cliente" ou "Administrador de Restaurante".



O formulário de registo para um administrador de restaurante contém os seguintes campos:

- Nome:** Campo de texto para o nome do utilizador.
- Email:** Campo de texto para o endereço de email.
- Password:** Campo de texto para a palavra-passe.
- Tipo de Utilizador:** Menu suspenso com a opção "Restaurant Administrator" selecionada.
- Registrar Utilizador:** Botão azul para submeter o registo.

Figura 3: Página de registo para um administrador de restaurante



A página de login contém os seguintes elementos:

- Email:** Campo de texto para o endereço de email, com um asterisco vermelho indicando obrigatoriedade.
- Password:** Campo de texto para a palavra-passe, com um asterisco vermelho indicando obrigatoriedade.
- Submit:** Botão azul para submeter as credenciais.
- Criar Nova Conta:** Link de texto para a página de registo.

Figura 2: Página de Login

Após o registo, a palavra-passe do utilizador é encriptada com recurso ao algoritmo bcrypt e armazenada na base de dados. No momento do login, Figura 4, a palavra-passe fornecida é comparada, através do método compare do bcrypt, com a palavra-passe associada ao email inserido. Caso os dados estejam corretos, é gerado um token contendo o ID do utilizador e o seu "role". Este token é guardado num cookie com validade de 24 horas, permitindo a validação e proteção das rotas adequadas. O administrador de restaurante é então redirecionado para a página inicial, Figura 7, destinada a utilizadores que ainda não submeteram um pedido de registo de restaurante.

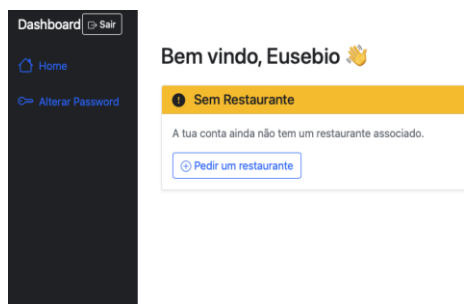


Figura 5: Página de Administrador de restaurante sem restaurante.

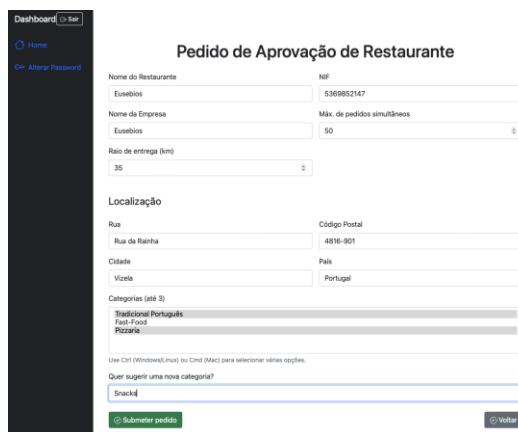


Figura 4: Pedido para registar um restaurante.

Nessa página, o administrador pode submeter o pedido de registo do restaurante, Figura 6, preenchendo todos os campos obrigatórios. No backend, é verificado se já existe um restaurante com o mesmo NIF. Caso exista, é apresentada uma mensagem de erro: "Já existe um restaurante com esse NIF". A morada fornecida é validada através de uma API externa, sendo convertida em coordenadas geográficas. Verifica-se também se o número de categorias escolhidas não ultrapassa o limite permitido.

Se tudo estiver correto, é criado o restaurante, bem como o respetivo pedido de aprovação e as novas categorias (caso tenham sido sugeridas). Estas são associadas à informação do administrador do restaurante.

Após a aprovação do restaurante por parte do administrador da aplicação, o administrador do restaurante passa a ter acesso a um dashboard mais completo, Figura 8, com funcionalidades de gestão dos pratos, menus, funcionários e encomendas do restaurante.

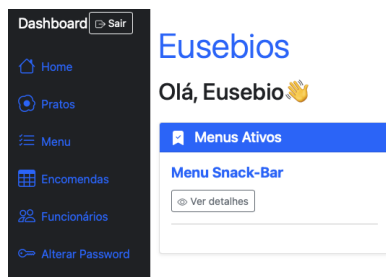


Figura 6: Página inicial de um administrador de restaurante quando o restaurante é aprovado.

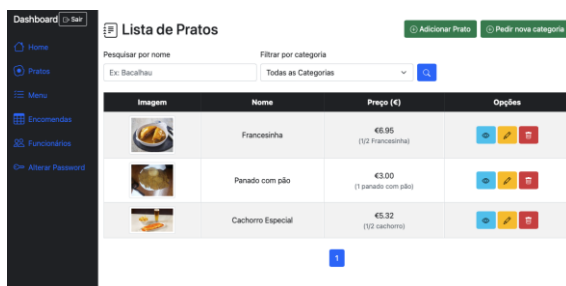


Figura 7: Página com a listagem de pratos.

Na barra lateral esquerda, está disponível a opção "Pratos", Figura 9, onde o administrador pode adicionar novos pratos, submeter pedidos de novas categorias (que requerem aprovação) e visualizar os pratos já criados.

Para adicionar um prato, é necessário preencher todos os campos do formulário, Figura 11, incluindo as doses e os respetivos preços. O administrador tem flexibilidade para definir os nomes das doses, adaptando-se à tipologia do prato (exemplo: "individual" e "familiar" para pizzas, ou "meia dose" e "dose completa" para pratos tradicionais). O pedido de nova categoria exige apenas o nome e fica pendente de aprovação.

Figura 9: Formulário para adicionar novo prato.

Figura 8: Lista de menus.

Dashboard Sair

Home

Pratos

Menu

Encomendas

Funcionários

Alterar Password

Lista de Menus

Criar Novo Menu

Nome do Menu	Data de Criação	Nº de Pratos	Opções
Menu Snack-Bar	23/04/2025	2	<div><div></div><div></div><div></div><div></div></div>
Menu Estudante	23/04/2025	3	<div><div></div><div></div><div></div><div></div></div>

Dashboard Sair

Home

Pratos

Menu

Encomendas

Funcionários

Alterar Password

Adicionar Novo Prato

Nome do prato

Francesinha

Descrição

Francesinha à moda do Porto, com molho azevado.

Doses e Preços

1/2 Francesinha

6,95

1 Francesinha

9,65

+ Adicionar dose

Categoria

Entradas

Prato Principal

Sobremesa

Sopa

Segura Ctrl (ou Cmd) para seleccionar várias.

Imagem

Escolher ficheiros

2 ficheiros

Adicionar

Cancelar

Na mesma página são listados todos os pratos criados, com opções para visualizar, editar ou remover. Caso existam muitos pratos, a lista é paginada com 5 itens por página, e inclui uma barra de pesquisa para facilitar a localização de pratos na página atual.

A secção "Menus" apresenta uma lista com nome, data de criação, número de pratos, e opções para ver detalhes, editar ou ativar/desativar o menu, Figura 10. Esta última funcionalidade será relevante para o lado do cliente, visto que apenas menus ativos são exibidos ao público.

Ao visualizar um menu em detalhe, são apresentados os pratos incluídos e a possibilidade de mudar o seu estado. Durante a edição, é possível alterar o nome, os pratos associados e o estado (ativo/inativo).

Figura 11: Lista de encomendas.

Dashboard

Home

Pratos

Menu

Encomendas

Funcionários

Alterar Password

Lista de Encomendas

Tempo médio por refeição: 3

Atualizar

Filtrar por MF

Filtrar por Email de Cliente

Mostrar por data

Só dígitos

email@exemplo.com

Limpar Filtros

Número	Cliente	Comanda	Hora do pedido	Hora de entrega	Tempo de preparação	Estado Atual da Pedido	Opções
11	Tiago	Francesinha	20:35	Não entregue	3 min	Cancelado	Cancelar Dados de cliente
10	Sem Nome	Cachorro Especial, Cachorro Especial	19:52	Não entregue	18 min	Cancelado	Cancelar Dados de cliente
9	Tiago	Francesinha	19:52	Não entregue	6 min	Cancelado	Cancelar Dados de cliente
8	Júlio	Francesinha	19:51	19:51	3 min	Entregue	Concluído - Cancelar Dados de cliente
7	Maria	Cachorro Especial	14:55	15:10	3 min	Entregue	Concluído - Cancelar Dados de cliente
6	Maria	Francesinha	14:53	Não entregue	3 min	Cancelado	Cancelar Dados de cliente
5	Júlio	Francesinha	14:35	14:45	8 min	Entregue	Concluído - Cancelar Dados de cliente

Figura 10:Lista de funcionários.

Dashboard

Sair

Home

Pratos

Menu

Encomendas

Funcionários

Alterar Password

Lista de Funcionários

Pesquisar funcionário...

Adicionar Funcionário

Nome	Email	Função	Data de Registo	Opções
func2Eusebios	func2eusebios@gmail.com	RestWorker	23/04/2025	<div></div> <div></div>
func1eusebios	func1eusebios@gmail.com	RestWorker	23/04/2025	<div></div> <div></div>

Na secção "Encomendas", estão listadas todas as encomendas feitas ao restaurante, ordenadas da mais recente para a mais antiga, Figura 13. Para facilitar a identificação, a numeração das encomendas reinicia-se diariamente. Cada encomenda apresenta: número, nome do cliente, pratos incluídos, hora do pedido, hora de entrega (caso já tenha sido feita), e tempo de preparação, calculado com base no tempo médio de confeção de cada prato e na quantidade de pedidos em estado "order" ou "preparing" que ainda não foram entregues nem cancelados.

Uma encomenda só pode ser cancelada nos primeiros cinco minutos após a sua criação e apenas se o seu estado ainda não tiver mudado para "preparing". O administrador ou funcionário do restaurante podem alterar o estado da encomenda para "em preparação", "pronto" ou "entregue" consoante o progresso.

O sistema permite filtrar encomendas por data, NIF do cliente ou email. Para criar uma nova encomenda localmente, o administrador pode seleccionar entre três tipos de cliente: um cliente já registado, um novo cliente (com registo completo), ou um cliente não registado (com nome, NIF e contacto opcionais). Após escolher o cliente, escolhe-se o menu e os pratos com as quantidades desejadas.

Na secção "Funcionários", Figura 12, é possível gerir os colaboradores do restaurante. São listados com nome, email, função e data de registo. Podem ser eliminados definitivamente da base de dados, pesquisados e novos funcionários podem ser adicionados através de um formulário com nome, email e palavra-passe.

Por fim, existe a opção para o administrador de restaurante atualizar a sua palavra-passe, sendo necessário introduzir a atual para verificação e confirmar a nova palavra-passe a definir.

