Universidade do Minho
Escola de Engenharia
Mestrado em Engenharia Informática

Requisitos e Arquiteturas de Software

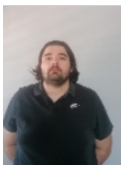
Ano Letivo de 2024/2025

PictuRAS

Eduardo Cunha
PG55939

Rodrigo Ralha
PG56005

Délio Alves
a94557

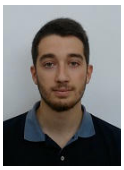


Rafael Peixoto
PG55998

Rui Gonçalves
PG56011

João Coelho
PG55954

Flávio Silva
PG57539



Filipe Oliveira
a80330

January 27, 2025

RAS

Índice

Lista de Figuras

1. Introdução

Este documento foi elaborado no contexto de uma proposta para desenvolver o sistema PictuRAS, uma plataforma de processamento e edição de imagens acessível a diferentes tipos de utilizadores. A principal motivação para a criação do PictuRAS reside na necessidade de uma ferramenta intuitiva e versátil, capaz de permitir tanto a amadores como a profissionais explorar um vasto conjunto de funcionalidades de edição de imagens de forma eficiente e acessível, suportada por uma arquitetura moderna e escalável na cloud.

No documento, são descritas as fases do planeamento para a implementação do sistema, conforme definido no relatório de arquitetura selecionado pela equipa docente. Para além disso, a nossa equipa considerou e definiu alguns componentes adicionais, que estão descritos neste relatório, bem como as tecnologias que optámos por utilizar. Apresenta-se a arquitetura definida, juntamente com as pequenas alterações que foram aplicadas, e descreve-se o processo de implementação da mesma. A implementação do sistema foi concluída e encontra-se detalhadamente descrita neste relatório, que documenta o processo de desenvolvimento do PictuRAS.

Por fim, é apresentado um levantamento completo dos requisitos cumpridos e não cumpridos, comparando-os com os requisitos iniciais, de forma a avaliar o grau de execução e a abrangência da solução.

2. Requisitos

2.1. Requisitos Funcionais

Inicialmente, optámos por enumerar os requisitos funcionais estabelecidos na segunda fase:

Requisito	Descrição	*Use Case*	Prioridade
Req1	O utilizador autentica-se utilizando as suas credenciais.	2	Must
Req2	O utilizador anónimo regista-se.	1	Must
Req3 e Req4	O utilizador escolhe o plano de subscrição (Gratuito ou *Premium*).	1	Must
Req6	O utilizador cria um projeto.	3	Must
Req7	O utilizador lista os seus projetos.	3	Must
Req8	O utilizador acede à área de edição de um projeto.	3	Must
Req9	O utilizador carrega imagens para um projeto.	3	Must
Req10	O utilizador remove uma imagem do projeto.	4	Must
Req11	O utilizador adiciona uma ferramenta de edição ao projeto.	4	Must
Req12	O utilizador desencadeia o processamento de um projeto.	4	Must
Req13	O utilizador transfere o resultado de um projeto para o dispositivo local	4	Must
Req14	O utilizador altera a ordem das imagens de um projeto.	4	Could
Req15	O utilizador altera a ordem das ferramentas de um projeto.	4	Must
Req16	O utilizador altera os parâmetros das ferramentas.	4	Must
Req17	O utilizador cancela o processamento de um projeto durante a sua execução.	4	Should
Req18	O utilizador registado acede às suas informações de perfil.	1	Should
Req19	O utilizador edita o seu perfil.	1	Should

Req20	O utilizador com subscrição Premium altera a recorrência de pagamento entre mensal ou anual.	1	Could
Req21	O utilizador *premium* cancela a sua subscrição *premium*.	1	Should
Req22	O utilizador registado termina a sua sessão.	2	Must
Req23	O utilizador registado remove a sua conta e dados.	1	Must
Req24	O utilizador partilha o resultado da edição de uma imagem diretamente nas redes sociais.	4	Must
Req25	O utilizador recorta manualmente imagens.	4	Must
Req26	O utilizador escala imagens para dimensões específicas.	4	Must
Req27	O utilizador adiciona borda a imagens.	4	Must
Req28	O utilizador altera a saturação a imagens.	4	Must
Req29	O utilizador ajusta o brilho a imagens.	4	Must
Req30	O utilizador ajusta o contraste a imagens.	4	Must
Req31	O utilizador binariza imagens.	4	Should
Req32	O utilizador roda imagens.	4	Must
Req33	O utilizador aplica um algoritmo de recorte automático de imagens com base no seu conteúdo.	4	Must
Req34	O utilizador aplica ajustes automáticos de otimização das imagens com base no conteúdo.	4	Could
Req35	O utilizador remove o fundo da imagem, mantendo apenas o objeto principal.	4	Must
Req36	O utilizador extrai texto de imagens.	4	Should
Req37	O utilizador aplica um algoritmo de reconhecimento de objetos em imagens.	4	Must
Req38	O utilizador aplica um algoritmo de contagem de pessoas em imagens.	4	Should

De seguida, apresentamos o diagrama de casos de uso, que auxilia na compreensão das ações que cada tipo de utilizador pode realizar.

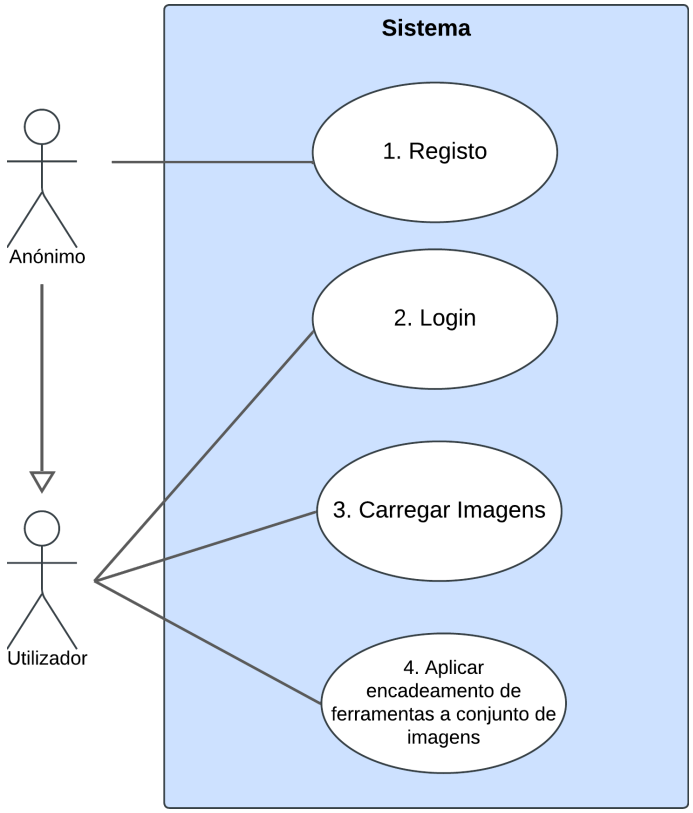


Figura 1: Diagrama de *use cases* do PictuRAS

De realçar que os utilizadores premium, como mencionado nos requisitos, têm maior liberdade no uso total da aplicação em comparação com um utilizador normal, uma vez que essa distinção não está especificada no diagrama.

2.2. Requisitos Não Funcionais

Decidimos expôr os requisitos não funcionais também:

Requisito	Descrição	Prioridade
Req1	A aplicação deve conter uma interface simples e clara.	Must
Req2	A aplicação deve conter ícones e botões que sejam claros e intuitivos.	Must
Req3	A aplicação deve utilizar cores suaves e neutras.	Could
Req4	A aplicação deve ser fácil de usar.	Must
Req5	A página de um projeto distingue visualmente entre as ferramentas básicas e avançadas.	Must
Req6	O utilizador vê todas as funcionalidades da aplicação mesmo que o seu perfil não lhes confira acesso.	Must

Requisito	Descrição	Prioridade
Req7	O utilizador cria um projeto implicitamente ao arrastar ficheiros para o dashboard da aplicação.	Must
Req8	O utilizador carrega imagens arquivadas num único ficheiro .zip.	Must
Req9	A listagem dos projetos é ordenada pela data do último acesso: mais recentes primeiro.	Should
Req10	A listagem dos projetos é pesquisável por nome.	Could
Req11	O utilizador é mantido informado acerca do estado de processamento de um projeto em tempo real.	Must
Req12	A aplicação fornece ajudas contextuais em várias páginas da aplicação, oferecendo informações guiadoras para o fluxo e a experiência do utilizador.	Could
Req13	A aplicação apresenta a pré-visualização do resultado da aplicação da sequência de ferramentas do projeto na imagem atualmente selecionada.	Must
Req14	A visualização de imagens grandes ou pequenas é auxiliada pelo utiliário zoom	Must
Req15	A visualização de imagens permite que se navegue em todas as duas direções arrastando o rato.	Must
Req16	A aplicação ajuda o utilizador a selecionar apenas ferramentas compatíveis.	Should
Req17	A aplicação fornece tutoriais ou dicas integradas.	Could
Req18	A aplicação é compatível com diferentes plataformas e browsers, incluindo de dispositivos móveis e desktop.	Must
Req19	A aplicação fornece feedback visual claro e imediato ao utilizador em caso de erro ou falha durante um procedimento demorado.	Should
Req20	O utilizador copia um link que aponta para a página do projeto.	Should
Req21	As ferramentas de edição de imagem devem ser processadas rapidamente.	Should
Req22	O sistema deve processar até 100 imagens ao mesmo tempo, sem quebras percetíveis no desempenho.	Must
Req23	A aplicação deve ser capaz de escalar horizontalmente de forma elástica para suportar o aumento de utilizadores e volume de processamentos, mantendo o desempenho mas também os custos controlados.	Must
Req24	A aplicação deve estar disponível 24 horas por dia, todos os dias.	Should

Requisito	Descrição	Prioridade
Req25	A aplicação deve ser integrável com outras plataformas e serviços de terceiros.	Should
Req26	A aplicação deve otimizar o uso de recursos computacionais do cliente (e.g., CPU, memória)	Should
Req27	A aplicação deve suportar os principais formatos de imagem, como JPEG, PNG, BMP e TIFF.	Should
Req28	A aplicação deve ser facilmente estendida com novas ferramentas de edição.	Must
Req29	A manutenção programada deve originar, no máximo, 10 minutos de downtime por mês, sendo realizada fora dos horários de maior utilização.	Should
Req30	O suporte ao cliente deve estar disponível por chat ou email, com tempos de resposta rápidos.	Should
Req31	As imagens devem ser cifradas durante o envio e quando são armazenadas no sistema.	Should
Req32	O sistema deve ser projetado para facilitar a execução de testes.	Must
Req33	A aplicação deve realizar backups automáticos dos dados e imagens dos utilizadores.	Should
Req34	O sistema garante que apenas os utilizadores que carregam as imagens têm acesso a elas.	Must
Req35	O sistema deve usar MFA para aumentar a segurança dos utilizadores registados.	Could
Req36	O sistema deve garantir que apenas utilizadores com permissões adequadas acedem a funcionalidades restritas, aplicando políticas de controlo de acesso claras e seguras.	Should
Req37	As sessões dos utilizadores devem ter um logout automático após longos períodos de inatividade.	Could
Req38	O sistema deve cumprir o RGPD.	Should
Req39	A aplicação deve estar disponível em vários idiomas.	Should
Req40	A aplicação deve ajustar-se aos diferentes formatos de data e hora.	Should
Req41	Os Termos e Condições da aplicação devem ser claros e acessíveis.	Should

2.3. Requisito Extra

Na fase anterior do projeto, surgiu a ideia de incluir um requisito não funcional, pelo que decidimos também implementá-lo. Segue o requisito:

Requisito #: RNF42	Tipo: Não funcional	Use cases #: —
Descrição	**Oferecer periodo de experimentção "trial" gratuito apenas a utilizadores sem histórico de trial ou plano premium.**	
Rationale	Aumentar conversões permitindo testes gratuitos para novos utilizadores elegíveis.	
Origem	*Engenheiro de Software*	
Fit criterion	Garantir que apenas utilizadores sem histórico de trial ou premium possam aceder ao trial.	
Prioridade	*Must*	
Data	2024/10/03	

Requisito 1: Trial

2.4. Restrições ao Desenvolvimento da Aplicação

Segue-se as restrições definidas para o desenvolvimento do projeto:

2.4.1. Restrições Técnicas

1. A aplicação deve ser exclusivamente desenvolvida como uma aplicação *web*.
2. A aplicação deverá estar preparada para ser disponibilizada numa infraestrutura Cloud (AWS, Azure, Google Cloud), considerando aspetos como escalabilidade, segurança e otimização de custos.
3. É obrigatório garantir que a plataforma implemente medidas robustas de segurança, incluindo a proteção contra ataques comuns (e.g., SQL injection, XSS) e políticas adequadas de gestão de dados dos utilizadores, conforme regulamentos de privacidade como o RGPD.
4. A arquitetura da aplicação deve ser elástica para suportar múltiplos utilizadores simultâneos, e o desempenho do sistema deve ser otimizado para garantir uma experiência fluida, independentemente do número de utilizadores ativos ou da carga de processamento.

