Mestrado em Engenharia Informática

Projeto de Informática

Relatório Técnico

Pack My Bag

PG53601 - Afonso Xavier Cardoso Marques
PG53651 - André Castro Alves
PG53797 - Eduardo Cardoso Pereira
PG53849 - Gonçalo Vilar Vale
PG53923 - João Miguel Faria Leite
PG53929 - João Paulo Peixoto Castro
PG54174 - Renato André Machado Gomes

Braga, 29 de novembro de 2024

Índice

Lista de Figuras

1. Resumo

O objetivo deste relatório técnico é documentar as decisões tomadas e os desenvolvimentos realizados no âmbito da criação da plataforma web da startup Pack My Bag, detalhando os fundamentos técnicos e estratégicos que sustentam a solução proposta. Este documento visa fornecer uma visão clara sobre a abordagem tecnológica adotada, os desafios enfrentados e as soluções implementadas, alinhadas aos objetivos de negócio.

A Pack My Bag posiciona-se como uma startup inovadora, focada em oferecer uma solução prática e sustentável para viajantes que buscam simplicidade e conveniência. A proposta do serviço inclui a possibilidade de os clientes alugarem roupas e acessórios diretamente no destino de viagem, eliminando a necessidade de transportar bagagens volumosas e contribuindo para a redução de custos relacionados ao transporte de bagagens em companhias aéreas.

2. Âmbito do Produto

A aplicação Pack My Bag é uma plataforma web inovadora que visa transformar a experiência de viagem dos utilizadores ao oferecer um serviço de aluguer de peças de roupa. O principal objetivo é eliminar a necessidade de transportar bagagem volumosa, permitindo que os viajantes aluguem roupas diretamente no destino da sua viagem.
Através da aplicação, os clientes podem:

- **Selecionar Peças de Roupa:** Explorar um catálogo diversificado de peças de diferentes marcas e lojas, organizadas por categoria e estilo.
- **Criar Conjuntos Personalizados:** Combinar peças para criar conjuntos que reflitam o seu gosto pessoal, com a possibilidade de incluir itens de várias marcas.
- **Optar por Conjuntos Pré-definidos:** Escolher entre conjuntos já preparados para maior conveniência.
- **Agendar Entregas:** Definir o local e a data de recolha das encomendas, seja num hotel, alojamento local, aeroporto ou outro ponto à sua escolha.
- **Receber Recomendações Personalizadas:** Aceder a sugestões de vestuário fornecidas por estilistas pessoais, com base num questionário de estilo preenchido pelo cliente.
- **Gerir Favoritos:** Guardar peças e conjuntos numa biblioteca pessoal para futuras consultas e alugueres.
- **Avaliar Peças Alugadas:** Deixar reviews sobre as peças alugadas, ajudando outros utilizadores nas suas escolhas.
- **Receber Notificações:** Manter-se informado sobre o estado das encomendas, disponibilidade de peças e lembretes de devolução.

Para as marcas e lojas parceiras, o PackMyBag oferece uma plataforma para expandir o seu alcance e proporcionar uma nova forma de interação com os clientes, através do aluguer das suas peças.

3. Levantamento de Requisitos

Após termos solidificado a ideia principal para a aplicação, procedeu-se ao levantamento de requisitos com o objetivo de compreender as necessidades e preferências dos potenciais utilizadores. Este processo foi fundamental para orientar o desenvolvimento da plataforma, assegurando que ela atenda efetivamente às expectativas do público-alvo.
Para obter uma visão abrangente dos desejos e necessidades dos clientes, foram utilizadas várias técnicas de coleta de informações:

- **Entrevistas com Viajantes Frequentes:** Foram realizadas entrevistas para entender os desafios enfrentados durante as viagens, especialmente relacionados ao transporte de roupas e bagagens volumosas.
- **Questionários Online:** Distribuímos questionários em plataformas digitais para recolher dados quantitativos sobre hábitos de consumo, preferências de estilo e receptividade a um serviço de aluguer de roupas.
- **Análise de Mercado:** Estudamos serviços similares em outros países para identificar funcionalidades bem-sucedidas.

Este levantamento permitiu alinhar as funcionalidades da aplicação com as reais necessidades dos utilizadores, assegurando que a Pack My Bag ofereça uma experiência valiosa e diferenciada no mercado de aluguer de roupas em Portugal.

3.0.1. Modelação de Requisitos

Para o levantamento de requisitos foi utilizada a *requirement shell* do modelo *Volere* como forma de representação, para os descrever concisamente.
Como caracterização da tabela de representação de requisitos, é necessário descrever os campos:

- **Requisito**: número de identificação do requisito.
- **Tipo**: tipo de requisito.
- **Use Cases**: número dos Use Cases associados.
- **Descrição**: descrição clara e concisa do requisito.
- **Origem**: quem originou o requisito.
- **Prioridade**:índice de prioridade para a implementação do requisito:
  -Must: requisito obrigatório;
  -Should: requisitos que deve ser implementados;
  -**Could**: requisito que não é necessário, mas é desejado;
  -**Won't**: requisito que pode ser considerado posteriormente.

3.1. Requisitos Funcionais

Requisito #: 1	Tipo: Funcional Use Cases #: Consultar informações de peças de roupa e sets
Descrição	**Um cliente deverá ser capaz de consultar o inventário onde estarão disponibilizados todos itens de peças de roupa e sets de roupa.**
Origem	Cliente
Prioridade	Must

Requisito #: 2	Tipo: Funcional Use Cases #: Consultar informações de peças de roupa e sets
Descrição	**O cliente deve ser capaz de visualizar o inventário com itens caracterizados por uma imagem representativa do produto, a sua designação e o preço.**

Requisito #: 2	Tipo: Funcional Use Cases #: Consultar informações de peças de roupa e sets
Origem	Cliente
Prioridade	Must

Requisito #: 3	Tipo: Funcional Use Cases #: Consultar informações de peças de roupa e sets
Descrição	**O cliente deve ser capaz de visualizar cada item numa página individual, onde haverá uma imagem representativa do produto, a sua designação, os tamanhos e as cores disponíveis e o preço de aluguer.**
Origem	Cliente
Prioridade	Must

Requisito #: 4	Tipo: Funcional Use Cases #: Alugar roupa
Descrição	**O cliente deverá ser capaz de adicionar uma peça de roupa ou set no seu cesto virtual.**
Origem	Cliente
Prioridade	Must

Requisito #: 5	Tipo: Funcional Use Cases #: Alugar roupa
Descrição	**O cliente deverá ser capaz de remover uma peça de roupa ou set no seu cesto virtual.**
Origem	Cliente
Prioridade	Must

Requisito #: 6	Tipo: Funcional Use Cases #: Alugar roupa
Descrição	**O cliente deverá ser capaz de efetuar o aluguer do conjunto de itens que se encontram no cesto virtual, fornecendo para isso o destino de recolha, o período de aluguer e informações de pagamento.**
Origem	Cliente
Prioridade	Must

Requisito #: 7	Tipo: Funcional Use Cases #: Comparar preço da viagem vs aluguer
Descrição	**O sistema deve simular o preço da viagem com mala, que varia de acordo com o peso das roupas que se encontram no cesto virtual do cliente, adicionalmente com a distância, a taxa física e a taxa de emissões.**
Origem	Cliente
Prioridade	**Could**

Requisito #: 8	Tipo: Funcional Use Cases #: Gerir coleção de favoritos
Descrição	**O cliente deverá ser capaz de adicionar um item(peça de roupa ou set) para a sua biblioteca pessoal de produtos favoritos.**
Origem	Cliente
Prioridade	Must

Requisito #: 9	Tipo: Funcional	Use Cases #: Gerir coleção de favoritos
Descrição	**O cliente deverá ser capaz de remover um item(peça de roupa ou set) da sua biblioteca pessoal de produtos favoritos.**	
Origem	Cliente	
Prioridade	Should	

Requisito #: 10	Tipo: Funcional	Use Cases #: Filtrar roupa por critérios
Descrição	**O cliente deverá ser capaz de visualizar itens de uma das marcas disponíveis no sistema, género, o seu preço, e o seu tamanho.**	
Origem	Cliente	
Prioridade	Must	

Requisito #: 11	Tipo: Funcional	Use Cases #: Consultar histórico de alugueres
Descrição	**O cliente deverá ser capaz de visualizar todas as encomendas de aluguer que efetuou até ao momento.**	
Origem	Cliente	
Prioridade	Should	

Requisito #: 12	Tipo: Funcional	Use Cases #: Consultar histórico de alugueres
Descrição	**O histórico deverá mostrar a data da realização da encomenda, o período em que foi alugada, a designação dos produtos e o local em que foi ou irá ser recolhido.**	
Origem	Cliente	
Prioridade	Should	

Requisito #: 13	Tipo: Funcional	Use Cases #: Publicar review de roupas
Descrição	**O cliente tem a opção de fazer uma review em peças de roupa que alugou.**	
Origem	Cliente	
Prioridade	Must	

Requisito #: 14	Tipo: Funcional	Use Cases #: Publicar review de roupas
Descrição	**O cliente pode editar as reviews que fez.**	
Origem	Cliente	
Prioridade	Must	

Requisito #: 15	Tipo: Funcional	Use Cases #: Publicar review de roupas
Descrição	**O cliente pode eliminar as reviews que fez.**	
Origem	Cliente	
Prioridade	Must	

Requisito #: 16	Tipo: Funcional	Use Cases #: Consultar notificações
Descrição	**O cliente deve ser capaz de aceder a uma aba/secção na plataforma onde pode visualizar as notificações recebidas.**	

Requisito #: 16	Tipo: Funcional Use Cases #: Consultar notificações
Origem	Cliente
Prioridade	Must

Requisito #: 17	Tipo: Funcional Use Cases #: Consultar recomendações do estilista
Descrição	**O cliente pode pagar por um serviço extra, onde é capaz de aceder a uma aba/secção que lhe permite receber recomendações de um estilista.**
Origem	Cliente
Prioridade	Must