5.2.Administrador da aplicação

O administrador da aplicação é uma conta pré-criada no sistema, não sendo possível o seu registo através da interface pública da aplicação. Este utilizador possui permissões elevadas e tem como responsabilidades a gestão e moderação de diversos elementos da plataforma, nomeadamente a aprovação de restaurantes, categorias de restaurantes e categorias de pratos, bem como a possibilidade de eliminar entidades que não estejam a ser utilizadas.

Na página principal (dashboard) do administrador geral, Figura 12, são apresentadas as opções de gestão mais relevantes, bem como um sistema de notificações que alerta para a existência de novos pedidos pendentes de aprovação. A navegação é realizada através de uma barra lateral esquerda, organizada por secções.

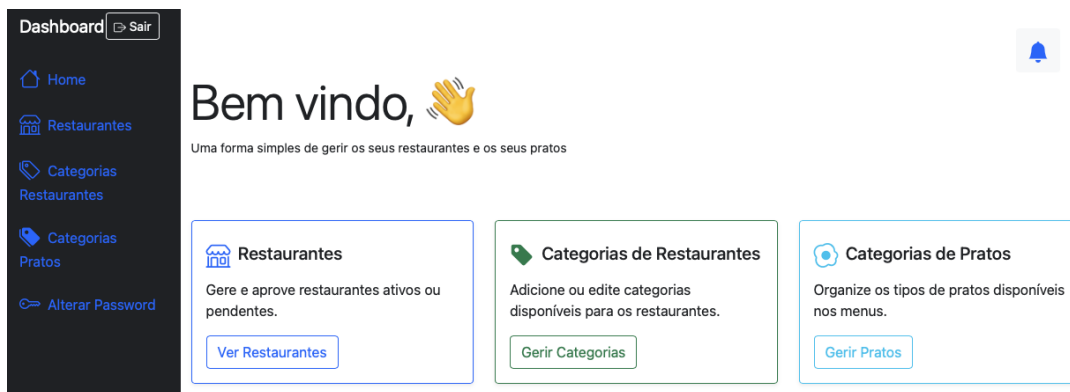


Figura 12: Dashboard do administrador

A secção "Restaurantes" permite visualizar todos os restaurantes existentes na plataforma, incluindo os que possuem pedidos de registo pendentes, Figura 13. A lista de restaurantes é paginada, apresentando até quatro restaurantes por página. É possível aplicar filtros por categoria e realizar pesquisas por cidade, com base na localização registada de cada restaurante.



Figura 13: Lista de restaurantes.

Quando um pedido de restaurante está pendente, o administrador pode:

- Aprovar o restaurante, atualizando o campo `approvedByAdmin` para `true` na base de dados.
- Rejeitar o pedido, o que resulta na remoção do restaurante e no envio de um email automático ao respetivo administrador de restaurante com a justificação da rejeição. Este envio é feito através de um transporter configurado com o serviço SMTP do Gmail.
- Eliminar um restaurante já aprovado, que não resulta numa eliminação total dos dados: os dados do restaurante são movidos para uma coleção alternativa chamada `removed_restaurants`, de forma a garantir a persistência da informação para eventual auditoria.

Na secção "Categorias de Restaurantes", Figura 16 o administrador pode:

- Adicionar novas categorias manualmente.

- Ativar categorias anteriormente desativadas.
- Desativar ou eliminar categorias, desde que estas não estejam a ser utilizadas por nenhum restaurante. Esta validação é feita através de uma consulta à coleção de restaurantes, verificando se a categoria está referenciada no array categories de algum documento.

Gerir Categorias de Restaurantes

Nova categoria

Nova Categoria: Adicionar

Categoria	Status	Ações
Italiano	Desativado	<button>Ativar</button> <button>Remover</button>
Japonês	Desativado	<button>Ativar</button> <button>Remover</button>
Chinês	Desativado	<button>Ativar</button> <button>Remover</button>
Tradicional Português	Ativado	<button>Desativar</button> <button>Remover</button>
Fast-Food	Ativado	<button>Desativar</button> <button>Remover</button>
Pizzaria	Ativado	<button>Desativar</button> <button>Remover</button>
Snacks	Ativado	<button>Desativar</button> <button>Remover</button>

Voltar ao Dashboard

Figura 14: Lista de categorias de restaurantes.

De forma análoga, a secção "Categorias de Pratos", Figura 15, permite ao administrador:

- Aprovar categorias de prato sugeridas por administradores de restaurante no momento da criação de pratos ou registo de restaurante.
- Desativar ou eliminar categorias que não estejam a ser referenciadas em nenhum prato de nenhum restaurante, através da verificação do array category presente nas meals embutidas nos documentos da coleção restaurants.
- Criar novas categorias para disponibilizar aos restaurantes.

Dashboard

Sair

Home

Restaurantes

Categorias Restaurantes

Categorias Pratos

Alterar Password

Gerir Categorias de Pratos

Nova categoria

Nova Categoria: Ex: Massas

Adicionar

Categoria	Status	Ações
Entradas	Ativado	Desativar Remove
Prato Principal	Ativado	Desativar Remove
Sobremesa	Ativado	Desativar Remove
Sopa	Desativado	Ativar Remove
Massas	Desativado	Ativar Remove
Snack	Ativado	Desativar Remove
Aperitivos	Ativado	Desativar Remove
Francesinha	Desativado	Ativar Remove
Brunch	Ativado	Desativar Remove

Voltar ao Dashboard

Figura 15:Lista de categorias de pratos.

Por fim, o administrador da aplicação pode atualizar a sua palavra-passe através da secção "Alterar Palavra-passe". Para tal, deve introduzir a palavra-passe atual e a nova palavra-passe. A verificação da palavra-passe atual é feita utilizando o método `bcrypt.compare`. Caso a verificação seja bem-sucedida, a nova palavra-passe é encriptada com `bcrypt.hash` e atualizada na base de dados.

Este conjunto de funcionalidades permite ao administrador da aplicação manter o controlo e garantir a qualidade da informação existente na plataforma, promovendo uma gestão segura e eficaz dos dados associados a restaurantes e categorias.

5.3.Funcionário do Restaurante

O funcionário do restaurante acede à aplicação através de credenciais previamente atribuídas pelo administrador do restaurante. Após autenticação bem-sucedida, é redirecionado para uma área reservada, cuja interface apresenta duas opções principais na barra lateral esquerda: "Encomendas" e "Alterar Palavra-passe".

A secção de encomendas permite ao funcionário visualizar e gerir todas as encomendas associadas ao restaurante ao qual está vinculado. As permissões atribuídas nesta secção são equivalentes às do administrador do restaurante. As encomendas são apresentadas por ordem decrescente de data de criação, e a numeração reinicia-se diariamente para facilitar a identificação dos pedidos, Figura 13.

Cada registo de encomenda inclui: número do pedido, nome do cliente, lista de pratos incluídos, hora de entrada do pedido, hora de entrega (caso aplicável) e o tempo estimado de preparação. Este último é calculado dinamicamente com base no tempo médio de confeção de cada prato e na quantidade de pedidos em estado "Ordered" ou "Preparing" que ainda não foram entregues nem cancelados. Este cálculo visa fornecer uma previsão mais realista do tempo de espera de cada encomenda.

É implementada uma restrição de cancelamento de encomendas: estas apenas podem ser canceladas nos primeiros cinco minutos após a sua criação e desde que ainda se encontrem no estado "Ordered". Tão cedo quanto o estado transite para "Preparing", o cancelamento deixa de ser possível. O funcionário (ou administrador) pode alterar o estado de uma encomenda para "Preparing", "Ready" ou "Delivered", consoante o progresso do seu processamento.

Para facilitar a navegação e organização da informação, estão disponíveis filtros por data, NIF do cliente e email. Adicionalmente, o sistema permite criar novas encomendas localmente. Neste contexto, o utilizador pode indicar o tipo de cliente, optando entre: um cliente já registado na plataforma, um novo cliente (com preenchimento de todos os dados obrigatórios), ou um cliente não registado (com campos opcionais como nome, NIF e contacto). Após selecionar o cliente, escolhem-se os menus e os respetivos pratos, bem como as quantidades desejadas.

Na secção "Alterar Palavra-passe", o funcionário pode proceder à atualização das suas credenciais de acesso. O processo exige a introdução da palavra-passe atual e da nova palavra-passe. A validação da palavra-passe atual é feita através do método `bcrypt.compare`, e, caso seja bem-sucedida, a nova palavra-passe é encriptada com `bcrypt.hash` antes de ser armazenada na base de dados. Este procedimento garante a segurança e a integridade dos dados de autenticação dos funcionários.

5.4.Cliente

Tendo em consideração que na primeira fase do projeto se encontra centrada exclusivamente no desenvolvimento do backoffice da aplicação, foram apenas implementadas as funcionalidades de registo e login para o tipo de utilizador "Cliente". O processo de registo encontra-se acessível através de um formulário onde o cliente deve preencher um conjunto de campos obrigatórios, tais como: nome, email (único e validado com regex), NIF (com validação de nove dígitos), contacto (com validação de prefixo e comprimento), morada de faturação e morada de entrega. As moradas seguem o modelo `locationSchema` e são validadas com apoio de uma API externa que permite obter as coordenadas geográficas associadas ao endereço.

Registar

Nome:

Email:

Password:

Tipo de Utilizador:

Client

Contacto:

NIF:

Morada de Faturação

Rua:

Cidade:

Código Postal:

Pais:

Portugal

☐ Morada de Entrega é igual à Morada de Faturação

Morada de Entrega

Rua:

Cidade:

Código Postal:

Pais:

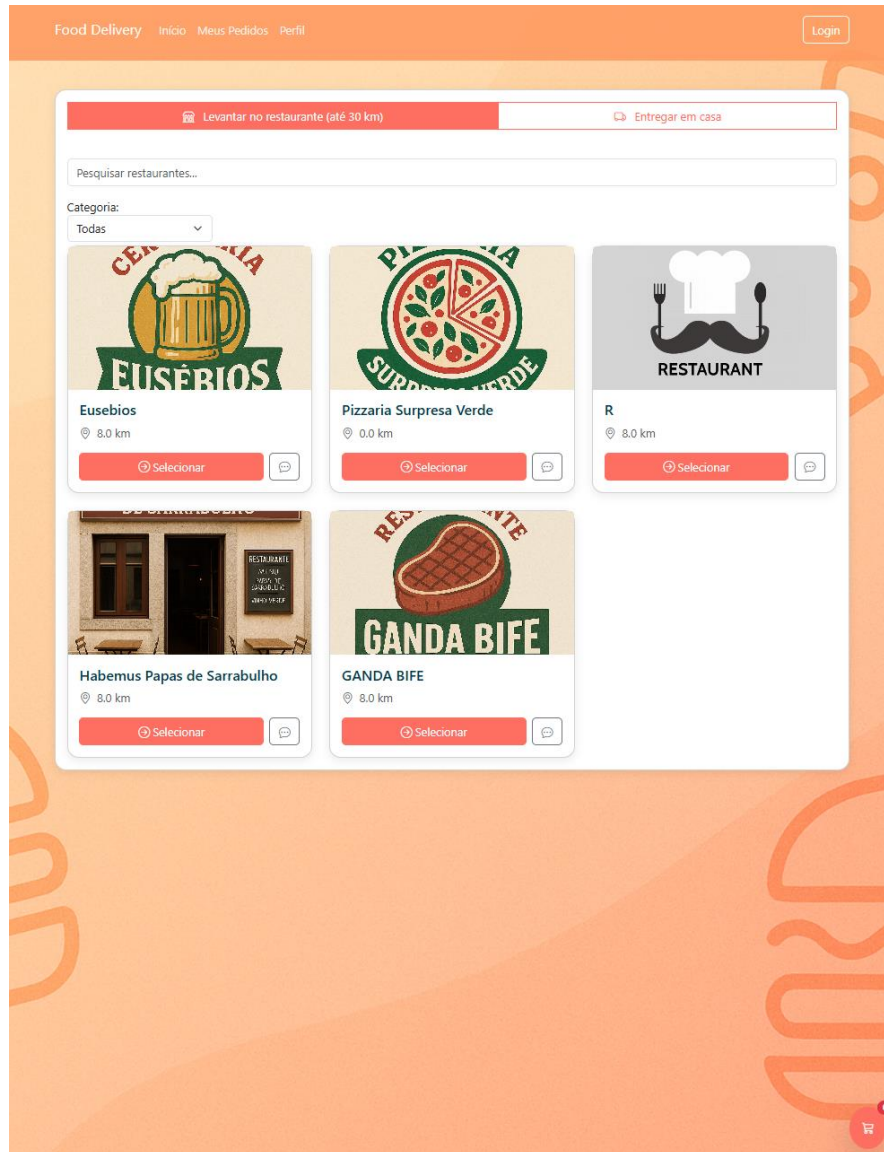
Registar Utilizador

Figura 16: Formulário de registo de clientes.

Após a submissão do formulário, a palavra-passe é encriptada com o algoritmo bcrypt antes de ser armazenada na base de dados. O login do cliente é realizado com base na combinação de email e palavra-passe, sendo a verificação efetuada com bcrypt.compare. Em caso de autenticação válida, é gerado um token JWT contendo o identificador do utilizador e o respetivo papel (role), que é armazenado num cookie com tempo de expiração definido, permitindo a gestão segura da sessão do cliente.

6. Funcionalidades Implementadas na 2ª Milestone

Na página inicial da aplicação de front-end o cliente consegue visualizar todos os restaurantes existentes na aplicação, sem existir qualquer tipo de consideração pela localização dos mesmos.



O componente base da aplicação foi definido inicialmente como o componente Dashboard. Foi o componente associado à rota raiz. É aqui que o cliente consegue visualizar todos os restaurantes. É neste componente que:

Consoante a escolha do utilizador, é feita a seleção do tipo de encomenda a ser feita: a levantar no restaurante (pickup) ou como entrega ao domicílio (delivery). Essa escolha é armazenada no CartService e serve para que ao fazer o checkout/finalização da compra, os dados de como vai ser feita essa encomenda sejam restringidos pelo tipo de compra. Como o cliente fica bloqueado de escolher “delivery” quando escolhe “pickup”, ao selecionar a barra correspondente que filtra os restaurantes pela localização, esta informação vai juntamente com o carrinho (a escolha é armazenada no CartService) para que ao fazer o checkout/finalização da compra, os dados da encomenda ao restaurante ficam restringidos ao tipo de encomenda escolhida.

De outra forma não seria possível saber se as refeições encomendadas pelo cliente, a um determinado restaurante, seriam de entrega ao domicílio ou de levantamento no próprio restaurante

Onde é feita a visualização de menus e pratos. Através do RestaurantsService o componente consegue selecionar um restaurante que tenha sido aprovado pelo administrador geral. Consegue visualizar apenas os menus que se encontram ativos desse restaurante, e as suas respetivas refeições. Apenas refeições incluídas em menus é que conseguem ser visualizadas pelo cliente.

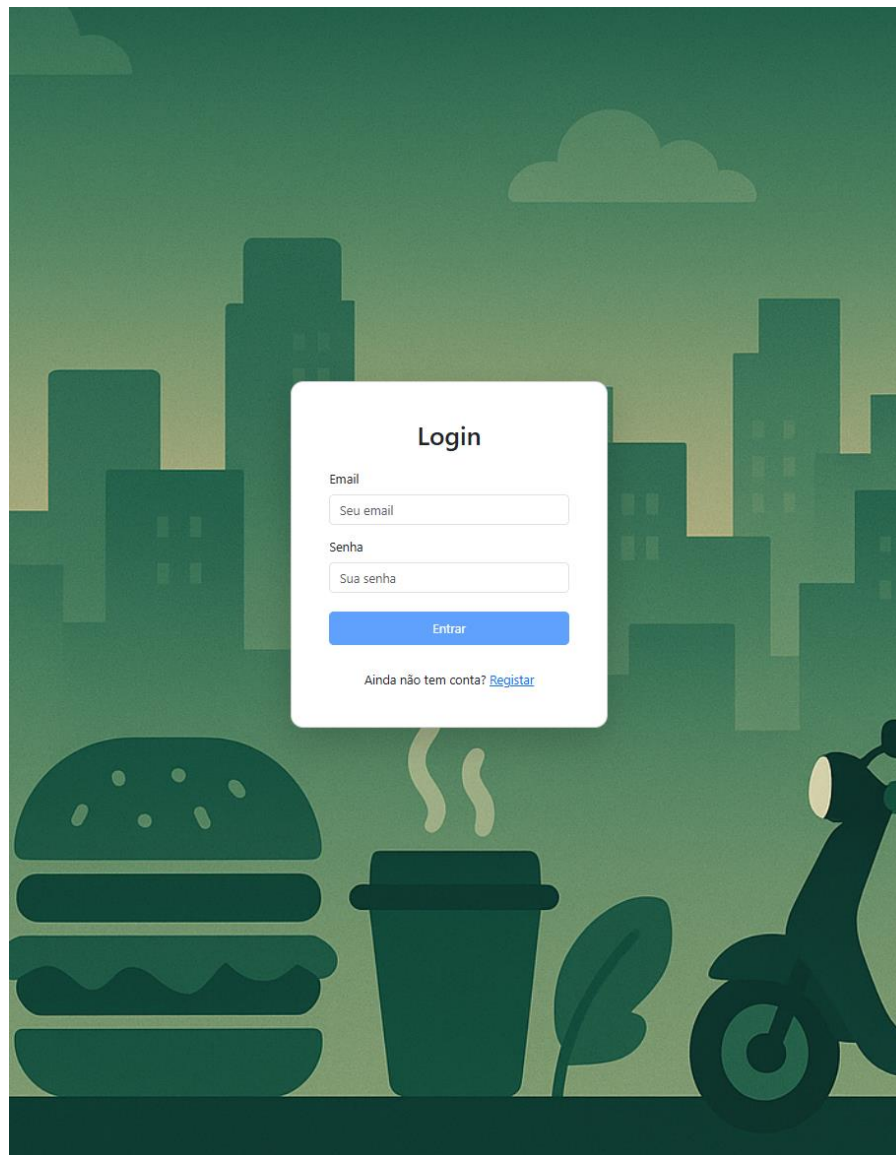
Onde é possível adicionar ao carrinho de compras os pratos selecionados. A adição de refeições ao carrinho de compras é feita através do método addToCart que comunica diretamente com o CartService. Desta forma é garantido que o restaurante associado ao carrinho, e que as suas refeições com respetivos tamanhos e quantidades, são adicionadas corretamente ao carrinho.

É feita a visualização de comentários (através do componente de CommentsList). A interação entre estes dois componentes é controlada pelas funções viewComments e closeComments. Quando o utilizador clica para ver comentários de um restaurante, é evocado o método viewComments de um restaurante (através do ID do restaurante). Este ID é recebido pelo componente de Comments que faz um pedido ao serviço de Comments. O serviço de Comments por sua vez faz o pedido à API REST.

É possível visualizar a distância da localização dos restaurantes em relação à morada na ficha de cliente (através do serviço de Location). Esta componente não lida diretamente com localização ou distâncias, mas integra o componente RestaurantLocator que encapsula este comportamento.

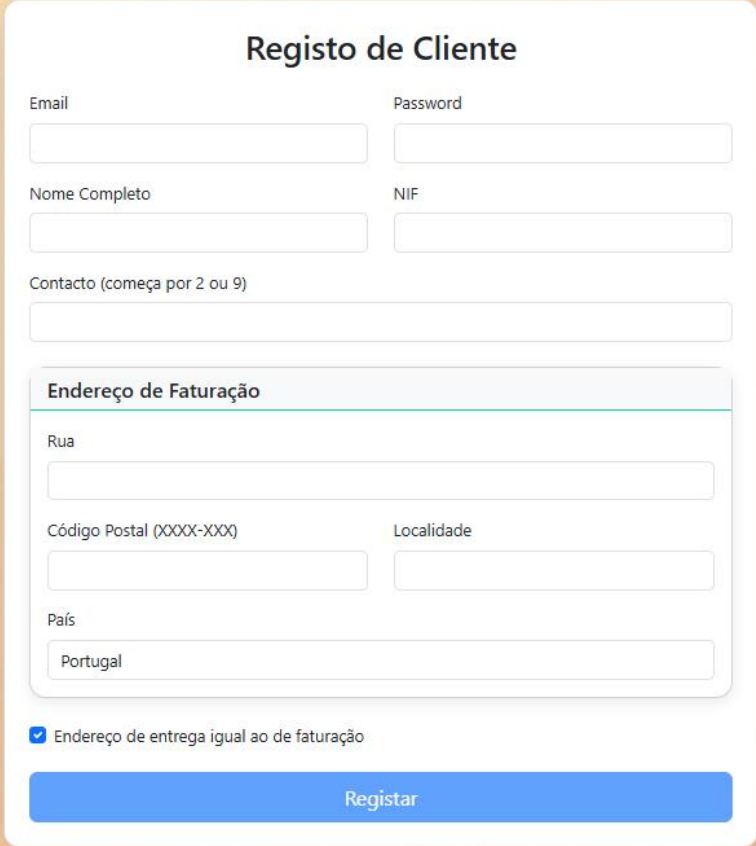
Depois do cliente autenticar-se pode finalizar a compra com as refeições que foram adicionadas ao carrinho (com pedidos ao serviço de Cart à componente [CartOverlay](#)). Os items são adicionados ao carrinho com recurso ao serviço de CartService. O serviço Cart notifica os subscritores de que o carrinho é alterado, e como o componente de CartOverlay está subscrito a este serviço as refeições são atualizadas no carrinho, com respetivas quantidades e preços.