2.4.2. Restrições de Negócio

1. Dependendo do tipo de utilizador existem limites associados ao número de processamentos diários, de imagens num projeto, ao tipo de ferramentas que podem ser utilizadas e à dimensão das imagens carregadas.
2. A entrega da implementação da aplicação deve ser realizada até 17 de janeiro de 2025, incluindo a versão final do *software* com todas as funcionalidades principais implementadas.

3. Contexto e Âmbito do Sistema

Segue o diagrama de contexto técnico, que é claro e suficiente para apresentar as mesmas informações de um diagrama de contexto tradicional. O diagrama evidência as interações do sistema com serviços como email, pagamento e armazenamento, além de identificar os serviços externos que poderiam ser utilizados: o Stripe, como plataforma de pagamento; o S3, como serviço de armazenamento; e o AWS SES, como serviço de email. ***Contudo, optámos por utilizar o Mailhog para agilizar no processo de desenvolvimento, sendo que o mesmo pode ser configurado em produção para encaminhas para o SES.***

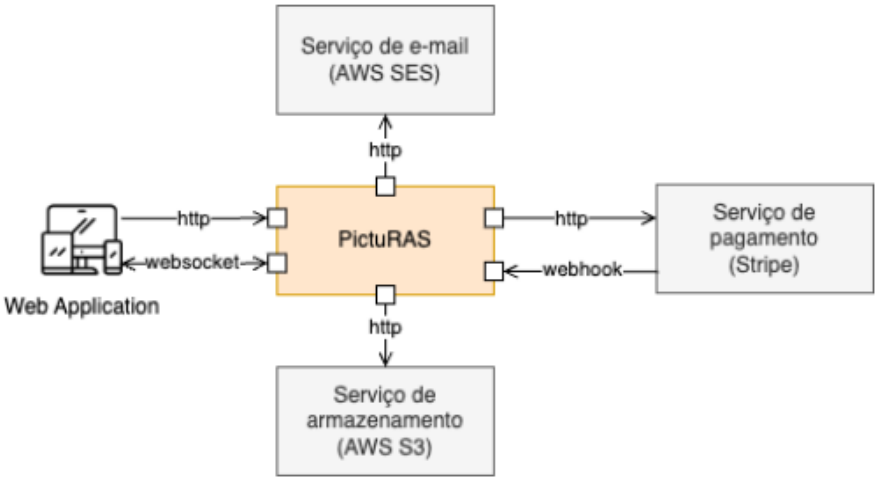


Figura 2: Diagrama de contexto técnico do PictuRAS

4. Estratégia da Solução

4.1. Decomposição Funcional

Segue a decomposição funcional do sistema, representada num diagrama que evidência a divisão das responsabilidades do sistema em diferentes módulos. Esta abordagem permite identificar subsistemas de forma clara, facilitando o desenvolvimento da solução final.

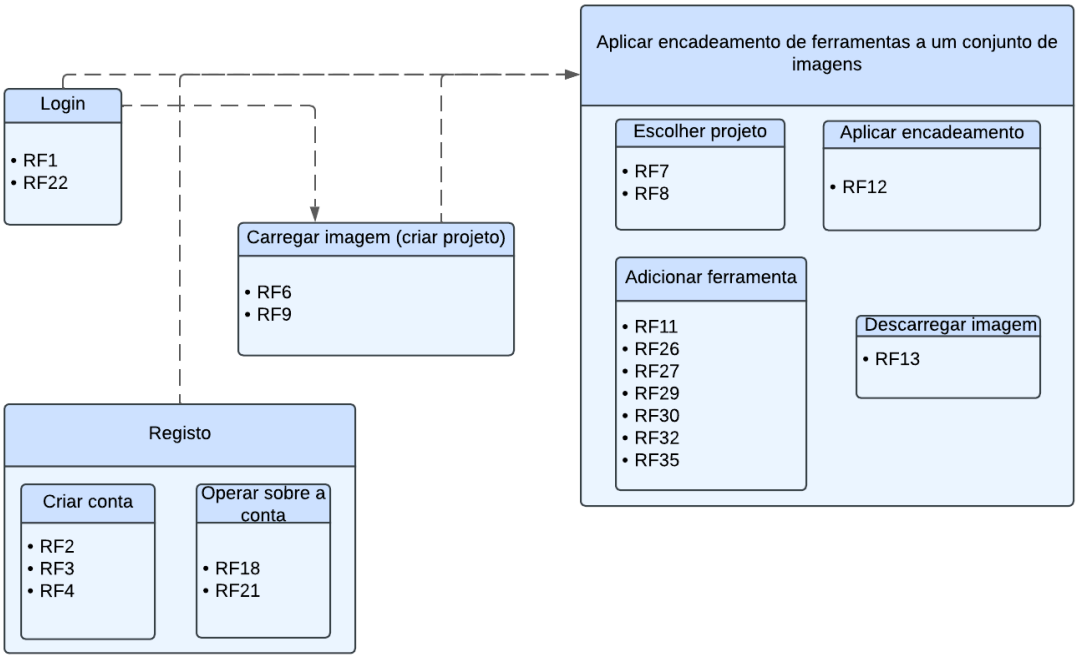


Figura 3: Diagrama de decomposição funcional do PictuRAS

Com base no diagrama apresentado, verifica-se que grande parte da complexidade do sistema está no encadeamento de ferramentas, que constitui o núcleo das interações e dependências entre os subsistemas. Este encadeamento requer a integração de diversos sub-módulos, exigindo um esforço adicional da equipa na implementação destas funcionalidades.

4.2. Padrão Arquitetural

A arquitetura de micros-serviços organiza uma aplicação em serviços independentes, cada um responsável por uma função específica. Cada serviço opera autonomamente, com o seu próprio ciclo de desenvolvimento e escalabilidade, comunicando através de APIs. Esta abordagem oferece elevada modularidade, permitindo o desenvolvimento, atualização ou substituição de serviços de forma isolada, sem afetar os outros. A escalabilidade também é favorecida, pois permite a expansão individual de serviços conforme necessário.

A solução final adota uma arquitetura de **Microsserviços**, incorporando elementos de uma arquitetura **Event-driven**, com o uso de um *message broker*. Assim, combina-se a modularidade dos microsserviços com a reatividade de uma arquitetura orientada a eventos.

5. Especificações da Arquitetura

Neste capítulo, apresentam-se as especificações da arquitetura.

5.1. Identificação de Microsserviços

Aqui está descrito a divisão dos microserviços da arquitetura:

Microsserviço de Subscrições
Este microsserviço será responsável pela gestão das subscrições e respetivos pagamentos dos utilizadores, garantindo que todos os processos decorrem de forma conforme e assegurando a satisfação dos utilizadores.

Microsserviço de Utilizadores
Em muitas arquiteturas de microsserviços, existe um componente denominado Gateway, que encaminha os pedidos dos clientes para os microsserviços apropriados.

Neste caso, a gestão de utilizadores é dividida entre duas componentes: um microsserviço dedicado à criação, edição e remoção de utilizadores, e uma camada de autenticação e autorização no Gateway. Esta abordagem visa minimizar a carga no microsserviço de utilizadores, evitando que a autenticação dependa exclusivamente deste serviço.

Microsserviço para cada uma das ferramentas
Cada ferramenta do sistema é gerida por um microsserviço específico. Esta abordagem facilita a escalabilidade individual de cada ferramenta, conforme a demanda dos utilizadores.

Além disso, permite o desenvolvimento paralelo de cada ferramenta por equipas especializadas, podendo utilizar tecnologias distintas. A extensão do sistema com novas ferramentas torna-se também mais simples, necessitando apenas da criação e configuração de novos microsserviços.

Microsserviço de Projetos
Este microsserviço será responsável pela gestão dos projetos dos utilizadores. Desde a criação, edição e remoção até ao manuseamento e armazenamento das imagens.

5.2. Comunicação entre Microsserviços

A decomposição funcional apresentada foca-se em garantir um elevado nível de desacoplamento entre os vários microsserviços, com a comunicação entre eles, na sua maioria, limitada à dinâmica de aplicação das ferramentas de edição. Para garantir características como escalabilidade, performance e redundância, esta comunicação deve ser realizada através de um sistema assíncrono, baseado na troca de mensagens (event-driven).

Neste contexto, há dois eventos que importa destacar:

1. Pedido para aplicação de uma ferramenta
Este evento é utilizado para solicitar a execução de uma transformação por parte de um microsserviço. O pedido inclui informações sobre o tipo de transformação a realizar (por exemplo, rotação, redimensionamento, contagem de pessoas) e os parâmetros necessários para a sua execução. O evento é enviado para a fila apropriada no RabbitMQ, onde será processado pelo microsserviço responsável.

Exemplo para pedido de rotação de imagem:

```
{
  "messageId": "request-2",
  "timestamp": "2024-11-01T12:00:00Z",
  "procedure": "rotate",
  "parameters": {
    "inputImageURI": "file:///images/request-1-out.jpg",
    "outputImageURI": "file:///images/request-2-out.jpg",
    "degrees": -90
  }
}
```


2. Conclusão da aplicação de uma ferramenta

Este evento é gerado quando um microsserviço responsável termina uma transformação, indicando se o processo foi concluído com sucesso ou se ocorreu um erro. O evento deve incluir o estado geral do processamento, informações sobre o ficheiro ou resultado gerado (em caso de sucesso) ou detalhes sobre o erro (em caso de falha), além de um identificador que permita associar a resposta ao pedido original. Este evento é utilizado para informar os sistemas interessados sobre o desfecho da operação.

Exemplo, em caso de sucesso:


```
{
  "messageId": "completion-2",
  "correlationId": "request-2",
  "timestamp": "2024-11-01T12:00:01Z",
  "status": "success",
  "output": {
    "type": "image",
    "imageURI": "file:///images/request-2-out.jpg"
  },
  "metadata": {
    "processingTime": 0.2,
    "microservice": "picturas-rotate-tool-ms"
  }
}
```


5.2.1. Message Broker

Relativamente à troca de pedidos utilizando o messager broker, apresenta-se abaixo o *schema* de pedido para a aplicação de um filtro (rotação, neste caso concreto).


```
{
  "rotation": {
    "isPremium": false,
    "schema": {
      "$ref": "#/definitions/rotation",
      "definitions": {
        "rotation": {
          "type": "object",
          "properties": {
            "angle": {
              "type": "number",
              "minimum": -180,
              "maximum": 180
            }
          },
```

```
                "required": [
                  "angle"
                ],
                "additionalProperties": false
              }
            },
            "$schema": "http://json-schema.org/draft-07/schema#"
          }
        }
}
```


5.3. Sistema de Dados

Segue-se o diagrama ER fornecido como auxílio para a criação da base de dados. Este diagrama oferece uma visão geral de como as diferentes tabelas de dados do sistema estão inter-relacionadas.