Requisito #: 18	Tipo: Funcional Use Cases #: Consultar recomendações do estilista
Descrição	**O cliente deve ser capaz de enviar um pedido de recomendação a um estilista, respondendo a um formulário.**
Origem	Cliente
Prioridade	Should

Requisito #: 19	Tipo: Funcional Use Cases #: Recomendar outfits
Descrição	**O estilista deve ser capaz de visualizar os formulários dos pedidos dos clientes.**
Origem	Estilista
Prioridade	Should

Requisito #: 20	Tipo: Funcional Use Cases #: Recomendar outfits
Descrição	**O estilista deve ser capaz de enviar a clientes combinações de peças feitas por ele.**
Origem	Estilista
Prioridade	Should

Requisito #: 21	Tipo: Funcional Use Cases #: Alterar estado de uma encomenda
Descrição	**O funcionário deve ser capaz de alterar o estado de uma encomenda.**
Origem	Funcionário
Prioridade	Must

Requisito #: 22	Tipo: Funcional Use Cases #: Gerir peças de roupas
Descrição	**O funcionário tem de ter permissão para adicionar e retirar peças do inventário.**
Origem	Funcionário
Prioridade	Must

Requisito #: 23	Tipo: Funcional Use Cases #: Filtrar roupa por critérios
Descrição	**O funcionário deve ser capaz de procurar por peças por determinados critérios como o "número de encomendas" para ser mais fácil de resolver problemas relacionados com a gestão de itens.**

Requisito #: 23	Tipo: Funcional	Use Cases #: Filtrar roupa por critérios
Origem	Funcionário	
Prioridade	**Could**	

Requisito #: 24	Tipo: Funcional	Use Cases #: Notificar o cliente da disponibilidade de peças de roupa
Descrição	**O sistema deve notificar o cliente, que pediu para ser avisado, quando determinada peça se encontrar novamente disponível para aluguer, disponibilizando a designação do item e a nova disponibilidade.**	
Origem	Sistema	
Prioridade	Should	

Requisito #: 25	Tipo: Funcional	Use Cases #: Notificar estado de trânsito das peças de roupa
Descrição	**O sistema deve notificar o cliente, com informações sobre a sua encomenda sempre que existe uma mudança de estado da encomenda.**	
Origem	Sistema	
Prioridade	Must	

Requisito #: 26	Tipo: Funcional	Use Cases #: Notificar sobre a data de devolução
Descrição	**O sistema deve notificar o cliente que a data de devolução da roupa se está a aproximar, 48h e 24h antes).**	
Origem	Sistema	
Prioridade	Must	

Requisito #: 27	Tipo: Funcional	Use Cases #: Aplicação de coima
Descrição	**Caso a roupa não seja devolvida, o sistema aplica uma coima diária, removendo dinheiro da caução.**	
Origem	Sistema	
Prioridade	Must	

Requisito #: 28	Tipo: Funcional	Use Cases #: Recomendar outfits conforme o conjunto
Descrição	**O sistema deve recomendar roupas ao cliente tendo em conta o seu estilo, alugueres prévios e o destino para que este vai viajar.**	
Origem	Estilista	
Prioridade	Must	

Requisito #: 29	Tipo: Funcional	Use Cases #: Autenticar na plataforma
Descrição	**O cliente, o estilista e o técnico devem ser capazes de fazerem login no sistema, acedendo à sua vista da aplicação e funcionalidades correspondentes.**	
Origem	Cliente, Estilista, Técnico	
Prioridade	Must	

Requisito #: 30	Tipo: Funcional Use Cases #: Autenticar na plataforma
Descrição	**O cliente deve ser capaz de criar uma conta na plataforma.**
Origem	Cliente
Prioridade	Must

3.2. Requisitos Não Funcionais

Requisito #: 31	Tipo: Performance Use Cases #
Descrição	**A aplicação deve ser capaz de escalar de acordo com o número de utilizadores.**
Origem	Cliente
Prioridade	Should

Requisito #: 32	Tipo: Disponibilidade Use Cases #
Descrição	**A plataforma deverá ter uma alta disponibilidade, pelo que deve funcionar corretamente todo o tempo.**
Origem	Cliente
Prioridade	Must

Requisito #: 33	Tipo: Performance Use Cases #
Descrição	**As reviews dadas aos itens podem ser classificadas de uma a cinco estrelas (obrigatoriamente) e com uma avaliação textual opcional.**
Origem	Cliente
Prioridade	Must

Requisito #: 34	Tipo: Usabilidade Use Cases #
Descrição	**O sistema deve ter uma interface atrativa e de fácil usabilidade de modo a que os clientes consigam efetuar os seus alugueres sem constrangimentos.**
Origem	Cliente
Prioridade	Must

Requisito #: 35	Tipo: Manutenção e Suporte Use Cases #
Descrição	**O sistema deve permitir a adição de novas funcionalidades sem interromper o funcionamento atual.**
Origem	Cliente
Prioridade	Must

Requisito #: 36	Tipo: Segurança Use Cases #
Descrição	**O sistema terá uma parte com acesso autorizado e outra com acesso não autorizado sendo que para realizar compras terá de efetuar um processo de autenticação.**

Requisito #: 36	Tipo: Segurança Use Cases #
Origem	Cliente
Prioridade	Must

Requisito #: 37	Tipo: Cultural e Politico Use Cases #
Descrição	**O sistema deverá ter suporte ao idioma Inglês.**
Origem	Cliente
Prioridade	Must

Requisito #: 38	Tipo: Functional Use Cases #
Descrição	**Os tipos de notificação do estado de encomenda devem conter Enviado, Não Entregue, Devolvido, Entregue.**
Origem	Cliente
Prioridade	Must

Requisito #: 39	Tipo: Functional Use Cases # Alugar roupa
Descrição	**O destino de recolha da encomenda deverá ser verificada pelo sistema.**
Origem	Equipa desenvolvimento
Prioridade	Must

4. Arquitetura

Com base na ideia principal consolidada e nos requisitos levantados junto dos potenciais utilizadores, procedemos à elaboração do modelo de domínio, do diagrama de casos de uso e do diagrama de componentes. Estas etapas foram fundamentais para estruturar a arquitetura da aplicação PackMyBag, assegurando que todas as funcionalidades essenciais fossem contempladas e que a interação entre os diferentes componentes fosse eficiente e coerente. Esta abordagem permitiu-nos delinear claramente as entidades, os processos e as interações do sistema, servindo como guia para o desenvolvimento e implementação da plataforma.

4.1. Modelo de Domínio

O Modelo de Domínio representa a estrutura conceitual da plataforma web PackMyBag, incluindo as entidades principais e suas relações. Este modelo descreve como as peças de roupa, os clientes, os funcionários técnicos, as encomendas, e outras entidades interagem entre si no contexto do serviço de aluguer. Foca-se na organização lógica dos dados e na modelação das operações básicas que suportam o funcionamento da plataforma, como a gestão de peças, notificações, e acompanhamento do estado das encomendas.

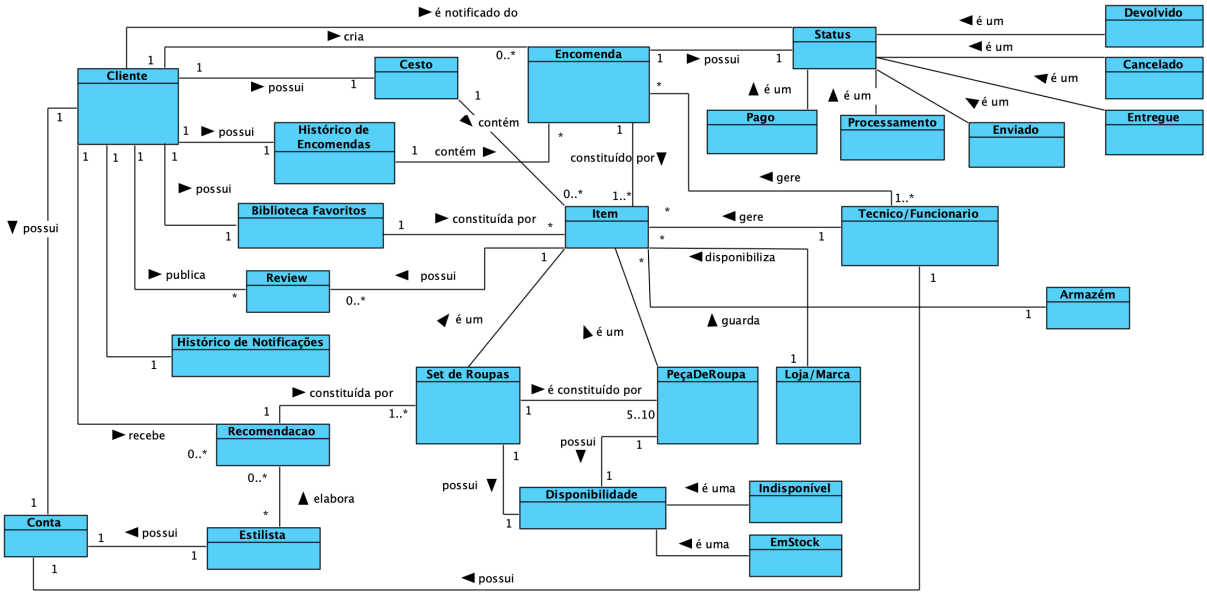

Figura 1: Modelo de Domínio

As principais entidades incluem:

- **Cliente:** Usuário que aluga as roupas.
- **Cesto:** Conjunto de roupas selecionadas pelo cliente para aluguel.
- **Encomenda:** Representa a transação de aluguer, com informações sobre o estado da encomenda.
- **Item:** Cada peça de roupa que compõe o cesto.
- **Biblioteca de Favoritos:** Armazena as peças favoritas do cliente.
- **Review:** Avaliação deixada pelo cliente sobre as peças alugadas.
- **Recomendação:** Sugestões de roupas com base no perfil do cliente.
- **SetDeRoupa:** Conjunto de peças de roupa selecionadas.
- **PecaDeRoupa:** Cada peça individual que compõe um conjunto.
- **Disponibilidade:** Rastreamento da disponibilidade das peças.
- **Conta:** Informações da conta do cliente.