Caso o cliente ainda não tenha feito login é reencaminhado para a respetiva página, caso tente finalizar a sua compra ou aceder a campos relacionados com a ficha de cliente:



Se o cliente já estiver registado, o token JWT é recebido e guardado em *local storage*. É validado o conteúdo do carrinho, pois pode adicionar refeições sem ter feito login, e caso o restaurante seja inválido (se ficar a uma distância superior a 30km da morada do cliente, ou se a morada de entrega do cliente estiver fora do raio de entrega desse restaurante), o seu conteúdo é completamente removido.

Se o cliente ainda não estiver registado:



Registo de Cliente

Email

Password

Nome Completo

NIF

Contacto (começa por 2 ou 9)

Endereço de Faturação

Rua

Código Postal (XXXX-XXX)

Localidade

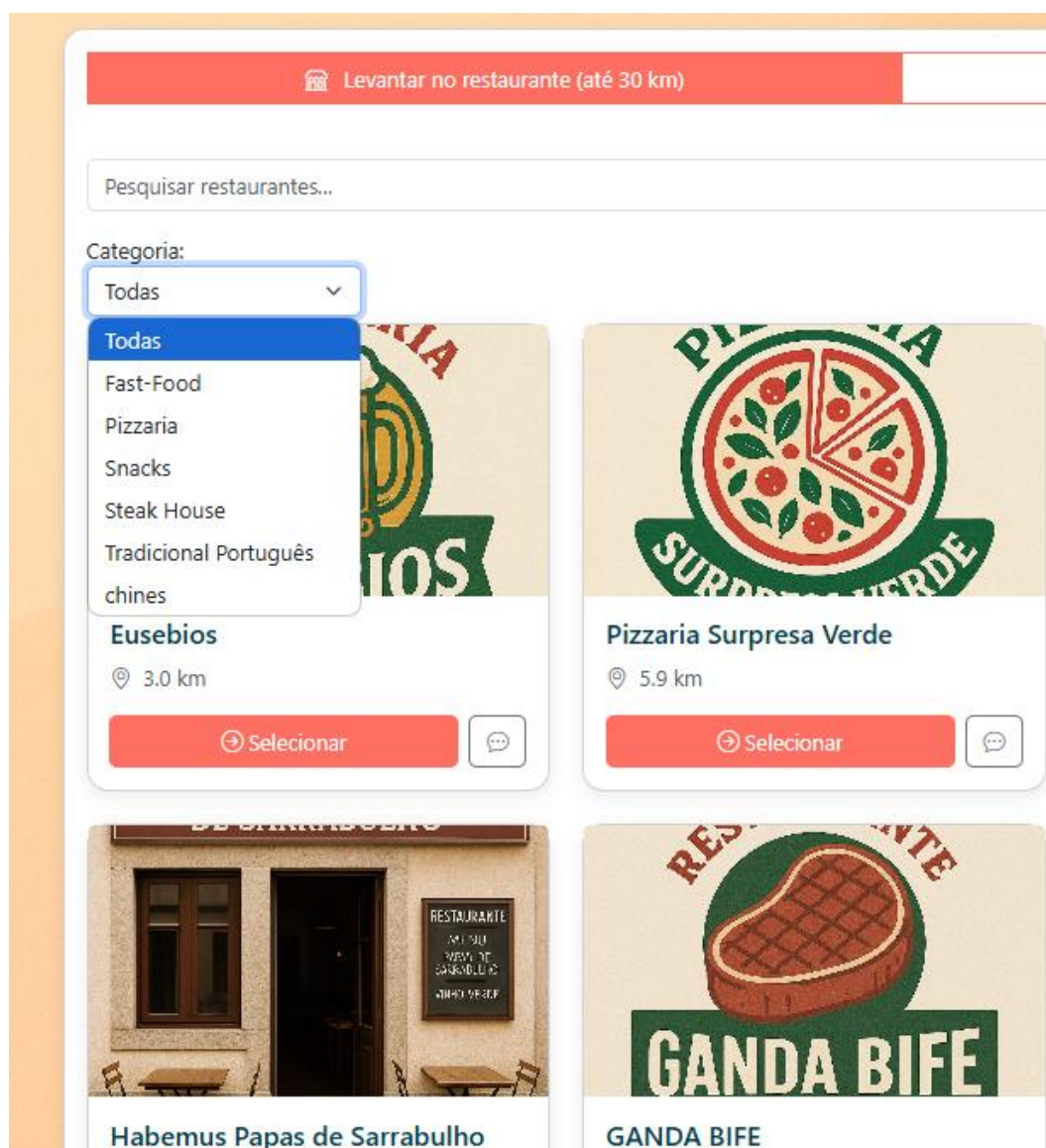
País

☒ Endereço de entrega igual ao de faturação

Registar

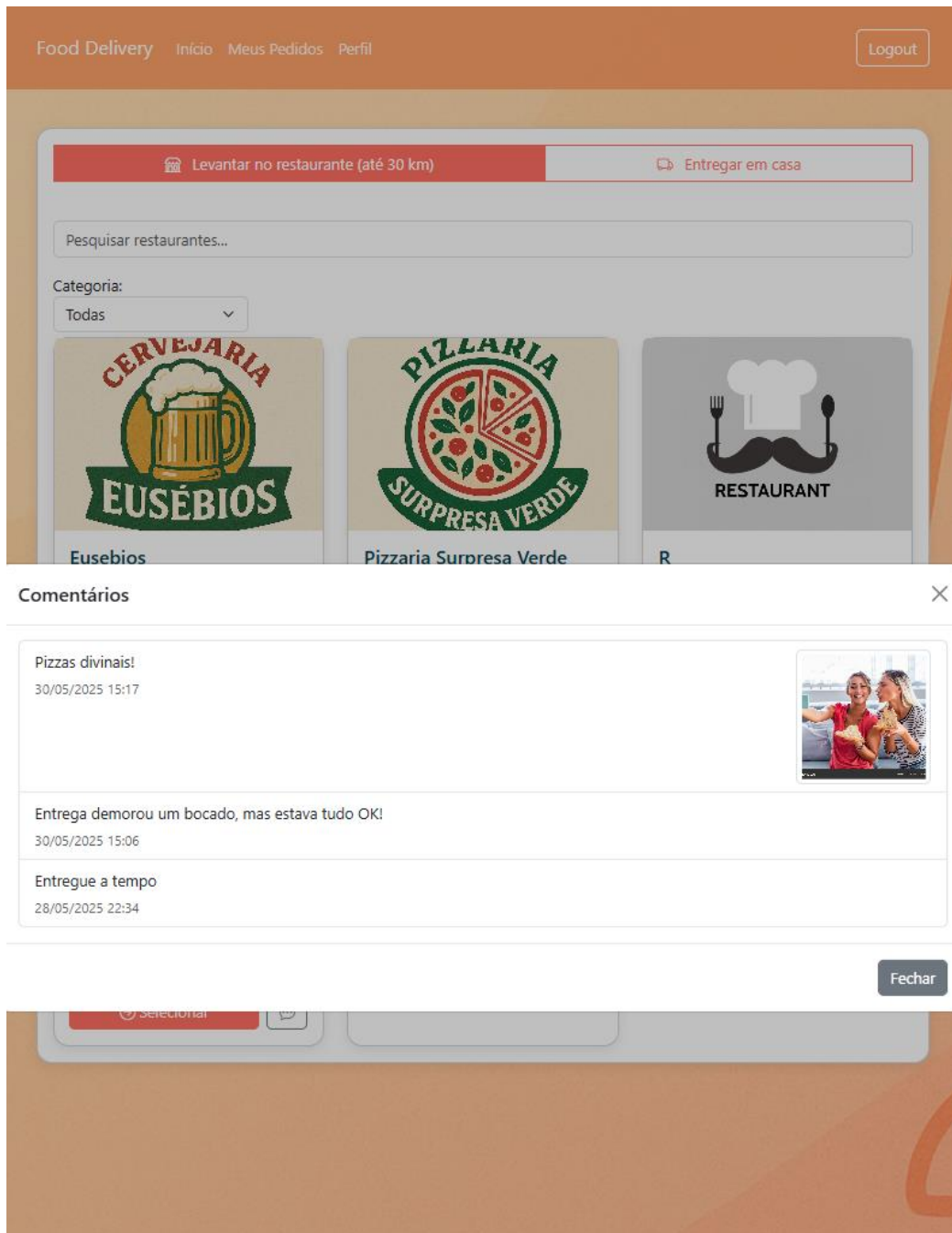
Este terá que fornecer todos os seus dados. A componente de *register* faz um pedido ao serviço de *auth*, que por sua vez trata de fazer o pedido à API REST através de um POST. Por sua vez o serviço devolve um token JWT, que é guardado em local storage, e o cliente é encaminhado para a página principal.

O cliente pode filtrar os restaurantes por categoria:



O componente de Restaurant Locator é utilizado para filtrar os restaurantes por categoria. Inicialmente faz um pedido ao serviço de Restaurants de todas as categorias. É feita uma filtragem com base nas categorias dos restaurantes que se encontram a uma distância útil do utilizador. A componente de Restaurant Locator encapsula toda a lógica de pesquisa de restaurantes, combinando as suas respetivas localizações e distâncias em função da morada do cliente.

Esta componente faz a pesquisa de categorias e restaurantes com recurso a pedidos ao serviço de Restaurants, geolocalização com recurso a pedidos ao serviço de Location, morada de entrega com recurso ao serviço de Clients e carrega imagens com recurso ao serviço de Image.



A função de visualização de comentários a restaurantes é delegada ao componente CommentsList. Faz uso do serviço Comments para obter a lista de comentários. O serviço Comments comunica com a API REST para carregar a lista de comentários do servidor através de um endpoint para o efeito. Quando o componente arranca faz a chamada ao serviço de Comments de imediato.

O utilizador pode sempre mudar a sua morada de entrega (deliveryLocation) e os restaurantes para entrega em casa e de levantar no restaurante irão ser atualizados. Como mostra na imagem:

Food Delivery

[Início](#) [Meus Pedidos](#) [Perfil](#)

Logout

🏠 Levantar no restaurante (até 30 km)

🏠 Entregar em casa

Rua

Rua Nossa Senhora de Fátima 10

Código Postal

4800-436


Cidade

Guimarães


📍 Confirmar morada

Pesquisar restaurantes...