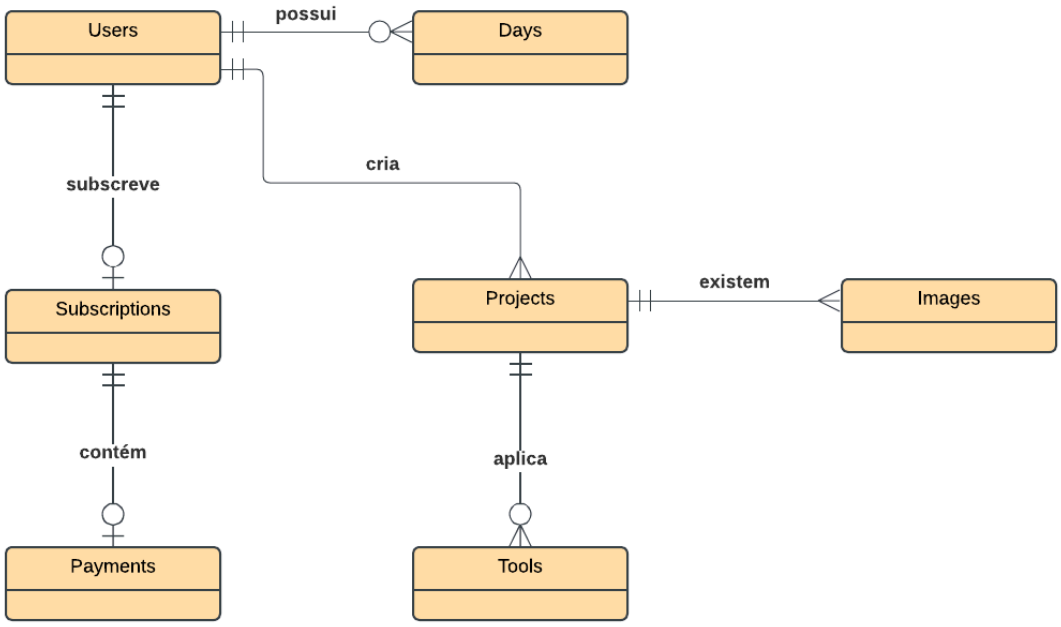


Figura 4: Diagrama ER

A seguir, apresenta-se uma lista dos modelos de dados efetivamente implementados para cada um dos microsserviços mencionados, os quais nem sempre seguiram as recomendações do diagrama. Cada modelo é acompanhado de um exemplo prático.

Vale ressaltar que estamos a utilizar collections do MongoDB, o que implica uma abordagem diferente da usada em bases de dados relacionais. Em MongoDB, não trabalhamos com tabelas, mas com coleções de documentos que podem conter dados estruturados e semiestruturados. Assim, as relações entre as entidades são geridas de maneira diferente, adaptando-se à flexibilidade do modelo de dados não relacional.

5.3.1. Subscrições

No microsserviço de subscrições, ao contrário do que é referido no diagrama, foi criada apenas uma coleção que centraliza os dados associados às subscrições dos utilizadores. Esta abordagem simplifica a estrutura da base de dados ao concentrar toda a informação relevante numa única entidade.

Subscriptions		
Campo	**Tipo**	**Descrição**
userId	`ObjectId`	Identificador único do utilizador associado à subscrição (obrigatório)
premium	`Boolean`	Indica se a subscrição é premium (*default*: `false`)
plan	`String`	Plano da subscrição: regular, mensal ou anual (*default*: `regular`)
trialUsed	`Boolean`	Indica se o período de teste foi utilizado (*default*: `false`)
stripeId	`String`	Identificador único gerado pela Stripe (obrigatório, índice único)
subscriptionId	`String`	Identificador opcional da subscrição, *default*: `null`

Tabela 6: Modelo de dados de uma subscrição

Campo	Valor
userId	64a0f1c6e3b5b47d12345678
premium	`true`
plan	`yearly`
trialUsed	`false`
stripeId	sub_1234567890abcdef
subscriptionId	sub_abcdef1234567890

Tabela 7: Exemplo de uma entrada da tabela das subscrições

5.3.2. Utilizadores

Igualmente neste exemplo optamos pela definição de apenas uma coleção de base de dados.

Users		
Campo	**Tipo**	**Descrição**
name	`String`	Nome do utilizador (obrigatório)
email	`String`	Email do utilizador (obrigatório, único, com índice)
password	`String`	Palavra-passe do utilizador (obrigatória)
username	`String`	Nome de utilizador (obrigatório, único, com índice)
profilePic	`String`	URL da imagem de perfil do utilizador (*default*: `null`)
location	`String`	Localização do utilizador (opcional)
bio	`String`	Descrição pessoal ou biografia (opcional)
refresh	`String`	Token de atualização (opcional)
active	`Boolean`	Indica se a conta está ativa (*default*: `false`)
otpSecret	`String`	Segredo do OTP para autenticação de dois fatores (opcional)
otpEnabled	`Boolean`	Indica se a autenticação de dois fatores está ativada (*default*: `false`)
emailPreferences	`Object`	Preferências de email: atualizações de projeto, marketing, etc. (*default*: todos `true`)
expireAt	`Date`	Data de expiração do utilizador (*default*: data de criação, expira em 24h)

Tabela 8: Modelo de dados de um utilizador

Campo	Valor
name	Cristiano Ronaldo
email	cris.georgina@example.com
password	$2b$10$eBj2jNdW1bDj1YKzXvQ9rOFGsAaGz4Z6jQo6f.n/EqU4QR9b1vKZW
username	CR7
profilePic	https://example.com/profiles/manUnitedRonaldo.jpg
location	Meca, Arábia Saudita
bio	À procura de conquistar um mundial com a seleção.
refresh	fgr4Tyr9Kz8PlU6
active	`true`
otpSecret	ABCD1234EFGH5678
otpEnabled	`true`
emailPreferences	projectUpdates: `true`, newFeatures: `true`, marketing: `false`, projectCollaborations: `true`, comments: `true`
expireAt	`2025-01-18T12:00:00Z`

Tabela 9: Exemplo de uma entrada da tabela dos utilizadores

5.3.3. Projetos

Este microserviço implementa quatro coleções na base de dados:

- **Imagens:** Armazena informações sobre as imagens e os seus formatos, com expiração em 48 horas.
- **Limites:** Regista os limites de execução de pipelines por utilizador, com expiração em 48 horas.
- **Pipelines:** Contém dados sobre os pipelines executados, com expiração após 24 horas.
- **Projetos:** Guarda os projetos, incluindo as ferramentas aplicadas e as imagens associadas, com expiração em 48 horas.

Estas coleções trabalham em conjunto para gerir eficientemente os dados dos utilizadores e os seus projetos.

Image		
Campo	**Tipo**	**Descrição**
_id	`ObjectId`	Identificador único da imagem (referência à coleção `Image`)
url	`String`	URL da imagem (obrigatória)
ttl	`Date`	Data e hora de expiração da imagem (*default*: data atual, expira em 48h)

Tabela 10: Modelo de dados de uma imagem

Campo	Valor
_id	60d21b9667d0d8992e610c85
url	https://example.com/images/sample-image.jpg
ttl	`2025-01-19T12:00:00Z`

Tabela 11: Exemplo de uma entrada da tabela das imagens

Limits		
Campo	**Tipo**	**Descrição**
_id	`ObjectId`	Referência ao identificador único do utilizador (referência à coleção `userId`)
qtdRunnedPipelines	`Number`	Número de pipelines executados (obrigatório)
ttl	`Date`	Data e hora de expiração do limite (*default*: data atual, expira em 48h)

Tabela 12: Modelo de dados de limites de utilizador

Campo	Valor
_id	60d21b9667d0d8992e610c85
qtdRunnedPipelines	5
ttl	`2025-01-19T12:00:00Z`

Tabela 13: Exemplo de uma entrada da tabela dos limites de utilizador

Pipeline		
Campo	**Tipo**	**Descrição**
_id	`ObjectId`	Referência ao identificador único do utilizador (referência à coleção `userId`)
projects	`Array ObjectId`	Lista de projetos associados ao utilizador (referência à coleção `Project`)
ttl	`Date`	Data e hora de expiração do pipeline (*default*: data atual, expira em 24h)

Tabela 14: Modelo de dados de pipeline de utilizador

Campo	Valor
_id	60d21b9667d0d8992e610c85
projects	60d21b9667d0d8992e610c99, 60d21b9667d0d8992e610c97
ttl	`2025-01-18T12:00:00Z`

Tabela 15: Exemplo de uma entrada da tabela dos pipelines de utilizador

Project		
Campo	**Tipo**	**Descrição**
userId	`ObjectId`	Identificador único do utilizador (referência à coleção `User`)
name	`String`	Nome do projeto (obrigatório)
tools	`Array Object`	Lista de ferramentas aplicadas ao projeto, contendo o nome do filtro e os argumentos associados
images	`Array Object`	Lista de imagens associadas ao projeto, incluindo o identificador da imagem e o formato
result	`Object`	Resultado do projeto, incluindo a data de expiração e o output gerado
ttl	`Date`	Data de expiração do projeto (*default*: null, expira em 48h)

Tabela 16: Modelo de dados de um projeto

Campo	Valor
userId	60d21b9667d0d8992e610c85
name	Example Project
tools	{filterName: “resize”, args: {width: 800, height: 600}} {filterName: “rotate”, args: {angle: 90}}
images	{id: 60d21b9667d0d8992e610c86, format: “png”} {id: 60d21b9667d0d8992e610c87, format: “jpg”}
result	expireDate: “2025-01-20T12:00:00Z”, output: “Processed image output”
ttl	`2025-01-19T12:00:00Z`

Tabela 17: Exemplo de uma entrada da tabela dos projetos

5.4. Rotas da API

5.4.1. Microsserviço de Subscrições

GET /plans
Esta rota é responsável por devolver uma lista de todos os planos de subscrição disponíveis. Os dados incluem preços, intervalos (mensal ou anual) e identificadores únicos configurados no Stripe. É utilizada para exibir as opções de subscrição disponíveis aos utilizadores.

POST /webhook
Esta rota processa notificações enviadas pelo Stripe sobre eventos relacionados com as subscrições e pagamentos. Três eventos principais são tratados: *invoice.payment_succeeded*, que atualiza o estado do utilizador para premium e regista o plano ativo; *invoice.payment_failed*, que remove o estado premium do utilizador; e *customer.subscription.deleted*, que desativa a subscrição e reverte o estado do utilizador para uma conta regular.

GET /
Esta rota devolve informações sobre a subscrição do utilizador autenticado. Ela verifica na base de dados se o utilizador possui um plano ativo, indicando se ele é premium, se o período de teste já foi utilizado e qual é o plano atual. Caso o utilizador não tenha subscrição, devolve informações padrão de conta regular.

POST /subscribe
Esta rota permite que o utilizador inicie ou atualize uma subscrição. O sistema verifica se o utilizador já possui uma subscrição ativa e, caso não tenha, cria um cliente no Stripe e inicia o processo de subscrição. A resposta inclui informações para concluir o pagamento ou a configuração, como um clientSecret.

DELETE /cancelSubscription
Esta rota cancela a subscrição ativa do utilizador. Após verificar se há uma subscrição em andamento, ela cancela a subscrição no Stripe e atualiza a base de dados para refletir que o utilizador não possui mais um plano ativo.

GET /transactionHistory
Esta rota devolve o histórico de transações do utilizador autenticado. São listadas as faturas associadas ao cliente no Stripe, incluindo detalhes como valores pagos, moeda, estado e data. É útil para exibir o histórico de pagamentos ao utilizador.

GET /billingInfo
Esta rota fornece informações de cobrança relacionadas ao utilizador autenticado, como os últimos quatro dígitos do cartão utilizado e a bandeira do cartão. Caso não existam métodos de pagamento salvos, devolve um objeto vazio.

5.4.2. Microsserviço de Utilizadores

POST /register
Esta rota é responsável pelo registo de novos utilizadores na aplicação. Ela recebe informações como nome, e-mail, palavra-passe e nome de utilizador, valida esses dados conforme o esquema definido, e armazena o novo utilizador na base de dados após criptografar a palavra-passe com o algoritmo bcrypt. Além disso, é gerado um token de validação que é enviado para o e-mail do utilizador para ativação da conta.

POST /register/2
Esta rota conclui o processo de registo do utilizador, verificando o token de validação recebido por e-mail. Caso o token seja válido e não tenha expirado, a conta do utilizador é ativada.

POST /login
Permite que um utilizador faça login na aplicação. Após validar as credenciais fornecidas (e-mail e palavra--passe), verifica se a conta está ativa e devolve um token de validação temporário, além de indicar se o utilizador tem autenticação em dois fatores (OTP) habilitada.

POST /login/2
Completa o processo de autenticação em duas etapas. A rota verifica o token de validação e, caso a autenticação OTP esteja ativada, valida o código de autenticação fornecido pelo utilizador. Em seguida, são gerados tokens de acesso e de atualização (JWT) para uso subsequente.

POST /guestLogin
Permite que utilizadores anónimos tenham uma access token que os identifique nos demais serviços.

POST /passwordRecovery
Inicia o processo de recuperação de palavra-passe. O sistema verifica se o e-mail fornecido está associado a uma conta ativa e, se válido, envia um token de recuperação para o e-mail do utilizador.

POST /passwordRecovery/2
Permite que o utilizador redefina a palavra-passe após validar o token de recuperação recebido no e-mail. A nova palavra-passe é criptografada e salva na base de dados.

POST /token
Usada para renovar o token de acesso quando ele expira. O token de atualização é validado e, se for legítimo, um novo token de acesso é emitido.