- **Estilista:** Profissional que faz recomendações personalizadas.
- **Técnico/Funcionário:** Responsável pela operação do sistema.
- **Loja/Marca:** Responsável por disponibilizar items aos clientes.
- **Histórico de Encomendas:** Contém informações sobre todas as encomendas realizadas pelo cliente.
- **Histórico de Notificações:** Contém informações sobre todas as notificações recebidas pelo cliente.

As interações entre essas entidades permitem que o sistema ofereça os serviços de aluguel de roupas e recomendações personalizadas aos clientes.

4.2. Diagrama de Use Cases

O Diagrama de Use Cases ilustra as interações entre os atores (como clientes, técnicos/funcionários e estilistas) e o PackMyBag. Este diagrama destaca as principais funcionalidades oferecidas pela aplicação, como o aluguer de roupa, gestão de favoritos, sugestões personalizadas, notificações sobre encomendas e estado de disponibilidade, entre outros. Serve como base para identificar e organizar os requisitos funcionais do sistema, fornecendo uma visão clara dos processos suportados.

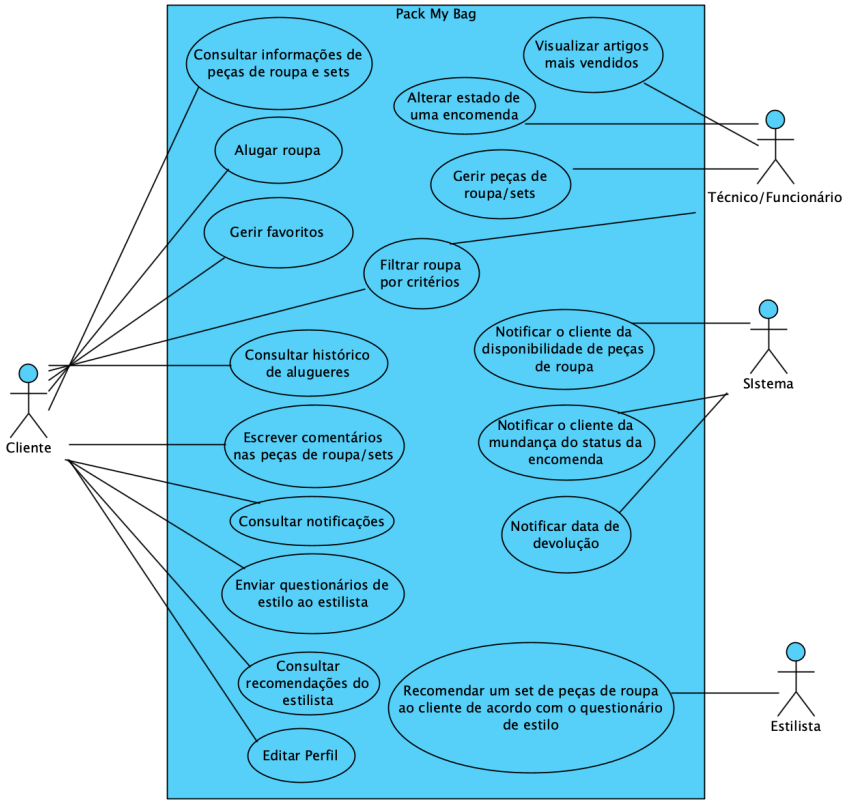

Figura 2: Diagrama de Use Cases

Conforme descrito na Figura 2, as principais funcionalidades do lado do cliente incluem:

- Consultar informações sobre peças de roupa e conjuntos disponíveis
- Alugar roupas
- Gerenciar sua coleção de favoritos
- Consultar o histórico de aluguéis
- Escrever comentários sobre as peças alugadas
- Consultar notificações do sistema
- Enviar questionários de estilo para o Estilista
- Editar seu perfil

Do lado do Técnico/Funcionário, as principais ações incluem:

- Alterar o estado de uma encomenda
- Gerir o stock de roupas de uma loja.
- Visualizar artigos mais vendidos.

Já o Estilista é responsável por:

- Recomendar um conjunto de peças de roupa de acordo com o questionário de estilo do Cliente

Adicionalmente, o sistema PackMyBag desempenha um papel importante, sendo responsável por:

- Notificar o cliente sobre a disponibilidade de peças de roupa
- Notificar o cliente sobre mudanças no status da encomenda
- Notificar a data de devolução

Ou seja, o sistema monitora alterações de estado e proativamente informa o usuário sobre essas atualizações relevantes.

4.3. Diagrama de Componentes

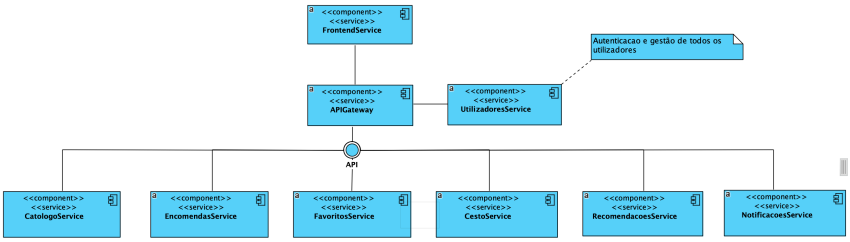

Figura 3: Diagrama de Componentes

O diagrama de componentes da aplicação Pack My Bag ilustra a sua arquitetura baseada em microsserviços. Esta abordagem modular oferece vantagens significativas em termos de escalabilidade, flexibilidade, e manutenção.
Esta arquitetura permitiu que cada componente fosse desenvolvido, implementado e escalado independentemente. Isto resultou numa maior agilidade de desenvolvimento, permitindo futuras atualizações mais rápidas e eficientes sem afetar o funcionamento de outras partes do sistema. Adicionalmente, ao utilizar esta arquitetura o isolamento de falhas é aprimorado, pois uma falha num único microsserviço não compromete a aplicação inteira. Finalmente, a organização em microsserviços permitiu que equipas menores trabalhassem em paralelo, melhorando a produtividade.
Os microsserviços representados no diagrama interagem de forma coordenada para proporcionar a experiência completa ao utilizador. Para melhor esclarecer o funcionamento, vamos descrever brevemente a função de cada microsserviço:

- **Frontend:** Responsável pela interface de utilizador (UI) e pela interação com o utilizador. Trata das requisições do lado cliente e renderiza a interface.
- **APIGateway:** Atua como um ponto de entrada único para todas as requisições, roteando-as para o microsserviço apropriado. Fornece segurança e gestão de tráfego.
- **UtilizadoresService:** Gerencia os dados dos utilizadores, incluindo registo, autenticação, perfil e gestão de preferências.
- **CatalogoService:** Responsável pela gestão do catálogo de roupas, incluindo pesquisa, filtragem, e detalhe dos produtos.

- **EncomendasService:** Gerencia as encomendas dos utilizadores, do processo de criação até à entrega.
- **NotificaçõesService:** Envia notificações aos utilizadores sobre o estado das encomendas, disponibilidade de roupas e outras informações relevantes.
- **RecomendaçõesService:** Fornece recomendações personalizadas de roupas aos utilizadores com base em um inquérito preenchido e enviado ao estilista.
- **FavoritosService:** Gerencia os itens de favoritos guardados pelo cliente.
- **CartService:** Responsável pela funcionalidade do carrinho de compras do cliente. Isto inclui adicionar itens ao carrinho, atualizar quantidades, remover itens e calcular o custo total dos itens no carrinho. Prepara os dados do carrinho para a criação da encomenda.

Esta abordagem modular torna a aplicação PackMyBag mais resiliente, escalável e fácil de manter, atendendo melhor às necessidades em constante evolução dos clientes.

5. Frontend

O frontend da aplicação foi desenvolvido utilizando o framework *Vue.js*, que permite a criação de interfaces de utilizador interativas e responsivas. A escolha do Vue.js deveu-se à sua facilidade de integração, curva de aprendizado amigável e performance eficiente. Com ele, construímos componentes reutilizáveis que facilitam a manutenção e evolução da interface. Além disso, a utilização do Vue possibilita a navegação dinâmica entre as diferentes vistas da aplicação. O frontend comunica-se com a API Gateway através de requisições HTTP assíncronas (estas permitem que um cliente faça uma solicitação e continue executando outras tarefas enquanto espera pela respota do servidor), consumindo assim os serviços disponibilizados pelos microserviços. Esta arquitetura proporciona uma separação clara entre a interface do utilizador e a lógica de negócio, aumentando a modularidade e escalabilidade do sistema.

6. API-Gateway

A API Gateway foi implementada em *Express.js*, servindo como ponto central para a comunicação entre o frontend e os microserviços. A função principal da API Gateway é rotear as requisições para o microserviço adequado, além de agregar, filtrar ou transformar as respostas quando necessário. Esta camada facilita a gestão de autenticação, autorização e outras políticas de segurança. A escolha pelo Express.js se deve-se à sua capacidade de lidar com um grande número de requisições simultâneas de forma não bloqueante, graças ao seu modelo de eventos assíncronos. Concluindo, a API Gateway simplifica a interação do frontend com a arquitetura de microserviços, ocultando a complexidade e proporcionando uma interface unificada. Ela comunica-se com os microserviços, que foram desenvolvidos em Spring Boot, garantindo a integração suave entre as diferentes tecnologias utilizadas no sistema.