Categoria:
Todas



Pizzaria Surpresa Verde
📍 0.0 km (máx. 3 km)
👉 Selecionar

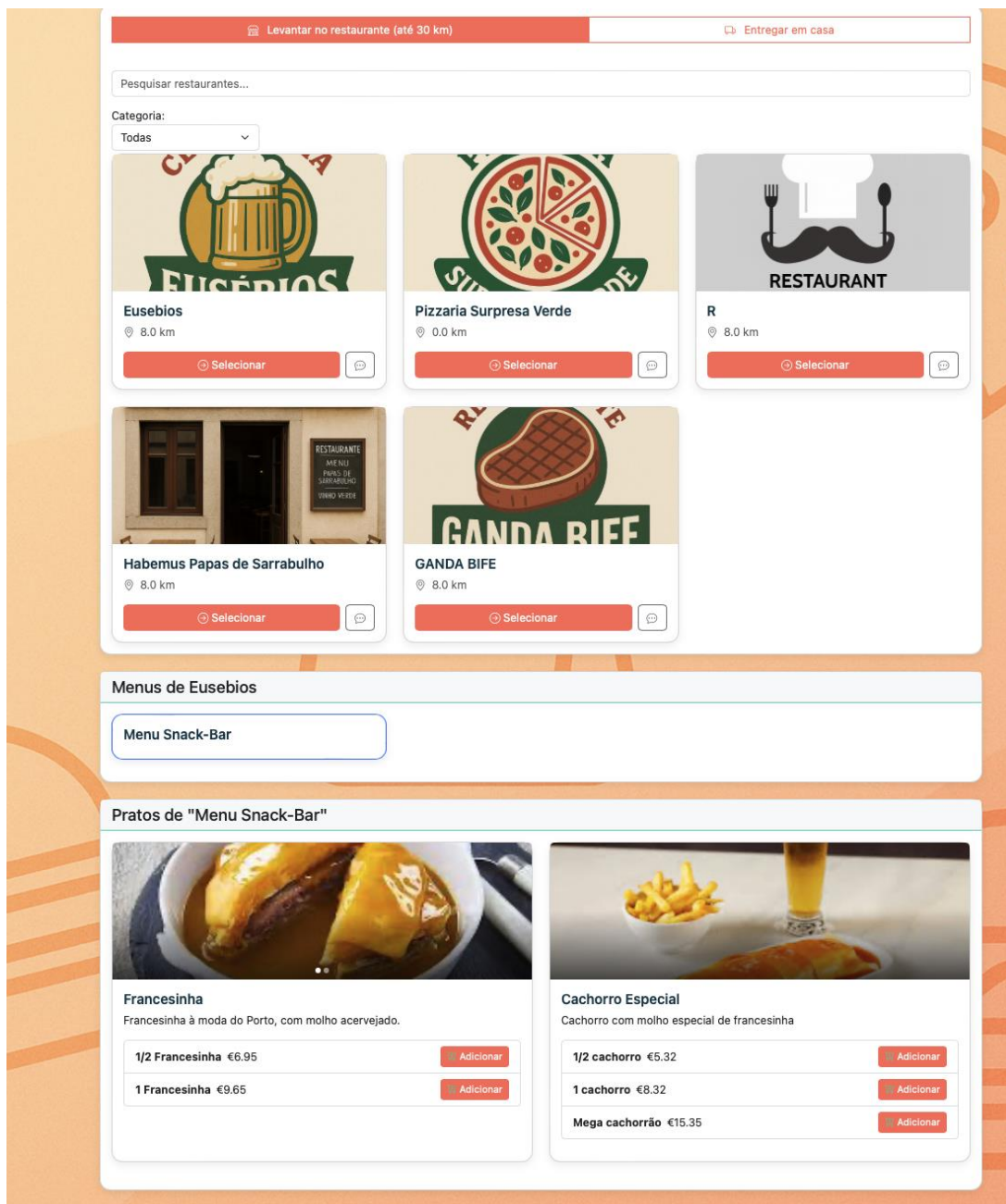


R
📍 8.0 km (máx. 50 km)
👉 Selecionar



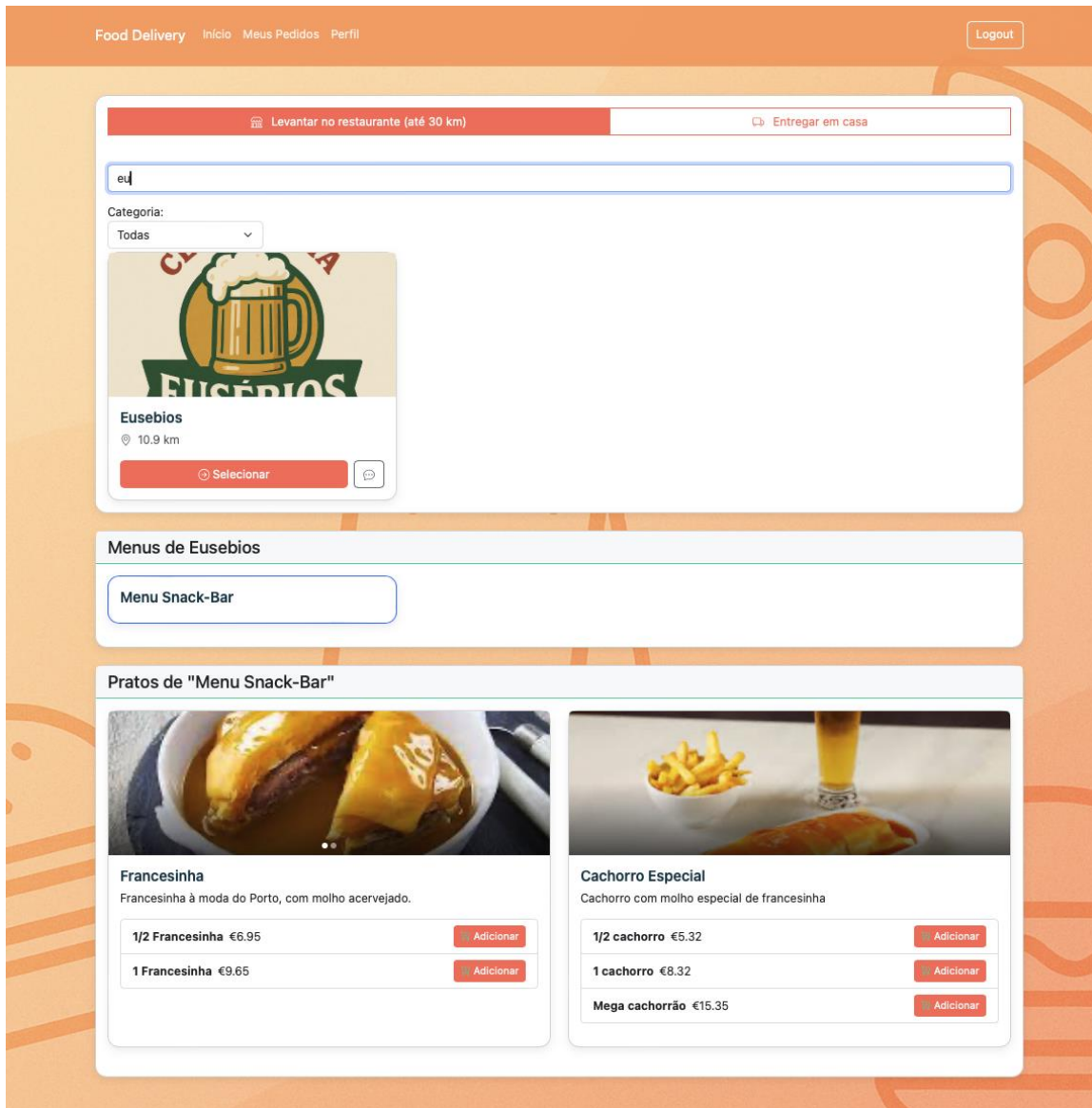
GANDA BIFE
📍 8.0 km (máx. 10 km)
👉 Selecionar

Caso o cliente mude carregue num dos restaurantes irá aparecer os menus e após escolhido o menu irá aparecer os pratos que o constituem:



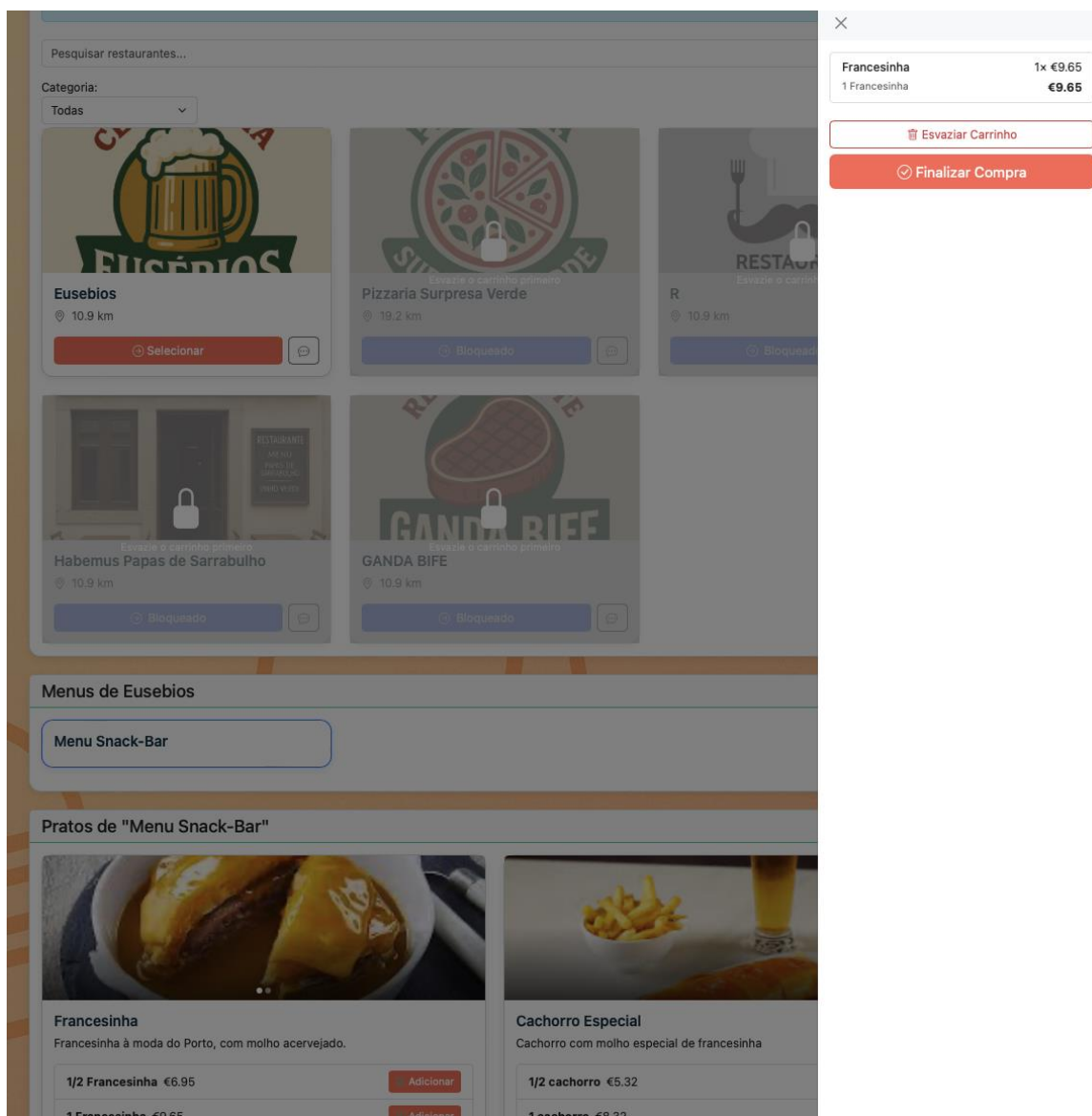
O utilizador ao clicar num restaurante é emitido um evento com os dados do restaurante escolhido. O componente de Dashboard recebe o restaurante selecionado e exibe os seus menus e pratos associados na interface. Os pratos de cada menu ficam visíveis ao utilizador, com preço, imagens, nome e respetivas doses, para poderem ser adicionados ao carrinho.