POST /logout
Usada para fazer o logout do utilizador da aplicação.

PUT /profilePic
Permite que o utilizador atualize a sua foto de perfil. A imagem enviada é processada, armazenada no MinIO, e o link da nova imagem é salvo na base de dados.

GET /
Devolve os detalhes do perfil do utilizador atualmente autenticado, incluindo informações como nome de utilizador, e-mail, localização, biografia e nome.

PUT /
Permite que o utilizador atualize informações pessoais como nome, localização ou biografia. Apenas os campos fornecidos na requisição são atualizados.

PUT /password
Permite que o utilizador altere a sua palavra-passe. A nova palavra-passe é criptografada antes de ser salva na base de dados.

POST /otp
Ativa a autenticação em duas etapas (OTP) para a conta do utilizador. Um secret é gerado e associado à conta do utilizador, permitindo que ele use uma aplicação autenticadora para gerar códigos temporários.

DELETE /otp
Desativa a autenticação em duas etapas (OTP) para a conta do utilizador. Remove o secret associado e desativa a funcionalidade na base de dados.

PUT /updateEmailPreferences
Permite que o utilizador atualize as suas preferências de notificação por e-mail, como atualizações de projeto, novos recursos e marketing. As alterações são salvas na base de dados.

DELETE /deleteAccount
Apaga todas as informações relativas a um utilizador bem como todos os seus projetos e subscrições.

5.4.3. Microsserviço de Projetos

5.4.3.1. projectRoutes.js

POST /
Esta rota é responsável pela criação de um novo projeto. Ela recebe os dados do projeto no corpo da requisição, como nome e outros atributos, e os valida. Após a validação, o projeto é adicionado à base de dados. Se o processo for bem-sucedido, o projeto criado será retornado com o estado HTTP 200. Caso contrário, será retornado um erro 500 com a mensagem "Failed to add project".

GET /:id
Esta rota permite recuperar os detalhes de um projeto específico, identificado pelo seu ID. Se o projeto for encontrado, ele será retornado com o estado HTTP 200. Caso contrário, será devolvido um erro 404 com a mensagem "Project not found".

GET /:id/output
Esta rota recupera a saída de um projeto específico, identificando-o pelo seu ID. Se o projeto ou a saída não forem encontrados, será devolvido um erro 404 com a mensagem "Project not found". Caso contrário, a saída do projeto será retornada com o estado HTTP 200.

GET /
Esta rota retorna uma lista de todos os projetos do utilizador, podendo incluir filtros de consulta. Caso os projetos sejam encontrados, serão retornados com o estado HTTP 200. Caso contrário, será devolvido um erro 404 com a mensagem "Projects not found".

PUT /:id
Esta rota permite atualizar os dados de um projeto específico. O id do projeto é extraído da URL e o corpo da requisição contém as informações a serem atualizadas. Se o projeto for encontrado e atualizado com sucesso, ele será retornado com o estado HTTP 200. Caso contrário, será devolvido um erro 404 com a mensagem "Project not found".

DELETE /:id
Esta rota permite excluir um projeto específico. O id do projeto é extraído da URL e, caso o projeto seja encontrado e removido com sucesso, será devolvido um estado HTTP 200 com o projeto excluído. Caso contrário, será devolvido um erro 404 com a mensagem "Project not found".

5.4.3.1.1. Ferramentas (Tools)

POST /:id/tool
Esta rota permite adicionar uma ferramenta a um projeto específico, identificando-o pelo seu id. A ferramenta é validada antes de ser adicionada ao projeto. Se a ferramenta for premium e o utilizador não for premium, será retornado um erro 401. Caso contrário, a ferramenta é adicionada com sucesso e é retornado um estado HTTP 200. Este enpoint retorna apenas o índice da ferramenta na lista de ferramentas.

DELETE /:id/tool/:idxTool
Esta rota permite remover uma ferramenta de um projeto específico, identificando o projeto pelo id e a ferramenta pelo índice idxTool. Se a remoção for bem-sucedida, será retornado um estado HTTP 200 com o projeto atualizado.

PUT /:id/tool/:idxTool
Esta rota permite reordenar as ferramentas dentro de um projeto. O id do projeto e o índice da ferramenta (idxTool) são passados na URL, enquanto o corpo da requisição contém o novo índice da ferramenta (idxToolAfter). Se a operação for bem-sucedida, o projeto atualizado será retornado com o estado HTTP 200.

5.4.3.1.2. Imagens (Images)

GET /:id/image/:idxImage
Esta rota permite recuperar uma imagem específica de um projeto, identificando o projeto pelo seu id e a imagem pelo índice idxImage. Se a imagem for encontrada, será retornada com o estado HTTP 200. Caso contrário, será devolvido um erro 404.

POST /:id/image
Esta rota permite adicionar uma nova imagem a um projeto específico. A imagem é enviada no corpo da requisição como um arquivo, e será validada antes de ser armazenada. Se a imagem for adicionada com sucesso, será retornado um estado HTTP 200 com o índice da imagem e a URL pública da imagem.

PUT /:id/image/:idxImage
Esta rota permite reordenar as imagens dentro de um projeto. O id do projeto e o índice da imagem (idxImage) são passados na URL, enquanto o corpo da requisição contém o novo índice da imagem (idxImageAfter). Se a operação for bem-sucedida, a imagem será movida para o novo índice e retornada com o estado HTTP 200.

DELETE /:id/image/:idxImage
Esta rota permite remover uma imagem de um projeto específico. O id do projeto e o índice da imagem (idxImage) são passados na URL. Se a remoção for bem-sucedida, será retornado um estado HTTP 200 com a imagem removida.

5.4.3.1.3. Processamento de um Projeto/Pipeline

POST /:id/process
Esta rota inicia o processamento de um projeto. O id do projeto é passado na URL, e a operação verifica se o utilizador atingiu o limite diário de processamento. Se o limite não for atingido, o projeto será adicionado à pipeline e o processamento será iniciado. Será retornado um estado HTTP 200 com a mensagem "Pipeline processing started."

POST /:id/preview
Esta rota permite gerar uma pré-visualização de um projeto. O id do projeto é passado na URL, e o corpo da requisição contém o imageId da imagem a ser processada. Se o processamento da pré-visualização for iniciado com sucesso, será retornado um estado HTTP 200 com a mensagem "Pipeline processing started."

DELETE /:id/process
Esta rota permite cancelar o processamento de um projeto. O id do projeto é passado na URL e, se o cancelamento for bem-sucedido, será retornado um estado HTTP 200 com a mensagem "Pipeline processing canceled."

5.4.4. Microsserviço para cada uma das ferramentas

Embora existam micros-serviços dedicados às ferramentas, as rotas para interagir com elas são geridas dentro das rotas dos Projetos, como foi descrito anteriormente. Isto significa que, embora a execução de uma ferramenta ocorra de forma independente no seu próprio micro-serviço, a gestão das ferramentas é realizada no contexto do projeto ao qual pertencem.

6. Tecnologias Aplicadas

Esta secção tem como objetivo descrever as tecnologias utilizadas pela equipa de trabalho na implementação das diversas componentes do sistema.

6.1. Frontend

O Vue.js foi a principal framework escolhida pela sua simplicidade, versatilidade e eficiência na criação de interfaces reativas. A introdução da Composition API proporcionou maior flexibilidade e organização à lógica dos componentes, permite que as funcionalidades fossem estruturadas de forma mais modular e reutilizável. Para complementar, usamos a liguagem TypeScript, uma extensão do JavaScript que adiciona tipagem estática ao código.

Vue Router foi utilizado para gestão de rotas, para permitir uma navegação fluida entre diferentes views da aplicação. Para design, recorremos ao Tailwind CSS, um framework utility-first que facilitou o desenvolvimento rápido e consistente da interface. Com o apoio do PostCSS, foi possível criar uma experiência visual moderna, responsiva e alinhada às necessidades do projeto.

Estrutura do Frontend

A estrutura do Frontend foi construido com o Vite, é uma ferramenta que se destina a oferecer uma experiência de desenvolvimento mais rápida e leve para projetos de web. Esta estrutura organizada e modular facilita o desenvolvimento, permite uma clara separação do projeto.

```
picturas_web
├── public/           Ficheiros estáticos (imagens do logo)
└── src/
    ├── api/          Contém toda a lógica de comunicação com o backend
    ├── assets/       Recursos estáticos
    ├── components/   Componentes reutilizáveis
    ├── router/       Configuração de rotas
    ├── stores/       Gestão do estado global
    ├── views/        Páginas da aplicação
    ├── App.vue       Componente raiz da aplicação
    └── main.ts       Ponto de inicialização da aplicação
```

Interação com o API Gateway

A comunicação entre o frontend e o backend foi implementada foi realizada via HTTP com o API Gateway, com a utilização da biblioteca Axios. Com esta abordagem, foi possível estabelecer uma interação eficiente com o API Gateway, oferecer suporte fluido nas operações síncronas e assíncronas.

6.2. API Gateway

O API Gateway é um componente central na arquitetura de microsserviços, servindo como ponto de entrada único para todos os pedidos realizados pelos clientes via HTTP. Este atua como intermediário, encaminhando os pedidos para os microsserviços apropriados. Além disso, assegura a autenticação correta dos utilizadores e a autorização adequada no acesso aos recursos.

Autenticação

O processo de autenticação no API Gateway utiliza JSON Web Tokens (JWT), armazenados do lado do cliente. Este processo funciona da seguinte forma:

- **Validação de *tokens***: Os *tokens* enviados pelos clientes são validados através de um segredo partilhado com o microsserviço de utilizadores, garantindo assim a sua autenticidade.

Os *tokens* armazenam informações sobre a identificação do utilizador e o seu tipo, servindo como base para validações futuras.

Autorização
A autorização é realizada com base num atributo na base de dados, permitindo comparações e pesquisas de forma mais versátil, o que facilita, por exemplo, o caso de futuras implementações de partilha entre utilizadores ou a criação de um quadro com projetos públicos, entre outros cenários.

Rate Limiting
Para evitar uma sobrecarga excessiva do sistema, cujo ponto de entrada é o próprio Gateway, foi implementado um sistema de *rate limiting*. Este mecanismo foi desenvolvido com o apoio do Redis para permitir a distribuição deste componente, sendo que o mesmo pode ser configurado para correr com multiplas instancias atraves do HPA.

6.3. WS Gateway

O WS Gateway é um componente essencial para o suporte a comunicações em tempo real no sistema. Este atua como ponto de entrada para conexões via WebSocket, garantindo que as mensagens entre o cliente (frontend) e os serviços internos sejam encaminhadas de forma eficiente, segura e escalável. O Gateway é particularmente importante para funcionalidades que requerem atualizações em tempo real, como o acompanhamento do estado de processamento de imagens e a receção dos respetivos resultados.

O componente será novamente, desenvolvido em JavaScript com o auxílio do Redis, pelos motivos supramencionados.

Autenticação
O processo de autenticação funciona de forma idêntica ao descrito no capítulo sobre o API Gateway.

6.4. Microserviços

Nas aplicações dos microsserviços, será utilizada a linguagem de programação ***JavaScript***, juntamente com o ambiente de execução ***Node.js*** e a framework ***Express***, bem como o ***Mongoose*** para ligação à Base de Dados.

Abaixo encontram-se especificações adicionais sobre cada microserviço.

6.4.1. Microsserviço de Subscrições

Stripe
O Stripe é um serviço externo para a gestão de pagamentos que decidimos integrar no nosso sistema. Este opera através de pedidos HTTP simples ou notificações via webhooks. Para simplificar a interação com o serviço, utilizaremos a biblioteca oficial do Stripe para JavaScript.

Implementámos o Stripe de forma a permitir a utilização de dinheiro real, garantindo a funcionalidade necessária para transações em ambiente de produção.

6.5. Microsserviço de Utilizadores

Autenticação
Conforme referido anteriormente este microsserviço é responsável pela geração dos tokens JWT (JSON Web Tokens).

Passwords
Para assegurar o cumprimento dos requisitos legais do projeto, as palavras-passe dos utilizadores serão armazenadas de forma encriptada. Para tal, utilizaremos o algoritmo de encriptação *bcrypt*, reconhecido pela sua segurança e eficiência.

MFA - Autenticação Multifator (OTP)
Para reforçar a segurança, o microsserviço irá integrar um sistema de Autenticação Multifator (MFA) através de um código único gerado por meio de OTP (One-Time Password). Este mecanismo será implementado como uma camada adicional de segurança, exigindo que os utilizadores confirmem a sua identidade através de um código temporário, normalmente enviado por mensagem e-mail, além da palavra-passe.