7. Microserviço do Catálogo

7.1. Diagrama de Classes do Catálogo

7.2. Diagrama ER do Catálogo

7.3. API do Catalogo

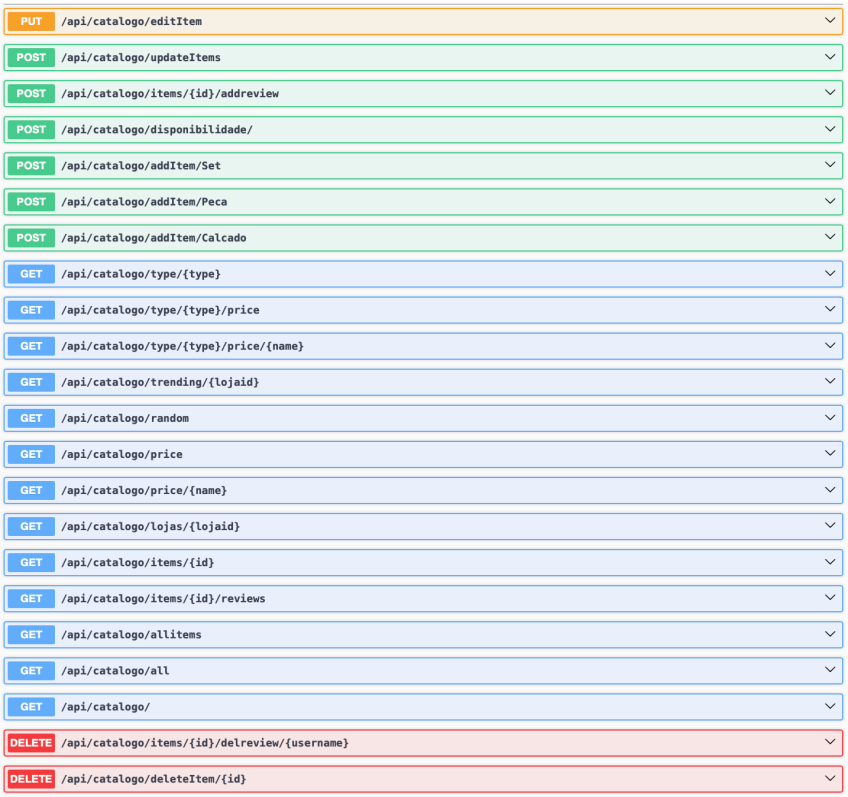

Figura 4: API do microserviço do catálogo

8. Microserviço do Cesto

O microserviço do cesto, construído sobre a tecnologia *Java Spring Boot* para o backend e *Vue.js* para o frontend, é responsável por gerir o processo de seleção e compra de artigos pelos clientes do nosso sistema. As principais funcionalidades deste microserviço incluem:

- **Exibir o cesto do cliente**: Os clientes podem visualizar o seu cesto, tal que contém todos os artigos que este pretende comprar.
- **Detalhar os produtos do cesto**: Ao visualizar o cesto, os clientes podem ver detalhadamente quais peças de roupa que colocaram no cesto, incluindo informações como o preço, a quantidade de cada item colocado e o total custo do cesto.
- **Fornecer informações sobre a encomenda**: Após a confirmação dos conteúdos do cesto, os clientes fornecem detalhes sobre a entrega, tal como o local desta e a duração da sua viagem.
- **Escolha do método de pagamento**: Após confirmar que todos os detalhes sobre a encomenda e o custo que irá ficar aos clientes pelo aluguer dos artigos do cesto, os clientes podem escolher pagar tanto com cartão multibanco como por paypal.

- **Histórico de pagamentos**: Os clientes podem consultar o histórico completo dos pagamentos passados, permitindo que eles acessem informações sobre pagamentos anteriores e completar pagamentos em curso.

8.1. Diagrama de Classes do Cesto

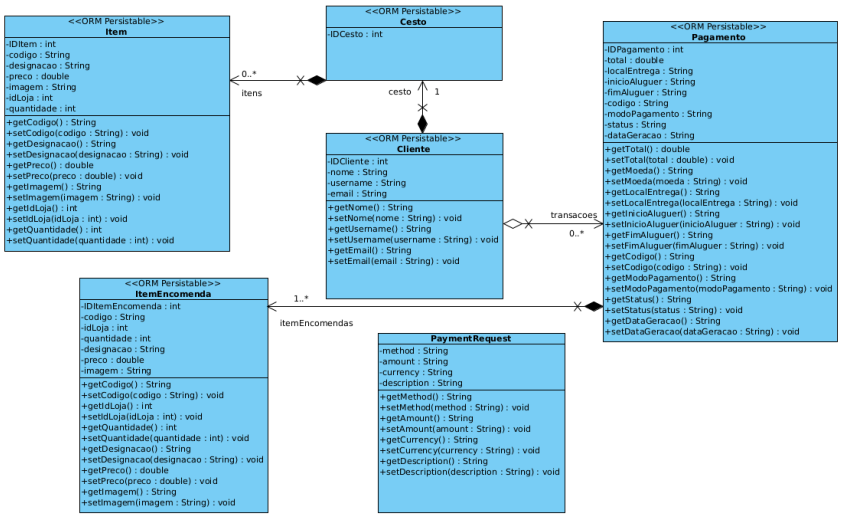

Figura 5: Diagrama de Classes do Microserviço do Cesto

O diagrama de classes mostra as principais entidades e seus relacionamentos:

- **Cliente**: Representa o utilizador que realiza as encomendas.
- **Cesto**: Contém todos os artigos que o cliente pretende alugar.
- **Item**: Cada produto que compõe um cesto.
- **Pagamento**: Informações sobre o cesto e a entrega.
- **ItemEncomenda**: Representa os artigos de um cesto para serem usados para criar uma encomenda.
- **PaymentRequest**: Usada para criar os pagamentos com o serviço do *PayPal*.

Estes elementos trabalham em conjunto para permitir que os clientes possam alugar os artigos na nossa aplicação e que o serviço de encomenda possa gerar as encomendas para os clientes.

8.2. Diagrama ER do Cesto

Figura 6: Diagrama ER do Microserviço do Cesto

O diagrama ER acima mostra a estrutura de dados subjacente ao microserviço do cesto:

- A entidade Cliente possui informações como nome, utilizador e e-mail.
- A entidade Pagamento armazena os detalhes do pagamento, como modo de pagamento e status, e detalhes para criação de uma encomenda, como local de entrega, data de geração, etc.
- A entidade Item representa cada produto que compõe um cesto.
- A entidade ItemEncomenda contém informações sobre cada item alugado pelo cliente.

As relações entre estas entidades permitem que o sistema gerencie todo o processo de um cesto, desde a criação pelo cliente até a criação de uma encomenda e confirmação de pedido.
Este microserviço atende tanto às necessidades dos clientes, que podem visualizar o histórico dos seus pagamentos, quanto aos funcionários das lojas, que podem gerir novos pedidos feitos pelos clientes.

8.3. API do Microserviço do Cesto

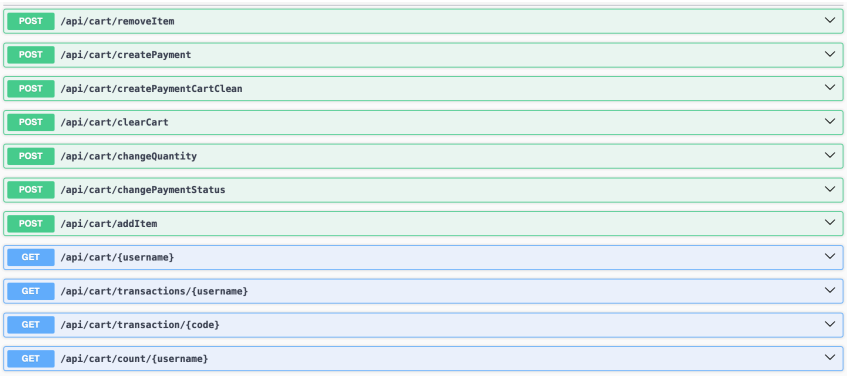

Figura 7: API do Microserviço do Cesto

As principais rotas da API do microserviço do Cesto são:

- **GET /api/cart/{username}**: Retorna todos os itens do cesto de um determinado cliente
- **GET /api/cart/transactions/{username}**: Retorna a lista de pagamentos de um determinado cliente
- **GET /api/cart/transaction/{code}**: Retorna um pagamento pelo id dele
- **GET /api/cart/count/{username}**: Retorna o número de artigos num cesto de um cliente
- **POST /api/cart/removeItem**: Remove um artigo de um cesto de um cliente
- **POST /api/cart/createPayment**: Cria um novo pagamento para o formulário dos estilistas
- **POST /api/cart/createPaymentCartClean**: Cria um novo pagamento, onde logo de seguida limpa o cesto do cliente
- **POST /api/cart/clearCart**: Limpa o carrinho de um cliente
- **POST /api/cart/changeQuantity**: Altera a quantidade de um item no cesto
- **POST /api/cart/changePaymentStatus**: Altera o estado de um pagamento
- **POST /api/cart/addItem**: Adiciona um item ao cesto de um cliente

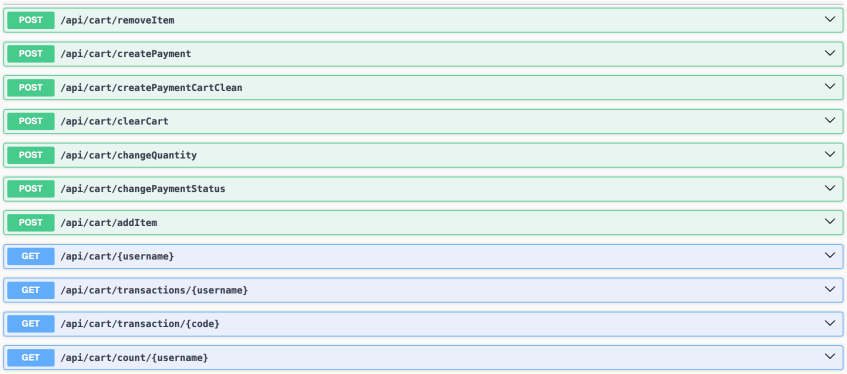

Figura 8: API do Microserviço do Cesto

Além disso, o microserviço também fornece endpoints para o pagamento através do serviço do paypal:

- **GET /api/cart/paypal/success**: Caso o pagamento seja aprovado executa o pagamento
- **GET /api/cart/paypal/cancel**: Cancela o pagamento
- **GET /api/cart/paypal/error**: Envia um erro sobre o pagamento
- **POST /api/cart/paypal/create**: Cria um novo pagamento

9. Microserviço de Encomendas

O microserviço de Encomendas desempenha um papel fundamental na aplicação *PackMyBag*, servindo tanto os clientes como os técnicos das lojas. Construído sobre a tecnologia *Java Spring Boot* para o backend e *Vue.js* para o frontend, este microserviço é responsável por gerir de forma abrangente o histórico de encomendas realizadas na plataforma. As principais funcionalidades deste microserviço incluem:

- **Exibir a lista de encomendas do cliente**: Os clientes podem visualizar todas as suas encomendas anteriores, acompanhando o status de cada uma delas.
- **Detalhar os produtos de cada encomenda**: Ao selecionar uma encomenda, os clientes podem ver detalhadamente quais peças de roupa foram alugadas, incluindo informações como o preço.
- **Acompanhar o status da encomenda**: Durante todo o ciclo de vida da encomenda, desde a criação até a devolução, os clientes podem acompanhar o status em tempo real, recebendo notificações sobre eventuais atualizações.
- **Fornecer informações sobre a entrega**: O microserviço armazena e exibe informações relevantes sobre a entrega, como data de entrega e devolução.
- **Histórico de encomendas anteriores**: Os clientes podem consultar o histórico completo das suas encomendas passadas, permitindo que eles acessem informações sobre peças alugadas anteriormente.

Além disso, este microserviço também desempenha um papel fundamental na gestão das encomendas pelas lojas parceiras. Os técnicos e funcionários de cada loja podem:

- **Visualizar a lista de encomendas da sua loja**: Os funcionários têm acesso à lista completa de encomendas realizadas para a sua loja, permitindo que eles acompanhem a demanda e o fluxo de atividades.
- **Alterar o status da encomenda**: Durante todo o processo, desde a separação dos itens até a devolução, os funcionários podem atualizar o status da encomenda, garantindo que os clientes sempre tenham informações precisas sobre o status dos seus pedidos.

Esta abrangência de funcionalidades posiciona o microserviço de Encomendas como um elemento-chave na arquitetura da aplicação *PackMyBag*. Além disso, a sua construção sobre tecnologias modernas, como *Java Spring Boot* e *Vue.js*, asseguram a escalabilidade e a flexibilidade necessárias para suportar eventuais atualizações e novas funcionalidades da plataforma no futuro.

9.1. Diagrama de Classes de Encomendas

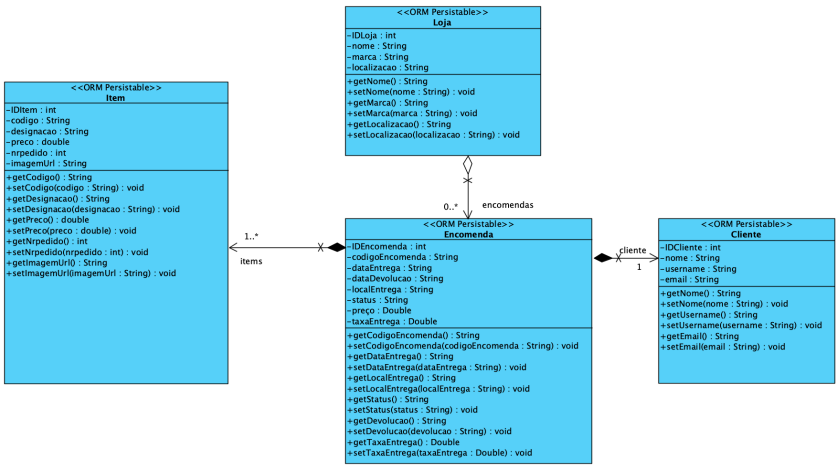

Figura 9: Diagrama de Classes do Microserviço de Encomendas

O diagrama de classes mostra as principais entidades e seus relacionamentos:

- **Cliente**: Representa o utilizador que realiza as encomendas.
- **Encomenda**: Registra os detalhes de cada encomenda realizada, como status, data de entrega, etc.
- **Item**: Cada produto que compõe uma encomenda.
- **Loja**: Informações sobre as lojas parceiras que fornecem as roupas para aluguel.
- **Tecnico/Funcionario**: Representa os funcionários das lojas que podem acessar e atualizar o status das encomendas.

Estes elementos trabalham em conjunto para permitir que os clientes realizem encomendas e os funcionários das lojas gerenciem o processo de entrega e devolução da encomenda.

9.2. Diagrama ER de Encomendas

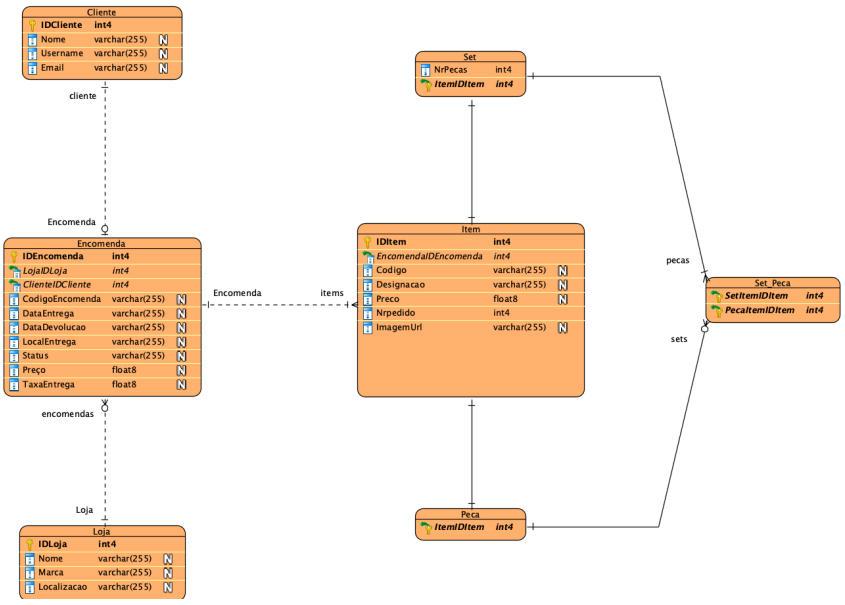

Figura 10: Diagrama ER do Microserviço de Encomendas

O diagrama ER mostra a estrutura de dados subjacente ao microserviço de Encomendas:

- A entidade Cliente possui informações como nome, utilizador e e-mail.
- A entidade Encomenda armazena os detalhes da encomenda, como status, data de entrega, etc.
- A entidade Item representa cada produto que compõe uma encomenda.
- A entidade Loja contém informações sobre as lojas parceiras, como nome, marca e localização.
- A entidade Tecnico/Funcionario regista os funcionários responsáveis por gerir as encomendas.

As relações entre estas entidades permitem que o sistema gerencie todo o ciclo de vida de uma encomenda, desde a criação pelo cliente até a entrega e devolução, com o acompanhamento dos funcionários das lojas.
Este microserviço atende tanto às necessidades dos clientes, que podem visualizar o histórico das suas encomendas, quanto aos funcionários das lojas, que podem gerir o status das encomendas durante todo o processo.

9.3. API do Microserviço de Encomendas

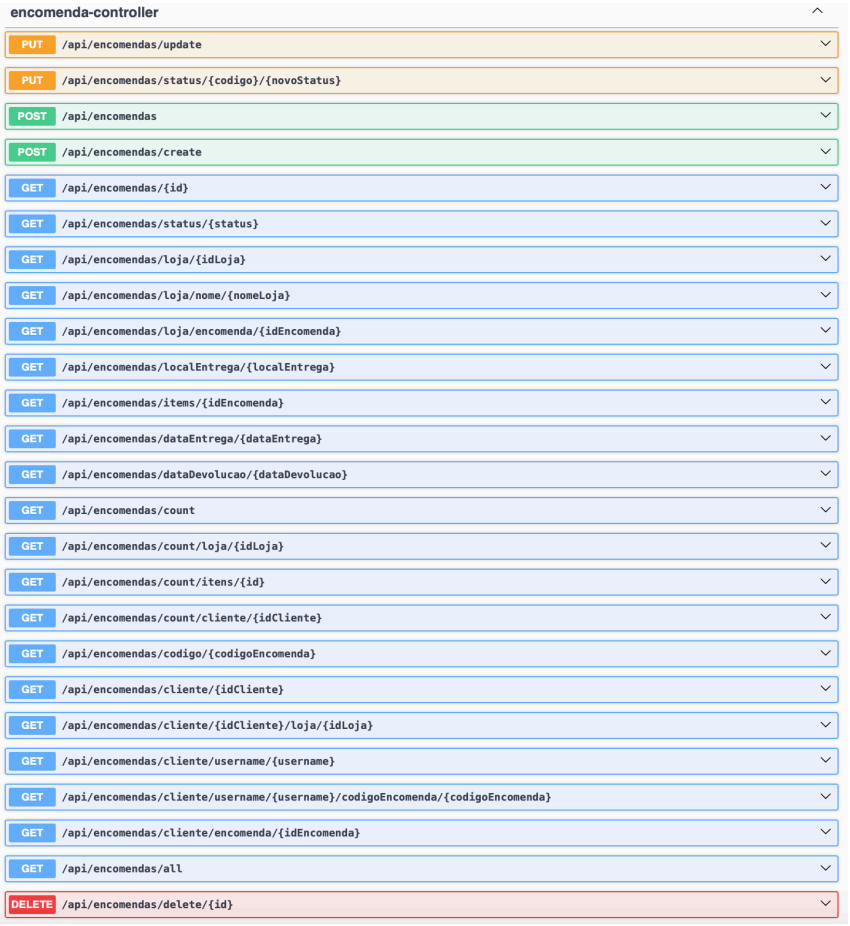

Figura 11: API do Microserviço de Encomendas

As principais rotas da API do microserviço de Encomendas são:

- **GET /api/encomendas/{id}**: Retorna os detalhes de uma encomenda específica
- **GET /api/encomendas/cliente/{idCliente}**: Retorna a lista de encomendas de um determinado cliente
- **GET /api/encomendas/loja/{idLoja}**: Retorna a lista de encomendas de uma determinada loja
- **GET /api/encomendas/dataEntrega/{dataEntrega}**: Retorna as encomendas com uma determinada data de entrega
- **GET /api/encomendas/dataDevoluacao/{dataDevoluacao}**: Retorna as encomendas com uma determinada data de devolução
- **POST /api/encomendas/create**: Cria uma nova encomenda
- **PUT /api/encomendas/update**: Atualiza os detalhes de uma encomenda
- **PUT /api/encomendas/status/{codigo}/{novoStatus}**: Atualiza o status de uma encomenda

Além disso, o microserviço também fornece endpoints para obter informações agregadas, como, por exemplo:

- **GET /api/encomendas/count**: Retorna o total de encomendas
- **GET /api/encomendas/count/loja/{idLoja}**: Retorna o total de encomendas de uma determinada loja
- **GET /api/encomendas/count/cliente/{idCliente}**: Retorna o total de encomendas de um determinado cliente

10. Microserviço dos Favoritos

O microserviço de itens favoritos gere uma das várias componentes de personalização da plataforma, sendo esta a possibilidade de guardar uma lista de artigos de roupa favoritos, ficando estes mais facilmente acessíveis para alugueres futuros. Semelhante a outros microserviços já explorados neste documento, teve o seu backend implementado em *Java Spring Boot* e o frontend em *Vue.js*. As principais funcionalidades deste microserviço incluem:

- **Exibir a lista de artigos favoritos do utilizador**: O utilizador pode aceder a uma página onde todos os itens que colocou como favorito se mantêm até que os elimine ou deixem de estar disponíveis na plataforma.
- **Filtrar a lista por tamanho, gênero e preço**: O utilizador escolhe um ou mais filtros que quando aplicados à lista, limitam o número de artigos expostos.
- **Adicionar um artigo aos favoritos**: O utilizador ao visualizar um artigo no catálogo, pode adicionar o artigo como favorito clicando no botão apropriado.
- **Remover um artigo aos favoritos**: O utilizador, na página dos favoritos, pode remover um artigo clicando no botão apropriado.

10.1. Diagrama de Classes dos Favoritos

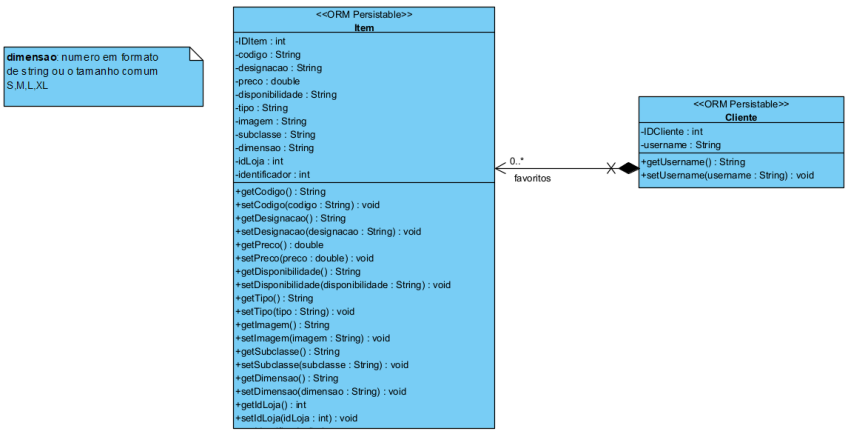

Figura 12: Diagrama de Classes do Microserviço de Favoritos

O diagrama de classes mostra as principais entidades e seus relacionamentos:

- **Cliente**: Representa o utilizador, numa forma mais simplificada, que está associado aos vários itens que considerou como favoritos.
- **Item**: Cada produto que constitui a lista dos favoritos.

Estes elementos trabalham em conjunto para permitir que os clientes possam consultar a lista dos artigos favoritos na nossa aplicação sem a ocorrência de conflitos a nível de dados.

10.2. Diagrama ER dos Favoritos

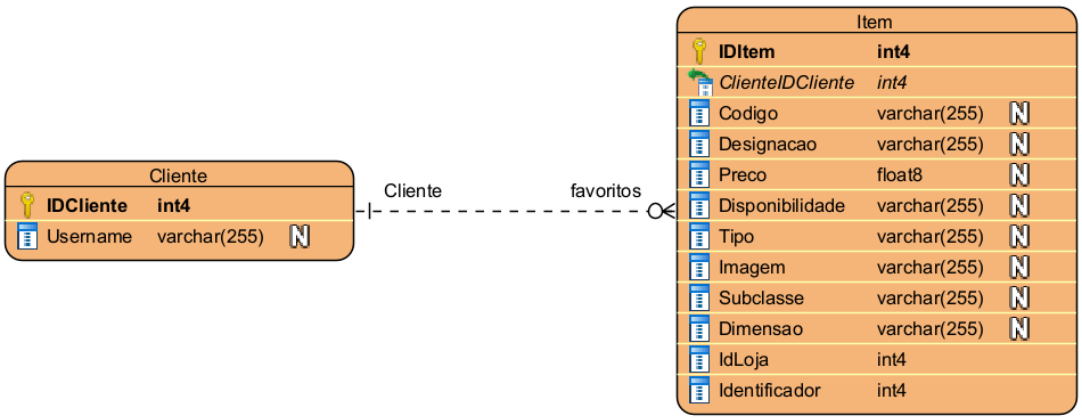

Figura 13: Diagrama ER do Microserviço de Favoritos

O diagrama ER acima mostra a estrutura de dados subjacente ao microserviço do cesto:

- A entidade Cliente possui informações como nome, utilizador e e-mail.
- A entidade Pagamento armazena os detalhes do pagamento, como modo de pagamento e status, e detalhes para criação de uma encomenda, como local de entrega, data de geração, etc.
- A entidade Item representa cada produto que compõe um cesto.
- A entidade ItemEncomenda contém informações sobre cada item alugado pelo cliente.

As relações entre estas entidades permitem que o sistema gerencie todo o processo de um cesto, desde a criação pelo cliente até a criação de uma encomenda e confirmação de pedido.
Este microserviço atende tanto às necessidades dos clientes, que podem visualizar o histórico dos seus pagamentos, quanto aos funcionários das lojas, que podem gerir novos pedidos feitos pelos clientes.

10.3. API dos Favoritos

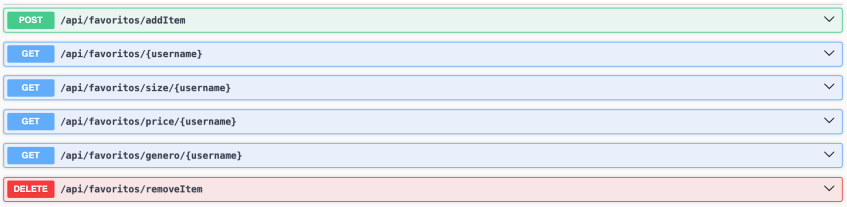

Figura 14: API do Microserviço dos Favoritos

-**POST /api/favoritos/addItem**:

-**GET /api/favoritos/{username}**:

-**GET /api/favoritos/size/{username}**:

-**GET /api/favoritos/price/{username}**:

-**GET /api/favoritos/genero/{username}**: Retorna os itens da lista de favoritos de um determinado utilizador do gênero selecionado.

-**GET /api/favoritos/removeItem**:

11. Microserviço de Notificações

O Microsserviço das Notificações é o serviço responsável por notificar o cliente sobre qualquer evento iminente, assim como completamento de certas ações, tais como pagamentos. Este foi sobre *Java Spring Boot* no backend, e Vue.js no frontend. Abrange apenas o cliente e consiste das seguintes funcionalidades:

- **Exibir um pop-up após certas ações**: após adicionar um item à lista dos interessados ou quando um pagamento é confirmado um pop-up é exibido a confirmar que a ação foi completada com sucesso;
- **Atualizar notificações quando houver mudança num item interessado**: sempre que houver uma alteração na disponibilidade de um item, o cliente que esteja interessado neste deve ser notificado;
- **Notificar cliente sobre a data de retorna**: após o pagamento de uma encomenda, uma notificação com a data de retorna deste é enviada ao cliente e uma outra é enviada 24 horas antes da dita data de retorna;
- **Exibir todas as notificações do cliente**: o cliente deve ter acesso a uma página em que pode visualizar todas as suas notificações;
- **Atualizar notificações quando houver mudança na encomenda**: o cliente deve ser notificado sempre que houver qualquer alteração inesperada a uma das suas encomenda.
- **Remover notificações da lista de notificações do cliente**: o cliente pode remover qualquer notificação da lista com todas as suas notificações;

11.1. Diagrama de Classes de Notificações

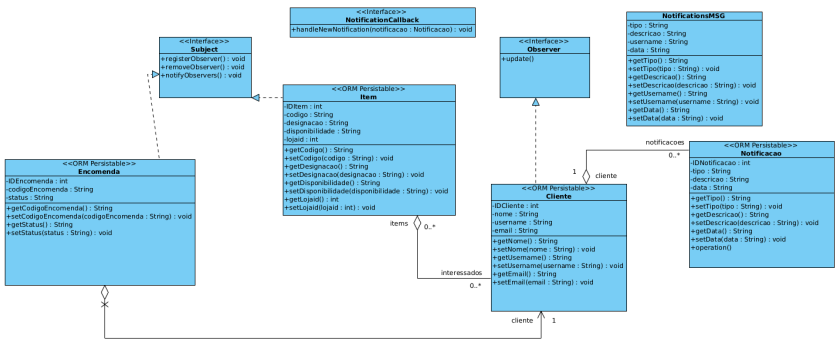

Figura 15: Diagrama de Classes do Microserviço de Notificações

O diagrama de classes mostra as principais entidades e seus relacionamentos:

- **Cliente**: Representa o utilizador que recebe as notificações.
- **Encomenda**: Registra os detalhes de cada encomenda realizada, como status, data de entrega, etc.
- **Item**: Cada produto que compõe uma encomenda.
- **Notificacao**: Contem o tipo e data de notificação, assim como a sua descrição e quaisquer outros dados pertinentes.