É também possível a pesquisa de restaurantes pelo nome:



O cliente escreve o nome no campo de pesquisa. Este input é recebido pela componente de Restaurant Locator que por sua vez faz a filtragem com base nesse input. O fluxo é todo feito através da emissão de eventos, pois a componente de Search Bar é uma componente simples.

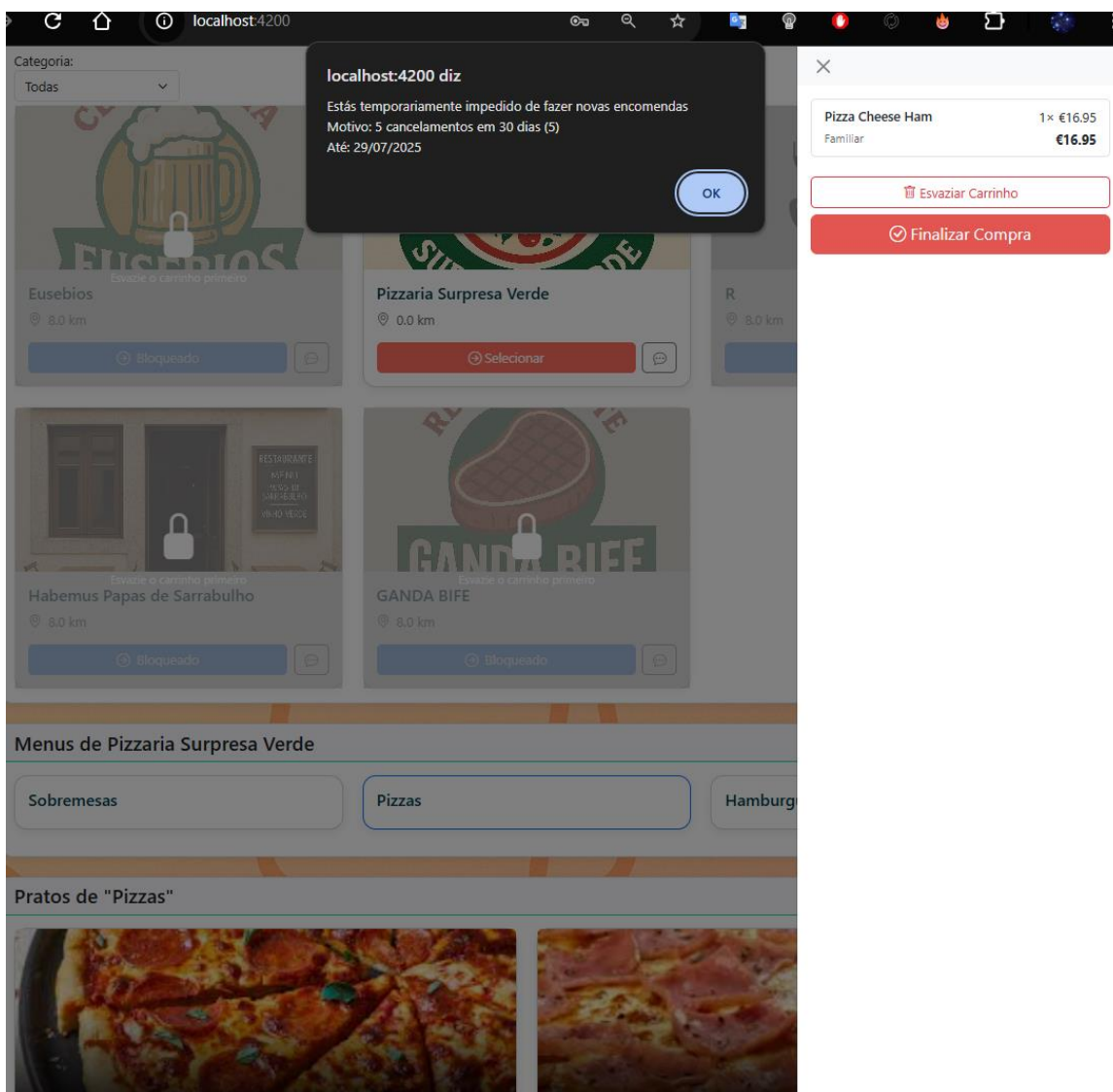
Depois de escolhidas as refeições é possível visualizá-las no carrinho:



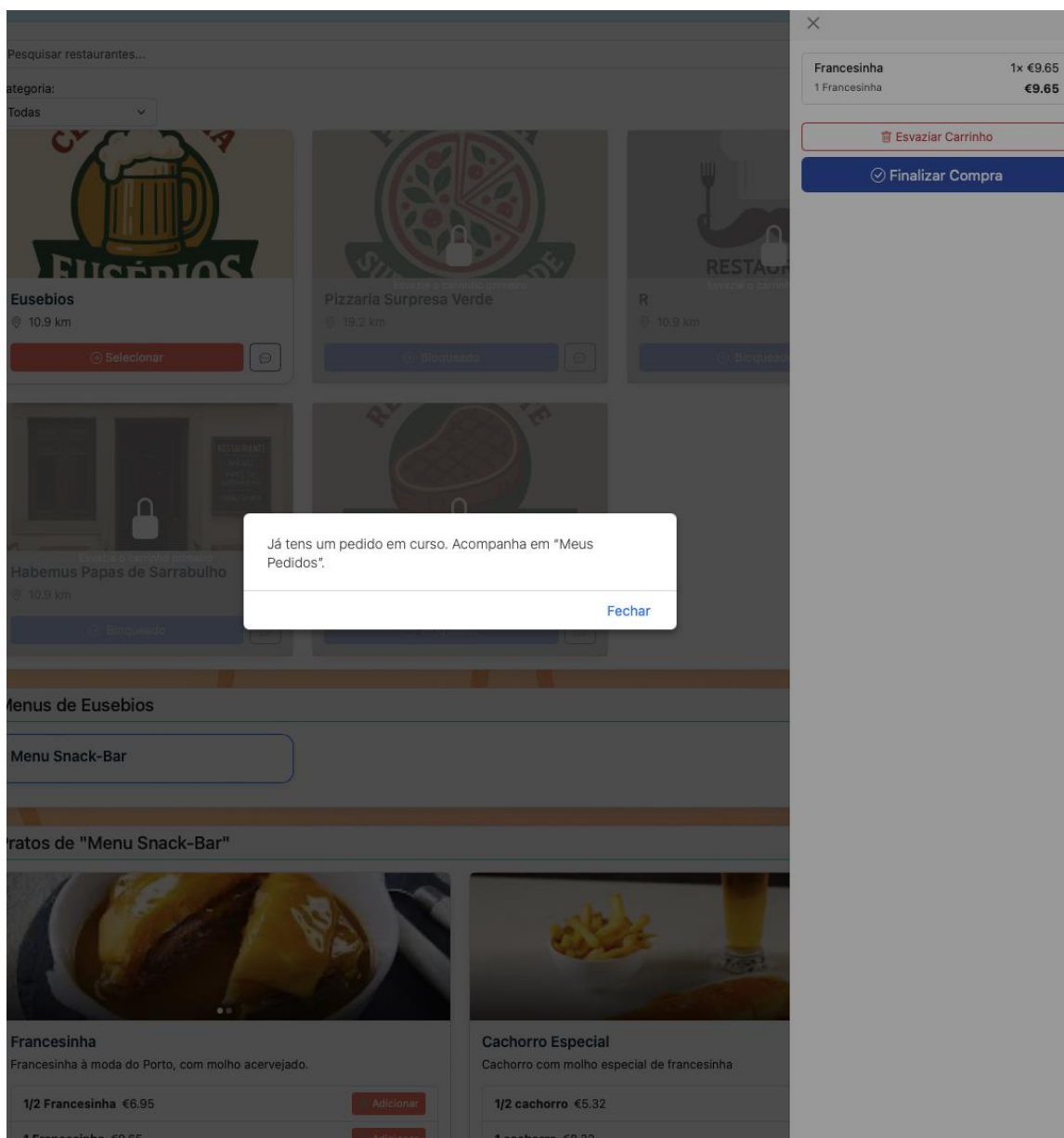
O componente de Cart Overlay é o componente responsável por apresentar num *sidebar* flutuante as refeições escolhidas pelo cliente. É possível esvaziar o carrinho ou finalizar a compra e avançar para o *checkout*. Este componente subscreve o serviço de Cart para poder ler em tempo real o que é adicionado ao carrinho. Por fim lê *snapshots* de clientes e restaurantes, para que essa informação possa ser transmitida para o checkout no serviço de cart. Também verifica se já existe alguma encomenda pendente, e neste caso o cliente fica impedido de prosseguir para o checkout.

Como o cliente pode optar por escolher outra morada em tempo real, é importante esta informação ser replicada nesta altura. O padrão de serviço e componente via Observable garante a sincronização do carrinho e reflete qualquer mudança do cliente.

Caso o cliente tente iniciar o checkout com refeições desse restaurante, e se estiver na blacklist associada a esse restaurante, então é impedido de prosseguir para o checkout.



Também é impedido de prosseguir para o checkout caso tenha uma encomenda pendente.



Ao finalizar a compra o cliente é reencaminhado para o checkout:

Food Delivery

Início

Meus Pedidos

Perfil

Logout

Resumo da Encomenda

Informação Nutricional Inteligente

"Parece que alguém está com um apetite duplo por pizzas de queijo e presunto! Duas meias doses, mas de diferentes categorias, se não me engano... Uma parece ser uma opção mais acessível (10.95€) e a outra, bem, parece ser uma versão um pouco mais "luxuosa" (16.95€). Provavelmente com ingredientes de melhor qualidade ou uma massa mais refinada.