6.6. Microsserviços de ferramentas

Segue abaixo uma tabela com todas as ferramentas implementadas e cada um dos responsáveis.

Ferramenta	Responsável
Filtro OCR	Rodrigo Ralha
Filtro de ajuste de Brilho	Rafael Peixoto
Filtro de ajuste de Contraste	Rafael Peixoto
Filtro de Binarização	Eduardo Cunha
Filtro de Recorte Inteligente	Rafael Peixoto
Filtro de Remoção de Fundo	João Coelho
Filtro de Contagem de Pessoas	Rui Gonçalves
Filtro de Adicionar Borda	Délio Alves
Filtro de Ajuste de Saturação	Flávio Silva
Filtro de Rotação de Imagem	Filipe Oliveira
Filtro de Redimensionamento	Filipe Oliveira
Filtro de Identificação de Objectos	Rodrigo Ralha

Estes serviços não possuem base de dados associada. É importante referir que o **filtro de remoção de fundo** não foi implementado na versão final do projeto devido ao tamanho e ao tempo de compilação das suas dependências. Apesar de estar completamente funcional, encontra-se apenas comentado.

Bibliotecas
Optámos por utilizar as seguintes bibliotecas para garantir a implementação eficiente dos requisitos relacionados com o processamento de imagens:

- **Tesseract OCR:** Utilizada para o processamento e extração de texto em imagens.
- **Sharp:** Aplicada para manipulação geral de imagens, incluindo operações como recorte e redimensionamento.
- **Sharp-remove-bg-ai:** Especificamente utilizada para a remoção do fundo (background) das imagens.
- **TensorFlow:** Integrado com o modelo coco-ssd, permitindo a deteção de objetos em imagens através de modelos avançados de machine learning.

Importa salientar que o filtro de marca de água (watermark) foi fornecido pelos professores e não implementado por nós. Como tal, este filtro não disponibiliza dados customizados para o Prometheus, ao contrarios das restantes componentes. No entanto, este ainda foi configurado e integrado em todo o worflow desenvolvido, sendo que mesmo assim consegue suportar upgrades sem tempo de inatividade (downtime).

6.7. Microsserviço de Projetos

Pipeline

Este microsserviço foi implementado com o objetivo de permitir a execução ordenada de várias ferramentas, de acordo com a sequência definida pelos utilizadores. Está associado a cada utilizador e a um conjunto de imagens, onde as ferramentas especificadas são aplicadas de forma automática e em ordem.

O pipeline assegura que cada etapa de processamento é concluída antes de avançar para a próxima, garantindo assim a integridade e coerência dos resultados. Esta abordagem é fundamental para cenários que exigem o uso encadeado de ferramentas para manipulação e processamento de imagens.

Interação com o Armazenamento
A gestão de armazenamento de imagens no sistema foi cuidadosamente planeada para equilibrar custos e desempenho. Utilizámos o MinIO como solução de armazenamento compatível com o padrão S3 para o desenvolvimento local. Neste ambiente, apenas as imagens finais foram armazenadas, sendo apagadas do armazenamento interno após o processamento. O MinIO/S3 oferece várias opções de configuração de persistência, como HDD, SSD, RAM e tape, desde que essas estejam assim disponíveis. Estas opções permitiriam uma redução de custos ao escolher o tipo de armazenamento mais adequado para diferentes necessidades. Estas políticas podem ser configuradas para realizarem a transição de uma midia para outra automaticamente atraves de regras de lifecycle.

Contudo, devido à maior latência do armazenamento em S3, as imagens são, inicialmente, descarregadas para um armazenamento partilhado, onde são aplicados filtros, sendo posteriormente apagadas para reduzir o número de solicitações de armazenamento. Num ambiente real, seria utilizado um servidor NFS de um provedor de serviços como a Google Cloud, para garantir um armazenamento mais eficiente e de baixo custo. No entanto, por questões de simplicidade e por estarmos a trabalhar num único nó, a equipa optou por mapear um volume local para simular este ambiente de forma mais prática.

6.8. Message Broker

O Message Broker é um elemento central na arquitetura de microsserviços do nosso sistema, garantindo a comunicação assíncrona entre os diversos componentes. Este permite que os microsserviços comuniquem de forma desacoplada, promovendo escalabilidade, resiliência e uma melhor organização do sistema.

Para atender às necessidades do sistema, optámos por utilizar o RabbitMQ como solução de Message Broker.

Single Point of Failure

Para evitar pontos únicos de falha no sistema, o RabbitMQ será configurado com mecanismos de resiliência, assegurando alta disponibilidade e integridade dos dados:

- **Cluster de RabbitMQ:**
  Múltiplas instâncias do RabbitMQ serão executadas num cluster, permitindo a distribuição da carga e garantindo a continuidade do serviço em caso de falha de uma instância.

6.9. Bases de Dados

As bases de dados desempenham um papel fundamental no armazenamento e na gestão das informações relacionadas com utilizadores, projetos e subscrições. Para a persistência de dados, como mencionado anteriormente, será utilizada a tecnologia MongoDB, um sistema de gestão de bases de dados não relacional.

7. Alterações na Arquitetura

Durante o desenvolvimento do sistema, algumas alterações na arquitetura foram necessárias para garantir maior eficiência e segurança. Uma das principais alterações foi a gestão do acesso aos dados armazenados no S3. Originalmente, os filtros eram projetados para ter acesso direto ao S3, permitindo um processamento de imagens de forma mais direta. Contudo, devido a questões de segurança e performance, foi decidido que os filtros não teriam acesso direto ao S3.

Durante o desenvolvimento, utilizámos MinIO como solução de armazenamento local para simular um ambiente S3. No entanto, é importante notar que a encriptação do S3 não foi configurada, uma vez que estamos a trabalhar em ambiente de desenvolvimento. A encriptação é uma configuração nativa do S3 (ou MinIO), que em um ambiente de produção seria ativada para garantir a segurança dos dados. Como estamos a desenvolver e testar localmente, a segurança dos dados não foi uma preocupação imediata, dado que os dados não são sensíveis e a persistência não foi configurada da mesma forma que seria em produção.

Para além disso, para permitir um começo de desenvolvimento mais rápido e frutuoso, o stripe foi configurado para realizar o seu setup inicial de forma autónoma, ou seja, a criação do produto e a definição dos preços é feita na inicialização da aplicação. Isto diminui a versatilidade do painel de configuração do stripe, pelo que em produtivo a nossa solução teria de ser modificada para permitir verificar na base de dados os ids e não depender da não modificação das configurações.

Da mesma forma, a gestão de backups também não foi implementada, uma vez que os backups são outra configuração crucial do S3 e do MongoDB para um ambiente de produção. No entanto, como estamos a trabalhar em desenvolvimento, onde a persistência de dados não é necessária, não implementámos backups, já que não há dados críticos a preservar. Além disso, no ambiente real, a configuração de backups seria uma parte importante da estratégia de resiliência e recuperação do sistema.

Por fim, no ambiente de desenvolvimento, optámos por remover toda a persistência de dados, já que o objetivo era testar funcionalidades sem a necessidade de manter dados entre execuções. Em um ambiente de produção, a persistência seria tratada de forma robusta, garantindo a integridade e a disponibilidade dos dados.

8. Padrões Design

O projeto implementa diversos Padrões Design que estruturam e organizam sua lógica de forma modular e eficiente, garante robustez e escalabilidade. Esses padrões ajudam a resolver problemas específicos, promove boas práticas de desenvolvimento.

O padrão MVC (Model-View-Controller) é utilizado para separar responsabilidades no sistema. A camada de Model faz a gestão dos dados e a lógica relacionada às assinaturas, como informações sobre os utilizadores, tipos de planos e o estado atual das assinaturas. A camada de View é representada pelas respostas enviadas para os clientes em formato JSON, exibe informações como planos disponíveis, histórico de transações e o status de uma assinatura. A camada de Controller é responsável por lidar com a lógica de negócios, conecta os dados do Model às ações necessárias e às respostas enviadas para o cliente. Isso inclui operações como criação de assinaturas, cancelamento e o tratamento de notificações externas.

O padrão Observer é aplicado no sistema de notificações do Stripe, onde eventos como pagamentos bem-sucedidos, falhas de pagamento ou cancelamentos de assinaturas são observados. O sistema reage automaticamente a essas notificações, altera o status da assinatura do utilizador com base no evento recebido. Essa abordagem permite que o sistema se mantenha sincronizado com mudanças externas de maneira eficiente e em tempo real, sem a necessidade de consultas manuais.

O padrão Singleton é utilizado para garantir que recursos críticos sejam instanciados apenas uma vez e reutilizados em todo o sistema. Um exemplo é a integração com a API de terceiros, onde uma única instância é criada para gestão de todas as interações, evita duplicação de configurações. Outro exemplo é a conexão com o banco de dados, que é configurada uma única vez e compartilhada por todas as operações, promove eficiência e consistência.

O padrão Chain of Responsibility usado no pipeline de processamento de requisições. Cada requisição percorre uma cadeia de processamento, onde diferentes componentes verificam ou manipulam aspetos específicos antes de encaminhá-la a próxima fase. Isso inclui a autenticação de utilizadores, recolha de métricas e validação de dados das requisições. Além disso, a validação segue este padrão ao processar parâmetros, corpo e query strings de forma sequencial e interrompe o fluxo em caso de erros.

O padrão Iterator é utilizado para iterar sobre coleções de dados e executar operações específicas. No sistema, este padrão surge ao percorrer intervalos de assinatura para configurar os planos disponíveis e garantir que todas as opções estejam devidamente configuradas. Também é aplicado no tratamento de notificações, onde diferentes tipos de eventos são processados iterativamente, com ações a serem executadas com base no tipo de evento recebido.

Com a utilização desses padrões design, foi possível construir uma arquitetura organizada e modular, que faz uso da reutilização de código de forma eficiente. Essa abordagem estruturada simplificou a integração do projeto com serviços externos como Stripe e MongoDB, o sistema torna-se mais escalável, robusto e prático para atender às necessidades atuais e futuras.

9. Building Block View

Nesta secção, apresentamos o PictuRAS sob a perspetiva da sua decomposição em diversos componentes, tal como definido pela equipa docente. Cada componente pode ser entendido como um subsistema, módulo, classe, interface, pacote, biblioteca, entre outros.

O primeiro diagrama oferece uma visão geral do sistema, com um elevado nível de abstração. Esta visão inclui todos os microsserviços descritos anteriormente, um API Gateway, uma fila de mensagens e diferentes tipos de Frontend. Além disso, são também referenciados outros componentes externos ao sistema, devido à sua interação com o PictuRAS.

Nos capítulos seguintes, será dada uma descrição mais detalhada de cada um dos componentes definidos na fase anterior.