Os objetos que estarão a ser observados pelo cliente são:

- **encomenda**: que após pago irá atualizada em relação ao seu status, o qual, sempre que aconteça irá resultar numa notificação ao cliente.
- **item**: que poderá sofrer alterações em relação à sua disponibilidade, o qual será notificado ao cliente de cada vez que aconteça.

11.2. Diagrama ER de Notificações

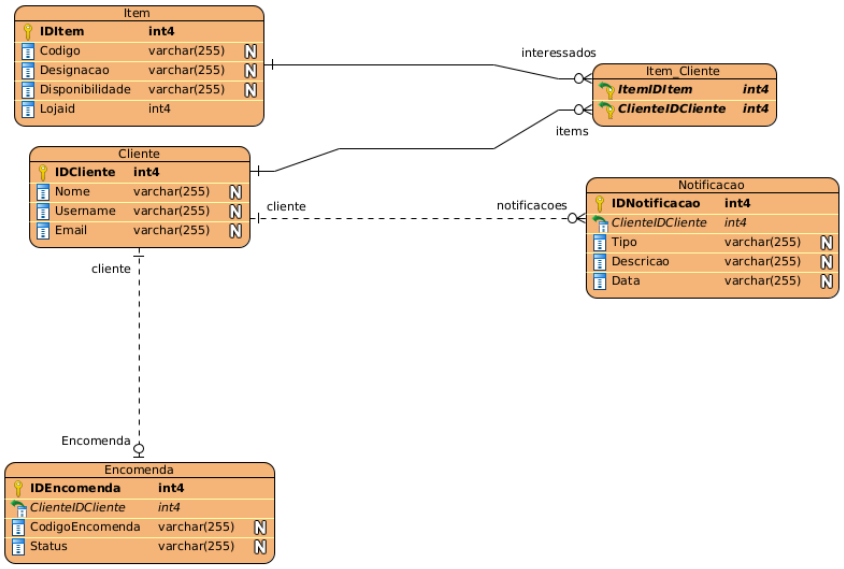

Figura 16: Diagrama ER do Microserviço de Notificações

O diagrama ER mostra a estrutura de dados subjacente ao microserviço de Notificações:

- A entidade Cliente possui informações como nome, utilizador e e-mail.
- A entidade Item representa cada produto que compõe uma encomenda.
- A entidade Encomenda armazena os detalhes da encomenda, como o status e id.
- A entidade Notificacao contém o tipo, data e descrição da notificação.

As relações entre estas entidades permitem que o sistema gerencie todo o ciclo de vida da notificação, desde a sua criação até à sua eliminação pelo cliente.
Este microsserviço atende somente ás necessidades dos clientes, notificando-o de toda a informação necessária sobre as suas encomendas, atualizações nos items em que está interessado, assim como servir de lembrete sobre o prazo para a devolução das encomendas após a sua utilização.

11.3. API do Microserviço de Notificações

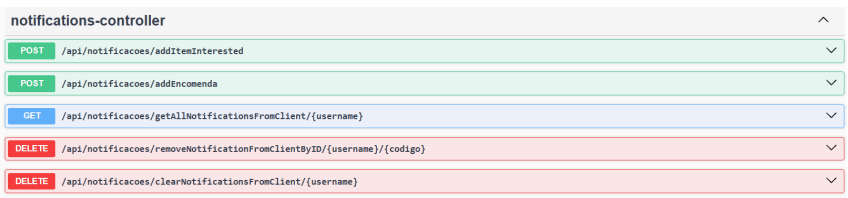

Figura 17: API do Microserviço de Notificações

As principais rotas da API do microserviço de Notificações são:

- **POST /api/notificacoes/addItemInterested**: adiciona um item à lista dos interessados do cliente.

- **POST /api/notificacoes/addEncomenda**: adiciona uma entrada de encomenda associada ao cliente.
- **GET /api/notificacoes/getAllNotificationsFromClient/{username}**: retorna ao cliente uma lista de todas as suas notificações.
- **DELETE /api/notificacoes/removeNotificationFromClientByID/{username}/{codigo}**: elimina uma notificação da lista de notificações do cliente.
- **DELETE /api/notificacoes/clearNotificationsFromClient/{username}**: elimina todas as notificações da lista de notificações do cliente.

12. Microserviço dos Utilizadores

O microserviço dos Utilizadores é um componente essencial da aplicação, responsável pela gestão e personalização de perfis do utilizador, bem como pela autenticação e registo de diferentes tipos de utilizadores. Este serviço, desenvolvido em Java Spring Boot, integra-se perfeitamente com outros microserviços da aplicação e lida com operações críticas relacionadas aos utilizadores, como atualização de dados, gestão de avatares e controlo de acessos.

12.1. Principais Funcionalidades

As principais responsabilidades deste microserviço incluem:

- **Autenticação (Login):** O microserviço trata do processo de login para todos os tipos de utilizadores, garantindo acesso seguro à aplicação através de credenciais válidas.
- **Registo (Signup) de Diferentes Tipos de Utilizadores:** Permite o registo de novos utilizadores nas seguintes categorias:
  - **Utilizador Normal:** Clientes que utilizam os serviços da aplicação.
  - **Estilista:** Profissionais responsáveis por recomendações de estilo ou serviços personalizados.
  - **Técnico:** Funcionários das lojas parceiras, encarregados de gerir operações relacionadas aos pedidos e logística.
- **Gestão de Perfis de Utilizador:** Os utilizadores podem visualizar e atualizar os seus dados pessoais, incluindo nome, email, password e outros detalhes relevantes.
- **Atualização do Avatar do Utilizador:** Implementa uma funcionalidade para que os utilizadores alterem a sua imagem de perfil, armazenando a nova imagem no servidor e associando-a ao utilizador.

Fluxo de Signup e Login

- **Signup:** O utilizador preenche um formulário com as informações necessárias (ex.: email, password, tipo de utilizador).Os dados são validados e armazenados na base de dados, garantindo que o utilizador seja registado com o tipo correto (normal, estilista ou técnico).
- **Login:** O utilizador fornece credenciais (email e password) para autenticação. O sistema verifica as credenciais e retorna um token de acesso para operações subsequentes.

Fluxo de Upload de Imagem

O utilizador envia uma imagem e o seu username através de um formulário. O backend recebe e valida os dados, armazenando a imagem num diretório. A imagem é associada ao utilizador na base de dados.

Fluxo de Atualização do Perfil

O microserviço de Utilizadores também permite aos utilizadores atualizar os dados do seu perfil, garantindo uma experiência personalizada e flexível. O fluxo para esta funcionalidade é o seguinte:

- **Solicitação de Atualização:** O utilizador envia uma requisição com os novos dados que deseja atualizar (ex.: nome, telemovel, etc). Esta solicitação é autenticada para garantir que apenas o próprio utilizador pode modificar os seus dados.
- **Validação de Dados:** O sistema verifica os dados recebidos, garantindo que estejam no formato correto e que as alterações sejam permitidas.
- **Atualização no Backend:** Os dados são enviados ao microserviço, onde as alterações são aplicadas na base de dados. Apenas os campos enviados são atualizados, mantendo os outros dados intactos.
- **Confirmação da Atualização:** Após o sucesso da operação, o microserviço retorna uma mensagem de confirmação ao utilizador, indicando que as informações foram atualizadas com sucesso.

13. Microserviço de Recomendações

O Microsserviço das Recomendações consiste num serviço de recomendações de outfits, e foi construído sobre a tecnologia *Java Spring Boot* no *backend*, e sobre o *framework Vue.js* no *frontend*. Este Microsserviço abrange tanto o cliente, como o estilista, e permite as seguintes funcionalidades:

- **Criação de um pedido de recomendação de outfit(s)**: O cliente poderá criar um pedido de recomendação de outfit(s) através do preenchimento e submissão de um *forms*. Apenas se tornará um pedido válido após a confirmação de pagamento;
- **Exibir a lista de pedidos de recomendação**: O estilista tem acesso a uma página com todos os pedidos de recomendação válidos que ainda não foram completados. Este terá acesso, em cada pedido, aos detalhes do *forms* preenchido pelo cliente;
- **Completar as recomendações dos pedidos correspondentes**: O estilista pode adicionar itens a cada recomendação, remover itens e criar uma descrição. A recomendação terá de ter obrigatoriamente itens e descrição para ele ter a possibilidade de completar o pedido;
- **Exibir a lista de recomendações completas**: O cliente tem acesso a uma página onde pode ver todas as recomendações que já recebeu, consoante os pedidos que efetuou;

13.0.1. Fluxo de realização do pedido

O utilizador preenche o formulário com os campos pedidos, sendo estes campos preferências pessoais em certos aspetos, número de *outfits*, *budget*. Quando este termina de preencher e submete, é associado um status de *pending*, e as informações são validadas e armazenadas na base de dados. Este estado tem uma duração máxima de 5 minutos, e após essa duração, se o pagamento não for confirmado, este pedido é invalidado e removido da base de dados. Caso seja pago dentro dos 5 minutos, o estado é alterado para pago, e o pedido é apresentado na página de pedidos e recomendações do estilista.