Agora, vamos falar sobre a informação nutricional... Uma pizza de queijo e presunto típica pode variar bastante em calorias e nutrientes dependendo da massa e dos ingredientes. Geralmente, uma fatia de pizza de queijo e presunto pode ter algo em torno de 250-350 calorias, com uma boa quantidade de carboidratos, proteínas e, claro, gordura. O queijo é rico em proteínas e cálcio, mas também em gorduras saturadas. O presunto, por sua vez, é uma boa fonte de proteínas, mas pode ser alto em sódio.

Em resumo, delícia garantida, mas com moderação, claro! Uma refeição assim pode facilmente chegar a umas 1000 calorias ou mais. Vale a pena saborear e aproveitar, mas também é bom ter em mente a balança e a saúde cardiovascular."

Cliente

Salazar Eusébio (cheiodefome@gmail.com)

 Escolheu a opção de levantar no restaurante

Restaurante

Pizzaria Surpresa Verde

Tipo de Encomenda

Pickup

Itens

1× Pizza Cheese Ham (Média)	€10.95
1× Pizza Cheese Ham (Familiar)	€16.95
Total: €27.90	

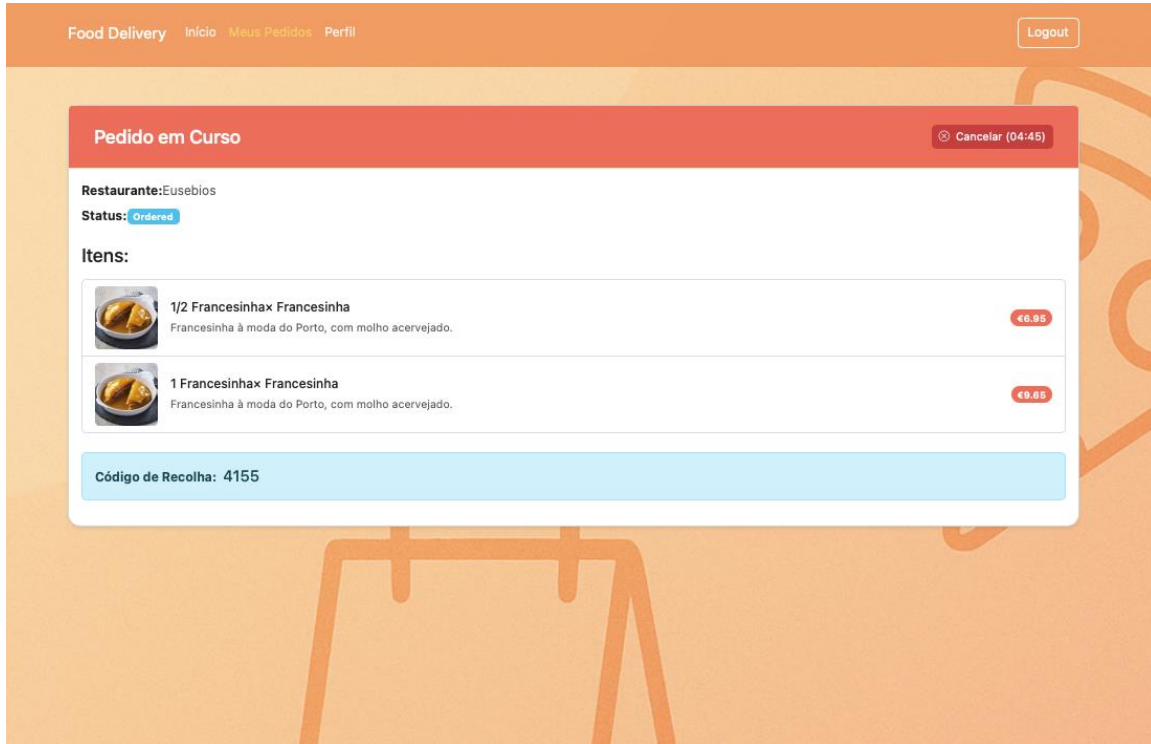
← Cancelar Encomenda

✓ Confirmar Encomenda

O componente de Cart Overlay lê do serviço de Cart o snapshot do restaurante e do cliente, e verifica se há encomendas pendentes. Depois do cliente clicar no botão de finalizar encomenda, é evocado o checkout. Os dados são recebidos pela componente de Checkout que constrói toda a informação recebida numa encomenda (order) para ser enviada para o servidor. A informação é construída através dos snapshots anteriormente criados.

Para além disso, é também construído um prompt para ser enviado para uma API externa de Inteligência Artificial, para analisar o conteúdo do carrinho do cliente e dar uma opinião simpática e informal sobre as refeições e o seu conteúdo nutricional. Este prompt é mostrado ao cliente antes de poder confirmar a sua encomenda.

Ao finalizar a encomenda o cliente é redirecionado para a página de “Meus pedidos”:



O objetivo é o de mostrar ao cliente os detalhes da encomenda pendente, apresentar um código de levantamento e um temporizador. O cliente tem 5 minutos para cancelar o seu pedido.

O tempo sobrannte é calculado com base na data/hora criada no documento da encomenda (order), e a partir da data/hora atual calcula o tempo sobrannte. Neste período de tempo o cliente consegue cancelar o pedido com sucesso, a não ser que o estado do pedido avance no *backoffice* por instrução de um funcionário.

Caso o cliente cancele 5 encomendas nos últimos 30 dias nesse restaurante, é adicionado a uma blacklist. No carrinho se o cliente tentar iniciar o checkout com refeições desse restaurante, e estiver na blacklist associada a esse restaurante, então é impedido de prosseguir para o checkout.

O mecanismo de SSE notifica o servidor e o cliente em tempo real da remoção, ou da alteração do estado da encomenda.

O cliente pode consultar o seu Perfil para fazer alteração de dados pessoais, visualizar o histórico de encomendas e comentar as encomendas desse restaurante:

Food Delivery Início Meus Pedidos Perfil Logout

O Meu Perfil

Informações Pessoais

Nome: z Email: z@z.pt

NIF: 321456987 Contacto: 910031816

Morada de Faturação

Rua: rua carvalho

Código Postal: 4650-502 Localidade: felgueiras

País: Portugal

Morada de Entrega

Rua: Rua Nossa Senhora de Fátima 10

Código Postal: 4800-436 Localidade: Guimarães

País: Portugal

Guardar Alterações Mudar Password

Histórico de Encomendas

Eusebios	4054	5/30/25, 3:20 AM	Delivered	Comentar	Ver Items
----------	------	------------------	-----------	----------	-----------

Ao carregar a página a componente de orders history faz um pedido ao serviço de Orders e preenche com todas as encomendas passadas. Cada encomenda apresenta um botão que permite expandir os detalhes dessa encomenda, com as respetivas refeições e imagens.

Caso ainda não o tenha feito pode clicar em “comentar” e deixar um comentário com uma imagem.

Ao consultar o histórico de encomendas é aqui que pode fazer um comentário ao seu pedido, onde pode submeter uma imagem juntamente com o comentário:

Código Postal

4810-081

Localidade

Vizela

País

Portugal

Guardar Alterações

Mudar Password

Histórico de Encomendas

Pizzaria Surpresa Verde

480e

5/30/25, 12:42 PM

Delivered

Comentar

Ver Itens


Escreve o teu comentário

Pizzas divinais

Fotografias (até 3)

Escolher Ficheiros

pizza-selfie.jpg



Enviar

Cancelar

Pizzaria Surpresa Verde

49c5

5/30/25, 12:32 PM

Delivered

Comentar

Ver Itens

Pizzaria Surpresa Verde

4911

5/30/25, 12:31 PM

Canceled

Ver Itens

Pizzaria Surpresa Verde

4856

5/30/25, 11:59 AM

Delivered

Comentar

Ver Itens

Pizzaria Surpresa Verde

4818

5/30/25, 11:59 AM

Canceled

Ver Itens

Pizzaria Surpresa Verde

4765

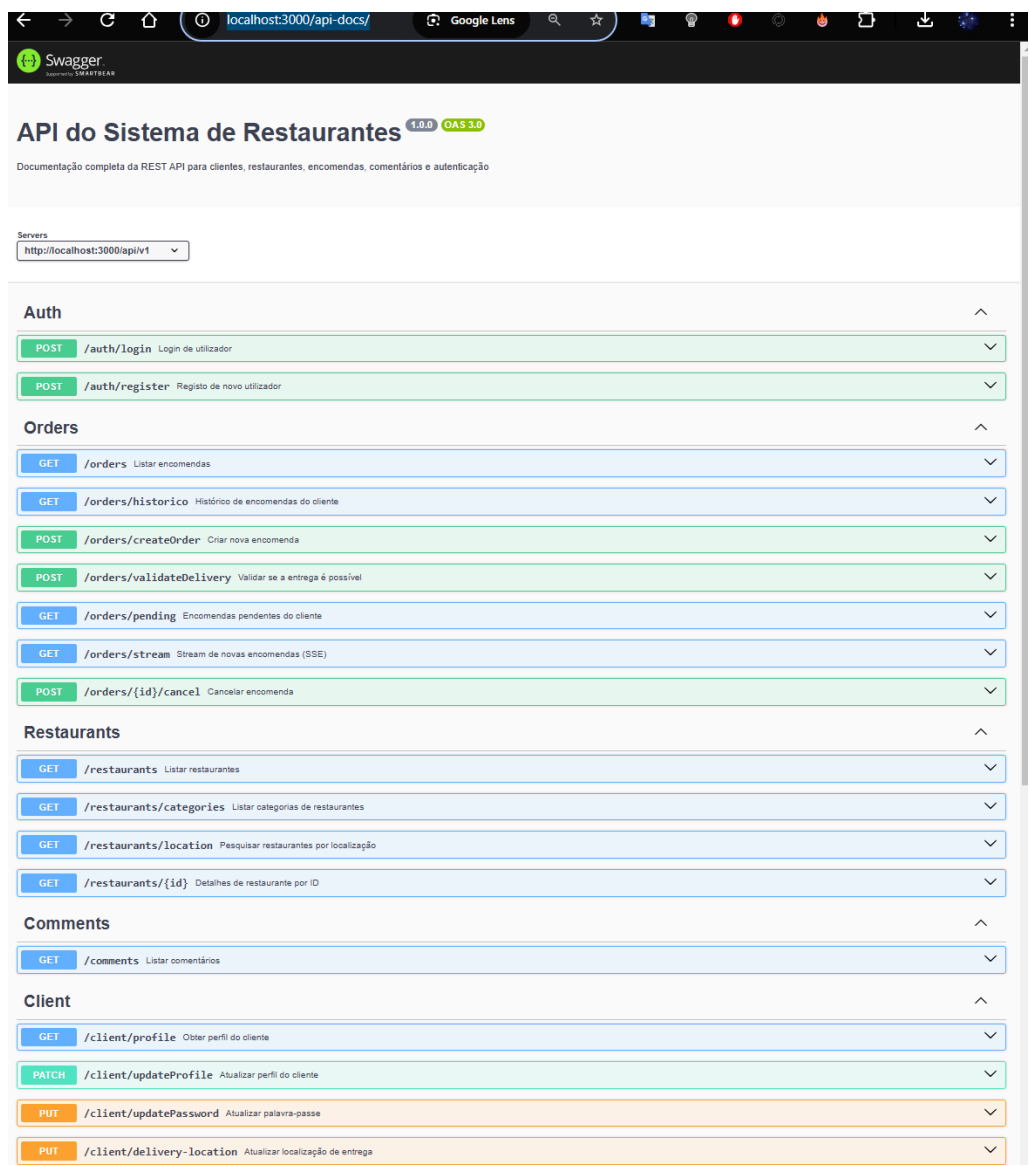
5/30/25, 11:51 AM

Delivered

Comentar

Ver Itens

Por fim foi implementada a documentação da API REST através do Swagger:



7. Desafios e Decisões

A escolha pela utilização de JWT (JSON Web Tokens) em conjunto com cookies no lugar de sessões tradicionais baseadas em servidor deve-se à flexibilidade e escalabilidade oferecida por esta abordagem. Com JWT, as credenciais são assinadas digitalmente e podem ser verificadas sem necessidade de armazenar sessões no lado do servidor, o que reduz a dependência de armazenamento em memória e facilita a escalabilidade horizontal da aplicação.

O recurso a APIs externas para validação de moradas e obtenção das respetivas coordenadas geográficas foi uma decisão estratégica. Esta integração permite assegurar que os endereços inseridos pelos utilizadores são válidos e que correspondem a localizações reais. A geração das coordenadas torna-se particularmente útil para a funcionalidade de cálculo de distância entre o cliente e o restaurante, permitindo determinar automaticamente se o pedido se encontra dentro do raio de entrega estabelecido pelo restaurante.

Adicionalmente, a aplicação está preparada para integração com APIs de validação de contactos telefónicos. Embora nesta fase o foco esteja em utilizadores localizados em Portugal, esta abordagem modular permite que, futuramente, a aplicação seja facilmente escalada para suportar outros países, beneficiando desde logo de uma base técnica preparada para esse cenário.

A segurança da aplicação é reforçada pela utilização de middlewares de proteção de rotas. Estes middlewares verificam se o utilizador está autenticado e se possui o papel (role) necessário para aceder à funcionalidade solicitada, garantindo assim o controlo de acessos conforme o perfil do utilizador (cliente, funcionário, administrador de restaurante ou administrador da aplicação).

Foi ainda definido um layout base que serve de estrutura comum para todas as páginas da aplicação. Esta uniformização facilitou o desenvolvimento do front-end, promovendo uma interface mais harmoniosa, coesa e intuitiva.

Por fim, tanto os restaurantes como as categorias (sejam elas de pratos ou de restaurantes) carecem de aprovação por parte do administrador da aplicação. Esta validação centralizada garante que apenas conteúdos apropriados e validados são disponibilizados, permitindo que

exista uma entidade de supervisão responsável por assegurar a integridade e qualidade da informação na plataforma.

Em relação ao front-end, uma das partes críticas ao seu funcionamento foi a forma de como o carrinho seria implementado. Como o carrinho teria que ser constantemente atualizado pelo cliente, e o seu estado influenciaria o comportamento de várias componentes diferentes, decidiu-se recorrer ao BehaviorSubject. Esta classe pertence à biblioteca RxJS que serve para gerir e partilhar estados reativos entre componentes e serviços, o que permitiu solucionar o carrinho. O uso desta classe foi muito útil pois o componente CartOverlay mostra os itens do carrinho em tempo real, o componente de Checkout, recebe os itens do carrinho, o componente Login verifica se os itens do carrinho são válidos e o componente PendingOrders limpa o carrinho.

Um dos desafios foi o de gerir os pedidos à API de geolocalização, pois facilmente se esgotava o limite de pedidos num minuto. Por causa desta limitação pensamos também como otimizar melhor os pedidos ao nosso servidor de *backend*, para que não fossem feitos pedidos sempre que o cliente navegasse em páginas diferentes, e regressasse a páginas cuja informação dependia da informação vinda do servidor por pedidos à API REST. Inicialmente sempre que o cliente alternava entre o perfil e a página inicial os pedidos eram refeitos. Acabamos por minimizar os pedidos e a informação é guardada em localstorage.

Finalmente o maior problema foi o do sincronismo de encomendas feitas pelo cliente ao *backend* e a atualização do estado da encomenda no *front-end*. Inicialmente o que acontecia é que quando o cliente finalizava o seu pedido, um funcionário no *backoffice* teria que fazer *reload* à página para que fosse informado de encomendas.

Se a página não fosse manualmente atualizada o *backoffice* nunca teria a informação atualizada de encomendas. De forma análoga o cliente com a página de “meus pedidos” aberta nunca veria o seu pedido atualizado sem atualizar manualmente a página.

Para resolver este problema decidiu-se implementar a tecnologia de Server-Sent Events (SSE). Esta tecnologia permite ao servidor enviar dados em tempo real para o cliente, e foi também utilizada para atualizar o *backend* para receber encomendas novas, e também para o cancelamento de encomendas por parte do cliente em tempo real.

Escolheu-se esta tecnologia como alternativa a websockets pois o único estado em tempo real, no contexto deste projeto, atualizado em servidor foi o de cancelamento de pedidos de cliente dentro do prazo estipulado de 5 minutos. Todas as restantes comunicações em tempo real são feitas do servidor para o cliente. A comunicação é praticamente unidirecional. Para além disso é compatível com pedidos REST, como já tínhamos a API REST implementada foi apenas uma questão de adicionar os endpoints necessários.

8. Conclusão

A primeira milestone deste projeto teve como principal objetivo o desenvolvimento da base funcional do sistema de backoffice, assegurando a estrutura necessária para a gestão eficiente de utilizadores, restaurantes, pratos, menus e encomendas. Com foco na robustez e segurança da aplicação, foram implementadas funcionalidades essenciais como o registo e autenticação de utilizadores (clientes, funcionários e administradores), a gestão de conteúdos sujeitos a aprovação por parte da entidade administradora da aplicação, e a utilização de mecanismos de proteção de rotas com verificação de permissões de acesso.

Optou-se por uma arquitetura moderna baseada em JWT e cookies para autenticação, privilegiando a escalabilidade e a eficiência da aplicação, ao mesmo tempo que se garantiu uma experiência de utilização fluída e segura. A integração com APIs externas — nomeadamente para a validação de moradas e obtenção de coordenadas geográficas — revelou-se fundamental não apenas para garantir a precisão dos dados inseridos pelos utilizadores, como também para sustentar, a longo prazo, funcionalidades como o cálculo automático da viabilidade de entregas com base na distância.

Adicionalmente, a estrutura modular do sistema, o uso consistente de layouts e a validação rigorosa de dados colocam a aplicação numa posição favorável para futuras expansões e internacionalização, estando já preparada para suportar cenários em diferentes regiões geográficas e culturais.

Uma parte que podemos melhorar é na verificação nos middleware, ou seja, enquanto estamos apenas a fazer verificações com roles e login feito, pode-se adicionar verificações de restaurante, para que ao alterar o *id* no *url*, um utilizador não possa aceder a informações de outros restaurantes.

Esta fase constituiu uma base sólida para a evolução do projeto na milestone seguinte, onde foram abordadas as interações entre cliente e restaurante no front-end e a finalização do ciclo completo da encomenda.

O projeto de front-end foi concebido com o propósito principal de proporcionar uma experiência de utilização simples e fluída para o cliente final.

Ao longo do desenvolvimento enfrentaram-se vários desafios técnicos, principalmente relacionados com arquiteturas e padrões de software que nunca tínhamos utilizado, e tecnologias com as quais nunca tivemos contacto. A implementação da arquitetura de componentes e serviços foi amadurecendo ao longo do projeto, mas na parte inicial não foi devidamente planeada. A lógica central da aplicação, como o carregamento de restaurantes, foi colocada no componente dashboard, em vez de estar mais distribuída na componente raiz “*app*”.

A componente *app* devia de ter sido o ponto inicial de partida do projeto, e acabou por ser pouco explorada. A ausência desta estrutura inicial levou a duplicações de chamadas em alguns componentes.

O único motivo pelo qual aconteceu foi porque inicialmente desconhecíamos a framework de Angular, e qualquer outra framework de programação web. Para a maior parte do grupo esta disciplina foi o primeiro contacto que tivemos com aplicações web.

Um aspeto extremamente relevante para a aplicação de front-end foi a utilização de Server Sent Events (SSE), para permitir a atualização em tempo real entre cliente e restaurantes. Evitamos a complexidade de web sockets porque é praticamente o servidor a enviar mensagens no nosso projeto. Esta abordagem foi mais simples, mas se for considerada a expansão futura da aplicação, web sockets já pode tornar-se numa solução mais escalável. Os web sockets conseguem lidar com cenários de alta concorrência de informação em tempo real, mas no contexto do nosso projeto decidimos explorar o uso de Server Sent Events.

A integração com um sistema de inteligência artificial no checkout foi feita numa fase tardia do projeto, pelo que não ficou implementada numa componente específica para o efeito. O objetivo inicial era o de fazer *fine tuning* através do Unsloth num dataset de refeições e informação nutricional, e alojar uma LLM (Large Language Model) no repositório da Hugging Face. No entanto, chegamos à conclusão que os pedidos para inferência na Hugging Face eram pagos. Para simplificar o processo descobrimos que existe uma plataforma desenvolvida pela Groq que permite lidar com modelos de machine learning, especificamente LLMs. A sua integração foi relativamente simples e é feita na componente de *checkout*.

Em geral o front-end conseguiu atingir um bom equilíbrio entre usabilidade e desempenho. A estrutura do projeto de front-end é escalável e permite a integração de mais componentes relevantes para a área de negócio, como pagamentos.

O principal desafio no desenvolvimento do projeto de backend e front-end foi o uso de tecnologias e de sintaxe que nos era completamente desconhecida. No entanto, com o auxílio da vasta informação que existe online, e da sua exploração contínua através de exemplos existentes, conseguimos desenvolver com maior ou menor dificuldade todos os requisitos propostos no enunciado do trabalho para a disciplina de PAW. O uso de todas estas ferramentas, bibliotecas e tecnologias permitiu-nos obter uma boa compreensão geral de como funciona o desenvolvimento *web* moderno.