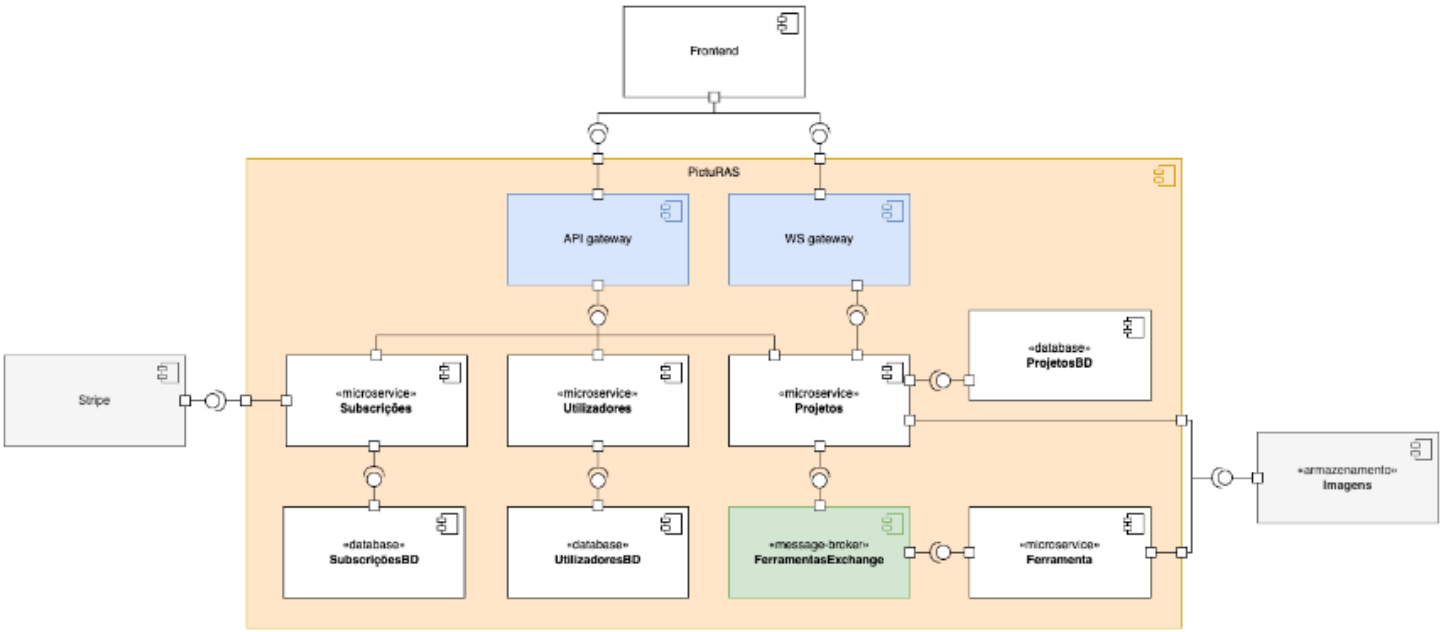


Figura 5: Diagrama de componentes do PictuRAS

9.1. Frontend

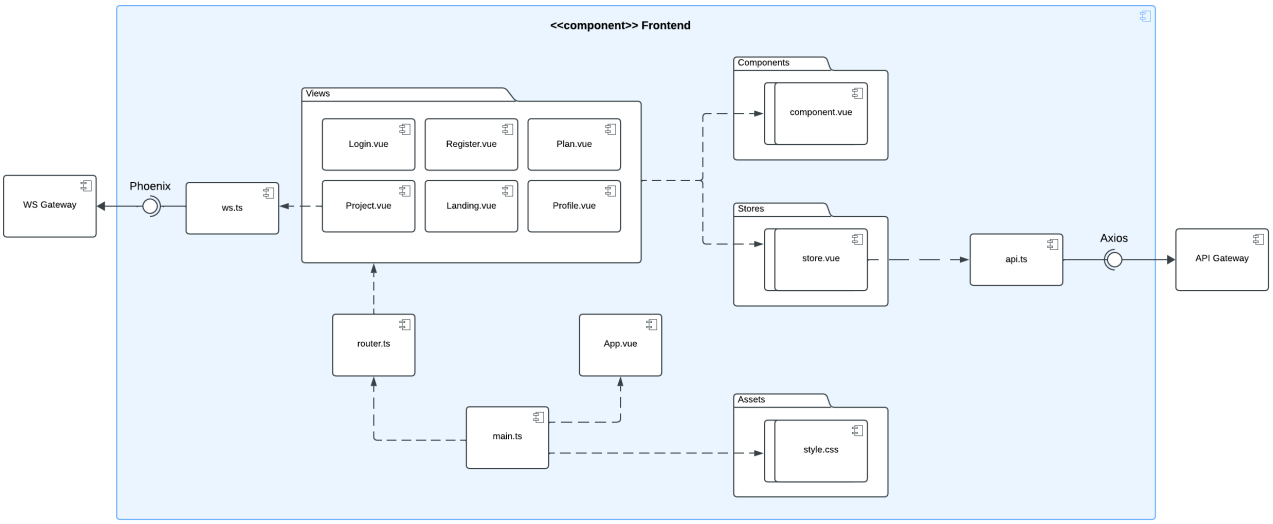


Figura 6: Diagrama de componentes do Frontend

9.2. API Gateway

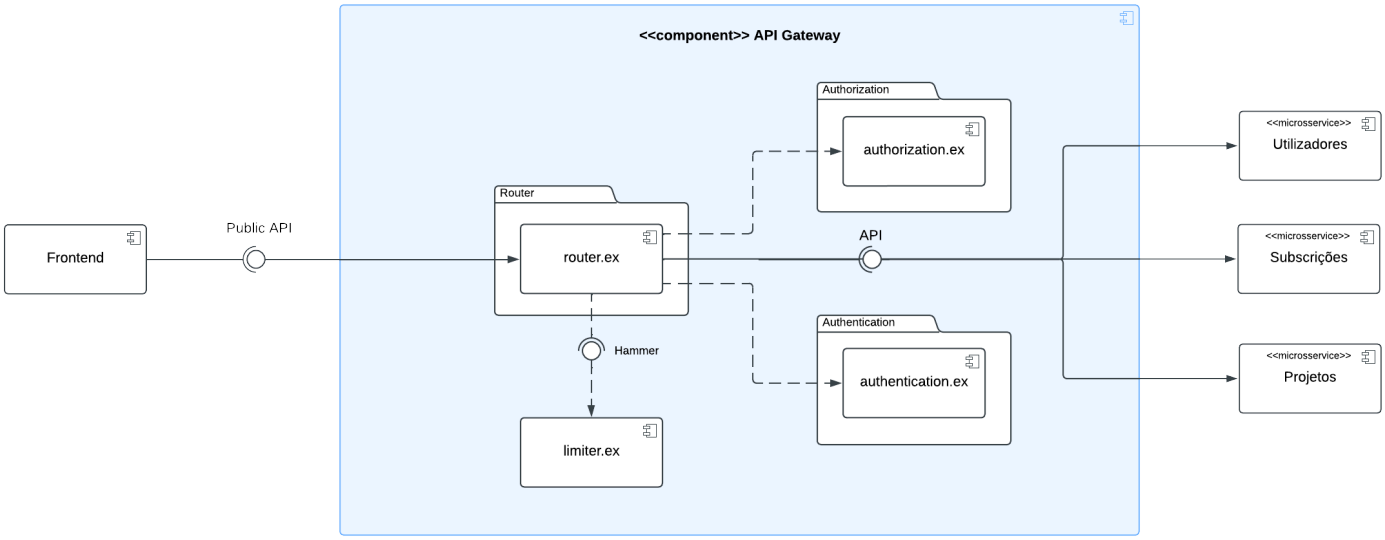


Figura 7: Diagrama de componentes do API Gateway

9.3. WS Gateway

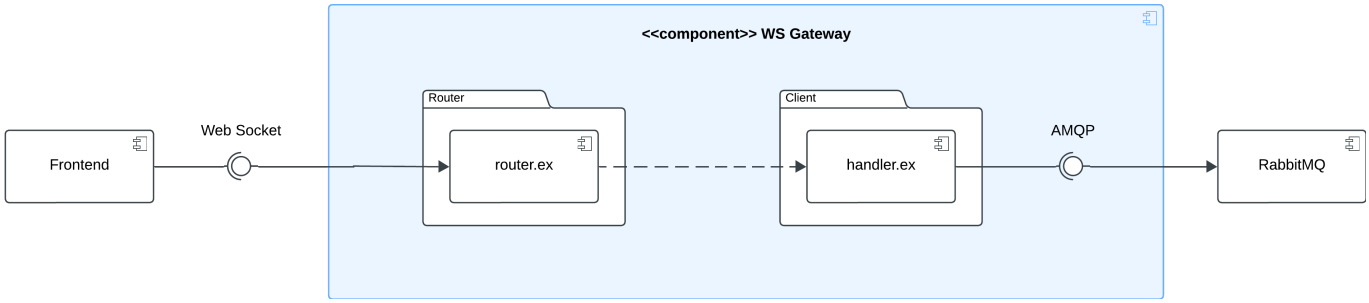


Figura 8: Diagrama de componentes do WS Gateway

9.4. Orquestrador

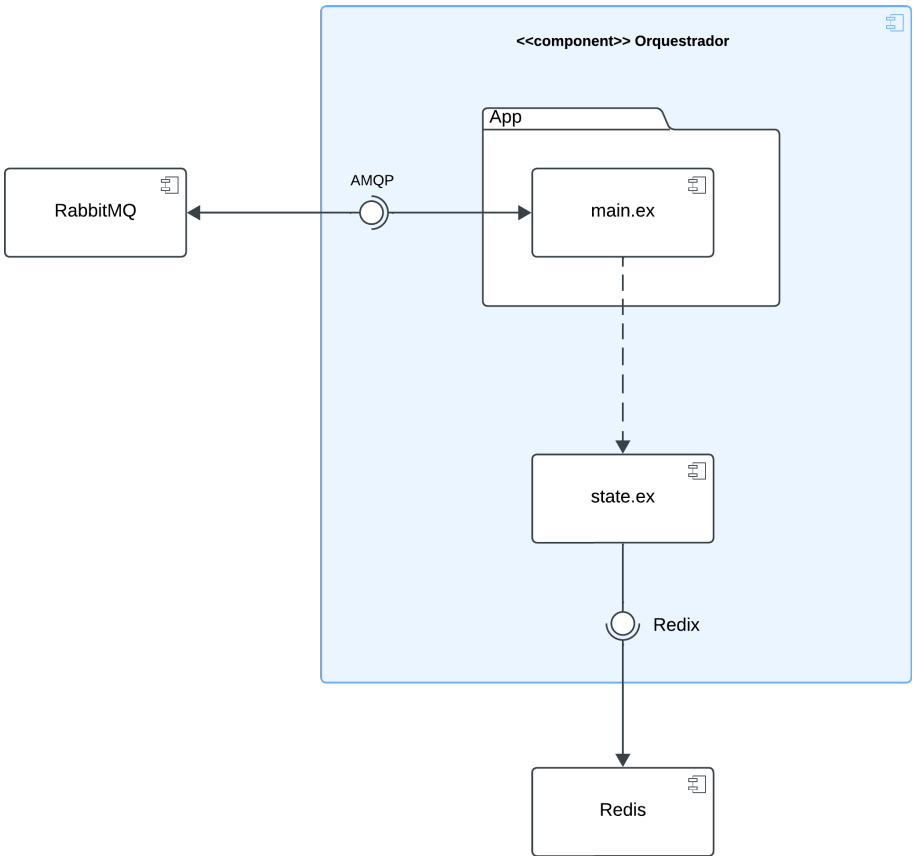


Figura 9: Diagrama de componentes do Orquestrador

9.5. Subscrições

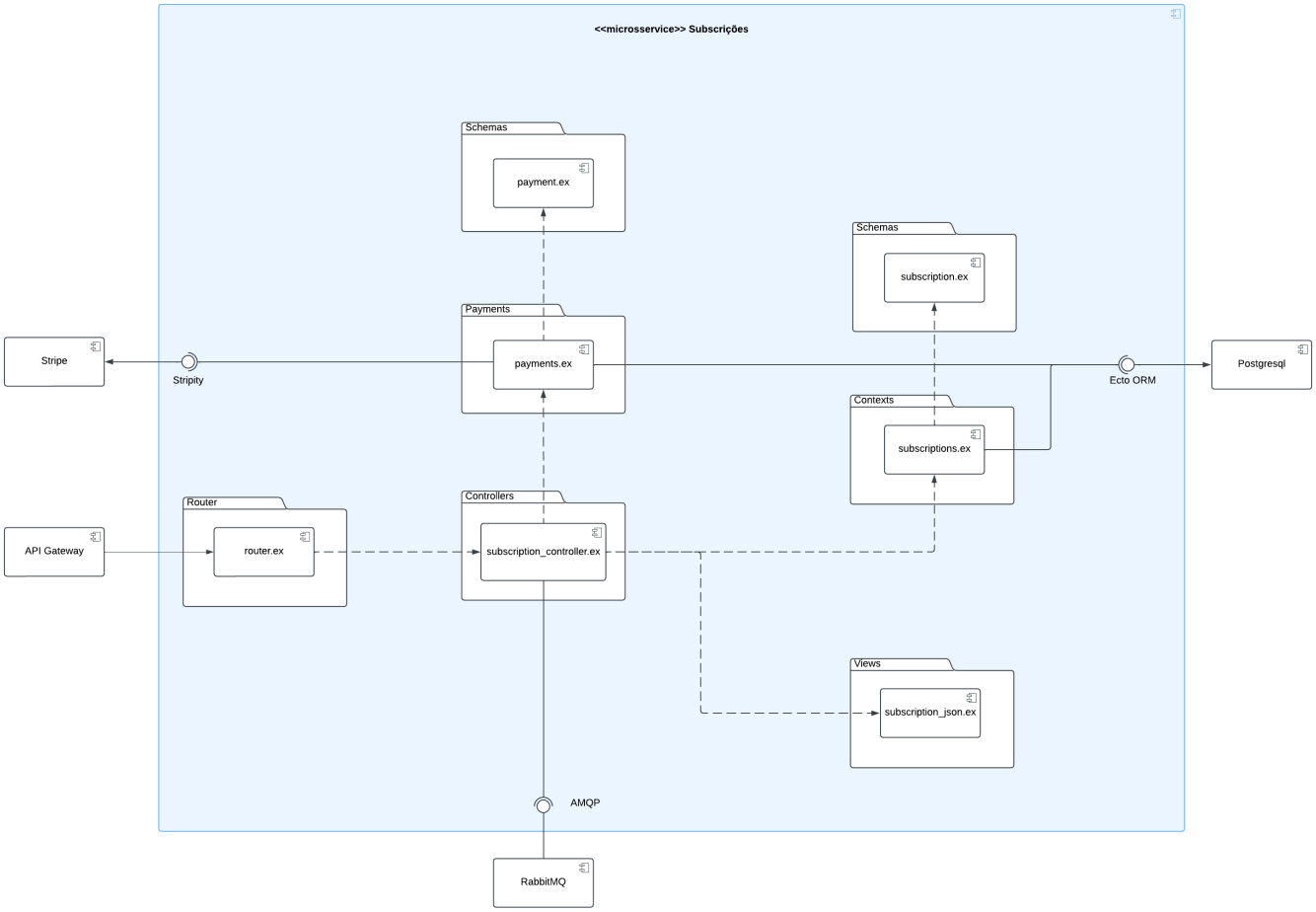


Figura 10: Diagrama de componentes do Microsserviço de Subscrições

9.6. Utilizadores

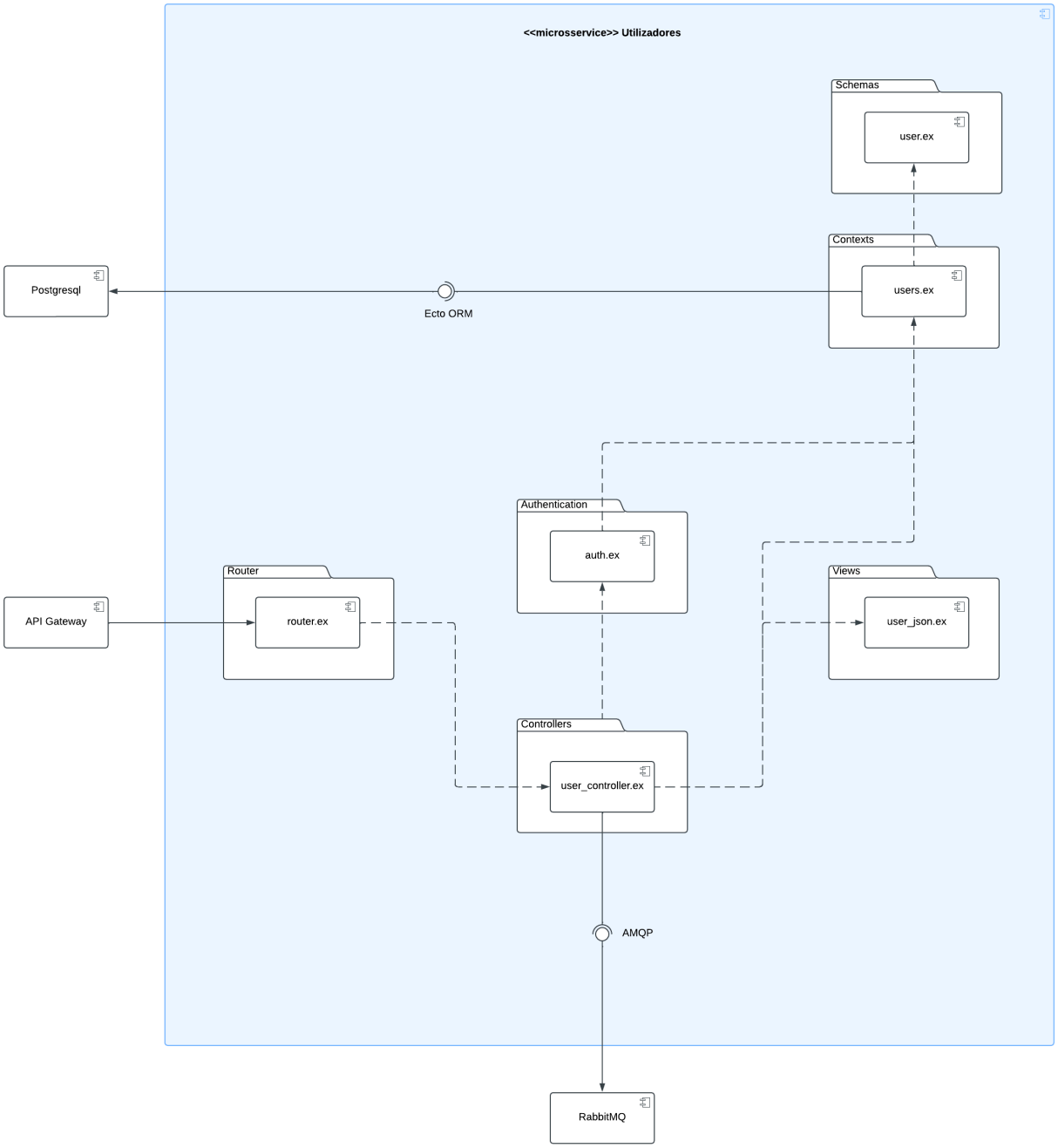


Figura 11: Diagrama de componentes do Microsserviço de Utilizadores

9.7. Projetos

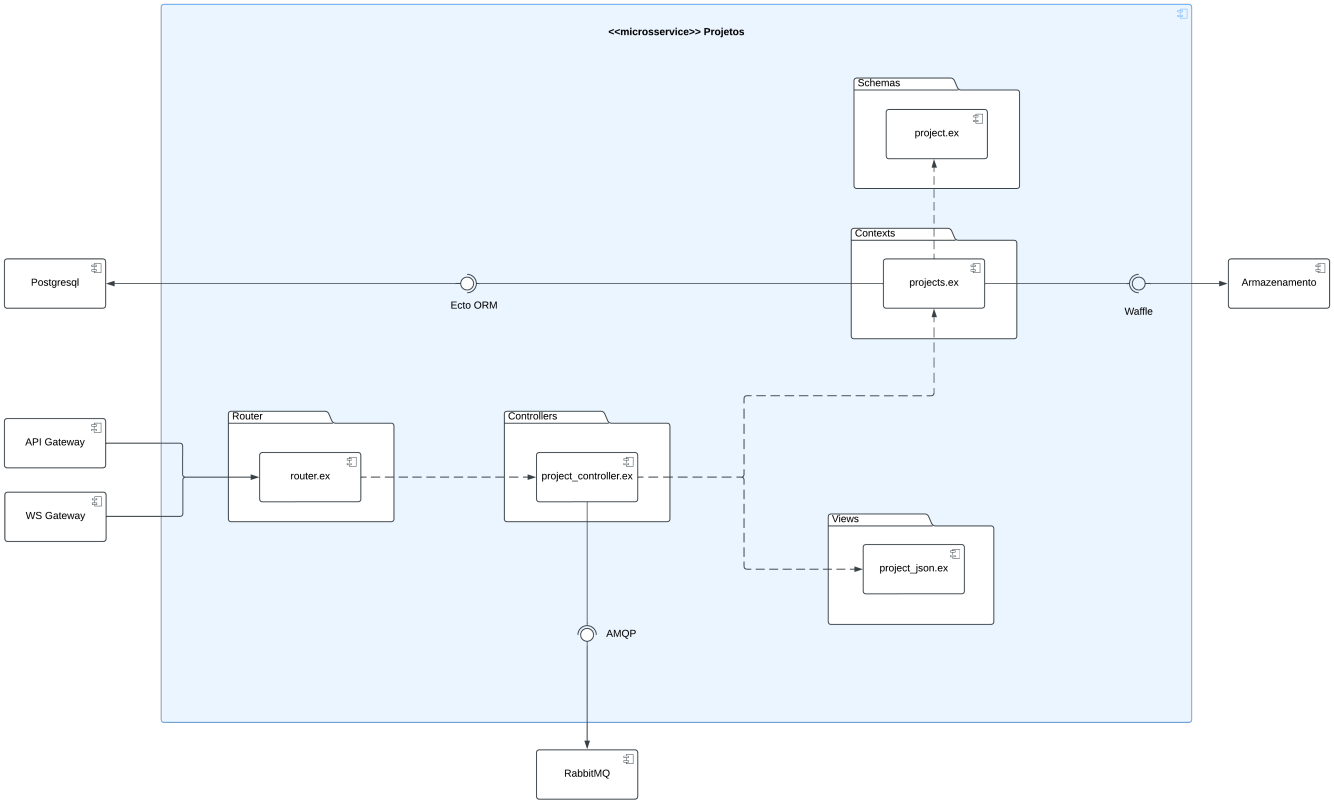


Figura 12: Diagrama de componentes do Microsserviço de Projetos

9.8. Ferramenta

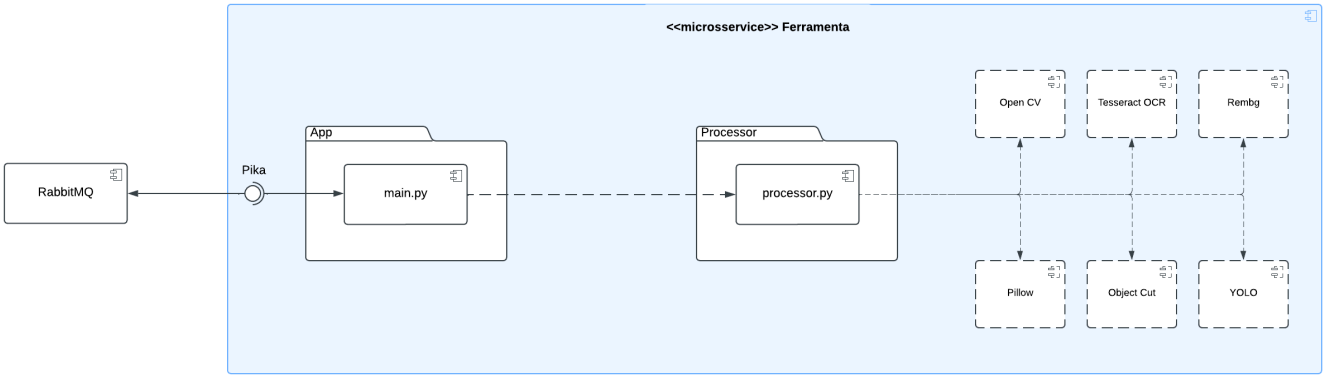


Figura 13: Diagrama de componentes do Microsserviço de uma ferramenta

10. Runtime View

Na secção abaixo, é exposto o comportamento dinâmico do sistema, conforme definido na fase anterior, através de diagramas de sequência relativos aos use cases desenvolvidos. Estes diagramas concentram-se nas interações principais entre os componentes do sistema.

10.1. Registo

Este use case descreve o registo de um utilizador que ainda não possui conta no PictuRAS. O processo inicia-se com o utilizador a inserir os dados necessários, seguido de uma validação e autenticação por parte do sistema.