13.0.2. Fluxo de realização da recomendação

O pedido aparece na página de pedidos do estilista após o seu pagamento. Este irá poder adicionar e remover itens, utilizando para isso uma opção exclusiva para estilistas nas páginas dos itens no catálogo, que também permite ver um resumo das várias recomendações por completar através de um pop-up. Este também terá de escrever uma descrição para a recomendação. Depois de ter isto tudo este poderá submeter a recomendação, que será validada e ficará com o estado completo.

13.1. Diagrama de Classes do Microsserviço das Recomendações

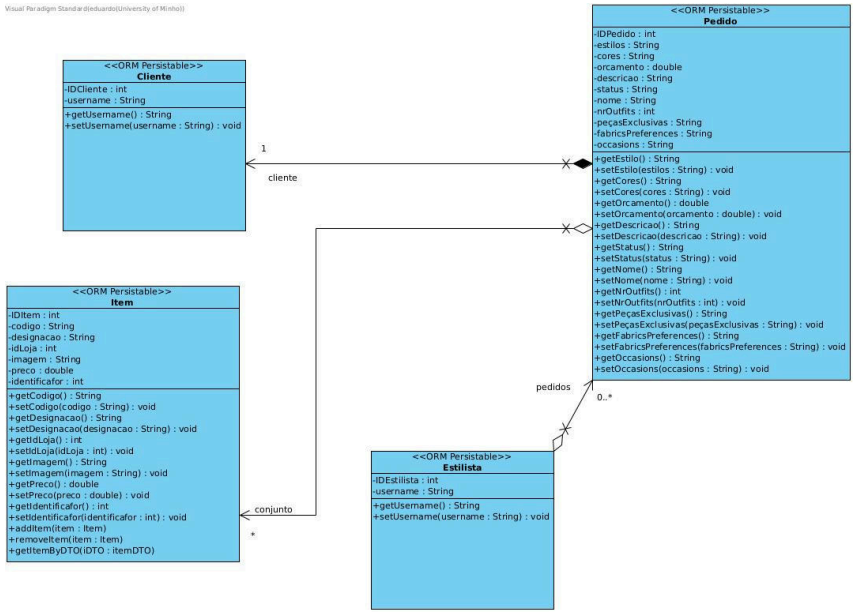

Figura 18: Diagrama de Classes do Microsserviço de Recomendações

O diagrama de classes representa as principais entidades deste Microsserviço, bem como as suas relações.

- **Cliente**: Representa o utilizador que realiza o pedido, e recebe as recomendações;
- **Estilista**: Representa o utilizador que completa as recomendações;
- **Item**: Representa as peças de roupa que são adicionadas a recomendações;
- **Pedido**: Regista os detalhes do pedido efetuado pelo Cliente, bem como o estado do pedido, e as informações da recomendação do estilista.

Estes elementos trabalham em conjunto para permitir que os clientes possam ter um apoio especializado na escolha de peças para aluguer, e os estilistas possam atender a esses pedidos de acordo com as exigências do cliente.

13.2. Diagrama ER do Microsserviço das Recomendações

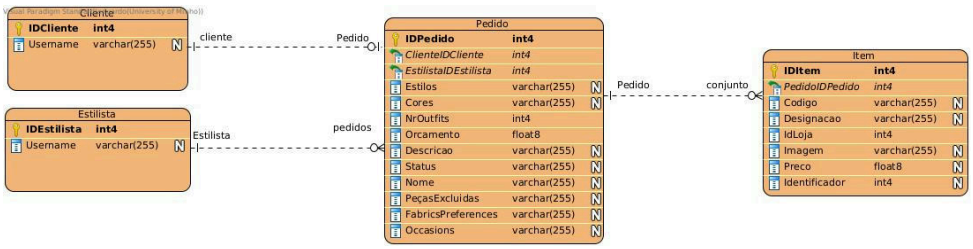

Figura 19: Diagrama ER do Microsserviço de Recomendações

O diagrama ER enfatiza a estrutura de dados subjacente ao microsserviço de Recomendações.

- A entidade *Cliente*, neste microsserviço, apenas é definida pelo *username*;
- A entidade *Estilista*, neste microsserviço, apenas é definida pelo *username*;
- A entidade *Item* é definida pelo seu código, id da loja, identificador no serviço do catálogo, imagem, preço e designação. Estes são necessários para o cliente poder adicionar os itens de uma recomendação completa ao carrinho;

- A entidade *Pedido* é definida pelo nome do pedido, e contém os dados do formulário preenchido pelo cliente, que contém informações como o número de *outfits*, o *budget*, as preferências de materiais, etc. Para além disso, possui os dados da recomendação do estilista, a descrição.

As relações entre estas entidades permitem que o sistema gerencie todo o processo da criação de um pedido, até ao pedido ser concluído e já possuir a recomendação do estilista.
Este microsserviço atende tanto ás necessidades dos clientes, que podem obter recomendações de *outfits* dos estilistas consoante as suas exigências, quanto aos estilistas que podem editar as recomendações enquanto não a sinalizam como completa.

13.3. API do Microsserviço das Recomendações

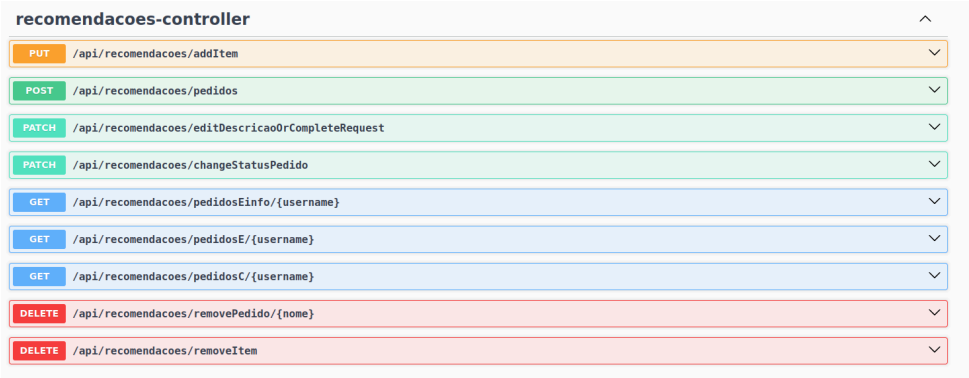

Figura 20: API do Microsserviço das Recomendações

As principais rotas da API deste serviço são:

- **GET /api/recomendacoes/pedidosE/{username}**: Retorna a um estilista a sua lista de pedidos/recomendações pagas e por completar;
- **GET /api/recomendacoes/pedidosEinfo/{username}**: Retorna a um estilista a sua lista de pedidos/recomendações pagas e por completar, com informações resumidas para o menu de adição de um item a uma recomendação nos itens do catálogo;
- **GET /api/recomendacoes/pedidosC/{username}**: Retorna a um cliente a sua lista de recomendações que já foram completadas pelo estilista;
- **POST /api/recomendacoes/pedidos**: Adiciona um pedido à estrutura de dados;
- **DELETE /api/recomendacoes/removePedido/{nome}**: Permite remover um pedido pendente que não foi dado como pago;
- **PUT /api/recomendacoes/addItem**: Adiciona um item ao conjunto de itens da recomendação de um pedido pago por completar;
- **DELETE /api/recomendacoes/removeItem**: Remove um item do conjunto de itens da recomendação de um pedido pago por completar;
- **PATCH /api/recomendacoes/editDescricaoOrCompleteRequest**: Permite ao estilista editar a descrição de uma recomendação e/ou completar um pedido;
- **PATCH /api/recomendacoes/changeStatusPedido**: Permite alterar o estado de um pedido, quando esta alteração é válida.

14. Conclusão

A realização deste projeto representou um percurso intenso de aprendizagem e superação de desafios. Desde o início, sabíamos que a implementação da aplicação PackMyBag exigiria a incorporação de tecnologias com as quais não estávamos totalmente familiarizados, como Spring Boot, Vue.js e Express.js. Esta necessidade impulsionou-nos a investir tempo e esforço significativos na aquisição de novos conhecimentos e competências técnicas.

Ao longo do processo, enfrentamos diversas dificuldades inerentes à adoção de uma arquitetura baseada em microsserviços. Tivemos de compreender profundamente como esses componentes interagem, garantindo a comunicação eficaz entre si e mantendo a integridade e a performance da aplicação. A integração destes microsserviços através da API Gateway, bem como a implementação de um frontend responsivo e intuitivo com Vue.js, foram etapas desafiadoras que ampliaram consideravelmente a nossa experiência em desenvolvimento web.

Paralelamente, o levantamento de requisitos junto dos potenciais utilizadores forneceu insights valiosos que orientaram o desenvolvimento das funcionalidades da plataforma. Foi essencial assegurar que a nossa solução fosse verdadeiramente alinhada com as necessidades e expectativas do público-alvo, oferecendo uma experiência de utilizador enriquecedora e diferenciada.
Apesar do tempo limitado, dedicámo-nos intensamente para construir uma plataforma web robusta, funcional e escalável. O resultado é uma aplicação que não só atende aos objetivos propostos mas também estabelece uma base sólida para futuras expansões e melhorias.
Em suma, acreditamos que a plataforma tem o potencial de inovar o mercado de aluguer de roupas em Portugal, proporcionando aos utilizadores uma solução conveniente, personalizada e eficiente para as suas necessidades durante viagens.

Bibliografia

[1] [Online]. Disponível em: https://swagger.io/

[2] [Online]. Disponível em: https://vuejs.org/

[3] [Online]. Disponível em: https://docs.spring.io/spring-boot/index.html

[4] [Online]. Disponível em: https://medium.com/@ozziefel/kafka-in-microservices-architecture--enabling-scalable-and-event-driven-systems-7ff474de49f4

[5] [Online]. Disponível em: https://docs.docker.com/compose/gettingstarted/