Após a validação inicial, o utilizador recebe um código OTP (One-Time Password) enviado por e-mail como parte do processo de autenticação multifator (MFA). Este código é necessário para confirmar a identidade do utilizador antes de avançar para a criação da conta.

De seguida, são apresentados os vários perfis disponíveis: gratuito, mensal e anual. Para os dois perfis pagos, o utilizador deverá inserir os dados de pagamento e confirmá-los, ficando, no final, corretamente registado no sistema.

Durante o processo de inserção de dados por parte do utilizador, o sistema realiza verificações adequadas e apresenta mensagens de erro, caso aplicável.

10.2. Login

O use case relativo ao login descreve o processo de autenticação de um utilizador previamente registado no sistema. O utilizador insere as suas credenciais, que são submetidas para validação.

Caso as credenciais estejam incorretas, o sistema informa o utilizador sobre o erro e permite a reinserção dos dados. Uma vez validadas corretamente as credenciais, o utilizador é submetido ao processo de autenticação multifator (MFA), onde, para completar o login, são gerados dois tokens:

1. Token de Sessão: O primeiro token é gerado e enviado após a validação das credenciais iniciais. Este token garante que o utilizador tenha acesso à sua conta, mas ainda necessita de validação adicional através do MFA.

2. Token OTP (One-Time Password): O segundo token, um código OTP, é enviado ao utilizador (por e-mail ou outro meio de comunicação seguro). O utilizador deve inserir este código no sistema para completar o processo de autenticação.

Após a validação do código OTP, o sistema autentica o utilizador e permite o acesso completo à sua conta.

Este processo assegura uma autenticação mais segura, garantindo que o utilizador é realmente quem diz ser antes de aceder aos recursos do sistema.

10.3. Carregar imagens

O *use case* de carregar imagens refere-se ao ato de o utilizador carregar imagens para um determinado projeto ou para um novo.

Após carregar as imagens para o sistema, é apresentada a possibilidade de criar um novo projeto, sendo necessário, para isso, introduzir um nome para o mesmo. O utilizador pode, também, simplesmente selecionar o projeto desejado. Depois de escolhido o projeto, as imagens são adicionadas ao mesmo.

Tal como em *use cases* anteriores, são realizadas validações quanto ao tamanho das imagens ou ao número de imagens carregadas, sendo sempre apresentadas mensagens relativas aos erros detetados.

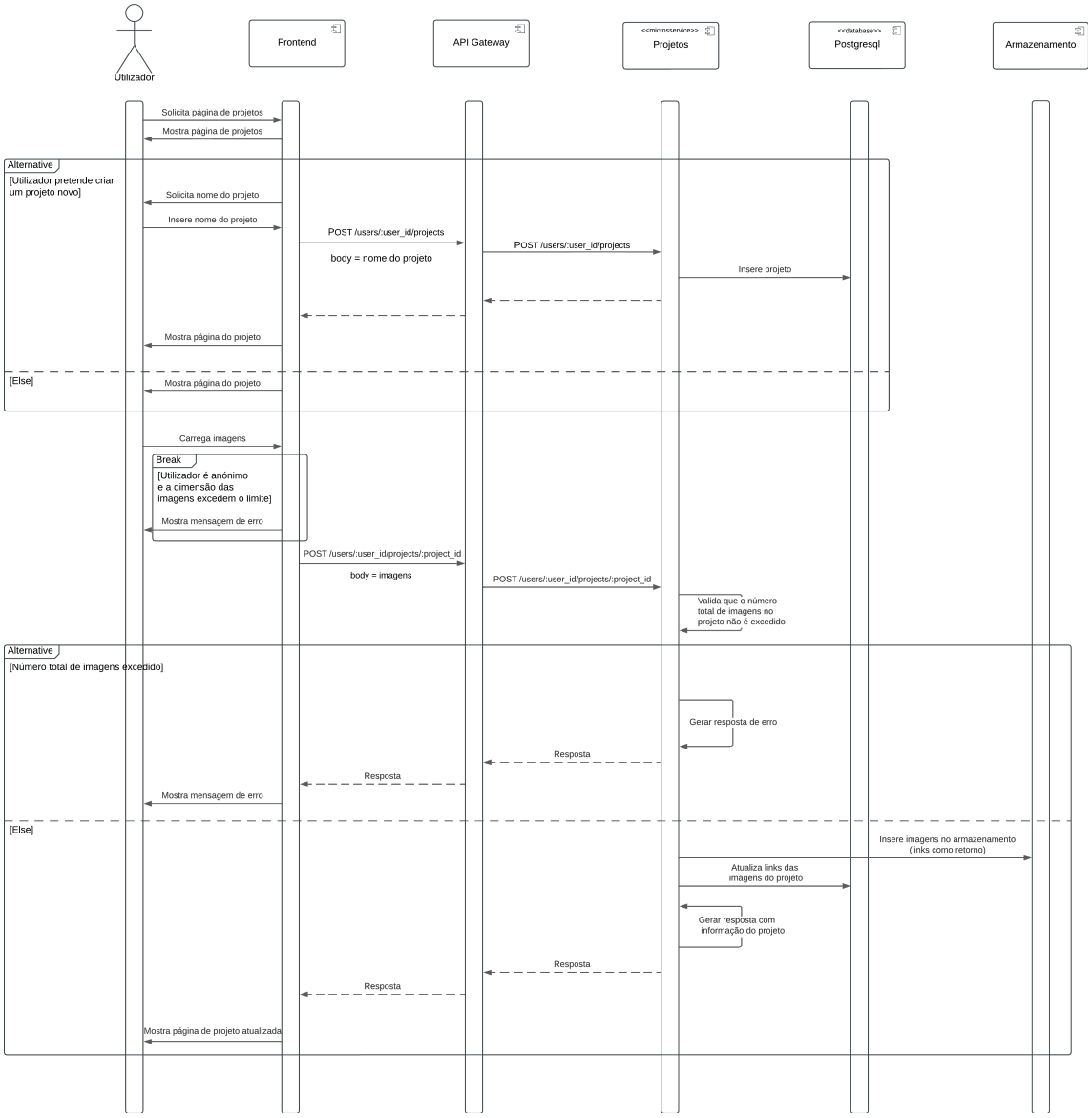


Figura 14: Diagrama de sequência referente ao carregamento de imagens

10.4. Aplicar encandeamento de ferramentas a conjunto de imagens

Este *use case* refere-se à funcionalidade de aplicar ferramentas (com diversos parâmetros) a um projeto.

Após selecionar o projeto desejado, o utilizador, de forma interativa, escolhe e parametriza as ferramentas pretendidas. De seguida, ocorre o processamento das imagens por parte do sistema. Caso o utilizador esteja satisfeito, poderá, no final, descarregar as imagens processadas.

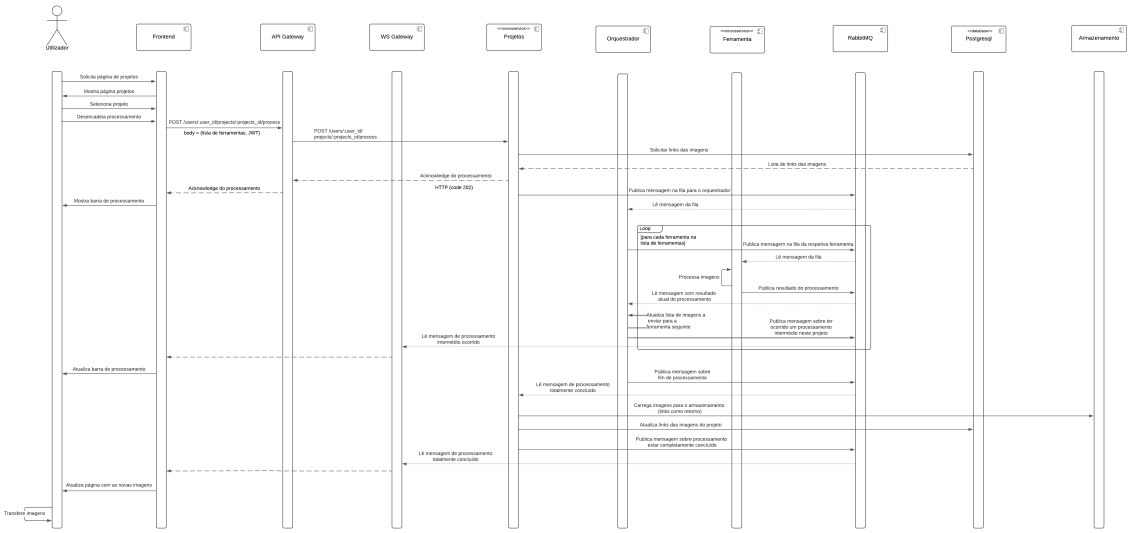


Figura 15: Diagrama de sequência referente à aplicação de um encadeamento de ferramentas a conjunto de imagens

11. Deployment View

Nesta secção, vamos abordar as principais diferenças relativamente ao previsto nas fases anteriores, dando ênfase ao workflow tanto nas vertentes de desenvolvimento, como na de produção. Para começar, é importante ressaltar a utilização da ferramenta Helm, um gestor de pacotes para Kubernetes, tendo feito uso do mesmo para instalar várias das componentes essenciais, como as bases de dados MongoDB, as camadas de cache e de memória partilhada Redis e o Message Broker RabbitMQ. O nosso chart foi concebido para permitir elevada versatilidade nas configurações, permitindo, por exemplo, que através de uma única key, realizar o deployment tal como seria feito em ambiente de produção.

Em contra partida, para desenvolvimento local, foi necessária a utilização do Minikube para criar um cluster de Kubernetes, utilizado para testar e aprimorar a aplicação. No entanto, esta abordagem apresenta várias vantagens, sendo que assemelha-se ao de produção, assegurando que o verdadeiro deployment funcione conforme o esperado.

No ambiente de produção, para assegurar consistência da infraestrutura, optamos por utilizar técnicas de Infraestructure as Code, pelo que fez uso da ferramenta Terraform para o provisionamento automático. Para além disso, existe uma outra vertente na pipeline de desenvolvimento, tratando-se do próprio deploy da aplicação. Para tal, recorremos ao GitHub Actions para implantar o serviço de forma automática no cluster de kubernetes. Relativamente à organização do cluster o grupo deve mencionar, que **TODOS** os serviços foram desenvolvidos para permitir uma utilização concorrente, sendo que todos eles poderiam ter um Horizontal Pod Autoscaler, contudo para assegurar alguma coerência com o planeado previamente, optamos por não configurar dessa forma, mas tal alteração seria frívola.

Por último, o grupo pretende ainda frisar a importância da análise e monitorização dos serviços, para permitir retirar melhores insights e optimizar o sistema. Neste sentido, configuramos automaticamente a stack do Prometheus e do Grafana em produtivo, sendo que foram desenvolvidas métricas próprias integradas automaticamente com o Prometheus. É de mencionar que optamos por utilizar o (Grafana) Loki ao invés do previsto ELK, devido à sua superior integração com as restantes componentes.

12. Métodos de Trabalho

A equipa de desenvolvimento é composta por 8 membros, sendo decidido que será organizada em duas equipas de trabalho: frontend e backend, com 4 membros atribuídos a cada equipa.

Esta divisão foi definida de forma a garantir uma abordagem mais focada e eficiente nas diferentes áreas de desenvolvimento.

Adicionalmente, optámos por utilizar o GitHub como ferramenta principal para a gestão do repositório e do backlog, facilitando a distribuição e o acompanhamento das tarefas de forma estruturada e colaborativa.

Para otimizar a comunicação e colaboração em tempo real, utilizamos o Discord como plataforma principal para conversas, reuniões de equipa e troca de ideias. Isso garantiu uma comunicação fluida entre os membros das equipas, permitindo discussões rápidas sobre o progresso do projeto, bem como a resolução ágil de problemas ou dúvidas técnicas.

Adicionalmente, utilizámos três ferramentas principais que desempenharam papéis cruciais durante o desenvolvimento do projeto:

Insomnia: Esta ferramenta foi essencial para o processo de debugging e testes a nível do backend. Com o Insomnia, conseguimos simular e validar chamadas de API, garantir que os endpoints desenvolvidos estavam a funcionar corretamente e com a performance esperada.

Minikube: Para criar e gerir clusters locais do Kubernetes, adotámos o Minikube. Ele facilitou os testes e a implementação em um ambiente controlado, permitiu-nos simular a infraestrutura de produção com eficiência.

Helm: O Helm foi utilizado como gestor de pacotes Kubernetes e ajudou-nos na instalação e na configuração. Optámos por implementar um conjunto de pacotes fundamentais para o nosso projeto.

13. Grau de Execução da Solução

Nesta secção, apresentamos uma análise dos requisitos funcionais e não funcionais que foram cumpridos na solução, com o objetivo de avaliar o nível de completude do produto final.

13.1. Requisitos Funcionais

O documento de requisitos utilizado como base para o desenvolvimento do projeto identifica um total de 38 requisitos funcionais. Com as funcionalidades implementadas até esta fase, foi possível cumprir os seguintes:

Requisito	**Descrição**	***Use Case***	**Prioridade**
Req1	O utilizador autentica-se utilizando as suas credenciais.	2	Must
Req2	O utilizador anónimo regista-se.	1	Must
Req3 e Req4	O utilizador escolhe o plano de subscrição (Gratuito ou *Premium*).	1	Must
Req6	O utilizador cria um projeto.	3	Must
Req7	O utilizador lista os seus projetos.	3	Must
Req8	O utilizador acede à área de edição de um projeto.	3	Must
Req9	O utilizador carrega imagens para um projeto.	3	Must
Req10	O utilizador remove uma imagem do projeto.	4	Must
Req11	O utilizador adiciona uma ferramenta de edição ao projeto.	4	Must
Req12	O utilizador desencadeia o processamento de um projeto.	4	Must
Req13	O utilizador transfere o resultado de um projeto para o dispositivo local	4	Must
Req14	O utilizador altera a ordem das imagens de um projeto.	4	Could
Req15	O utilizador altera a ordem das ferramentas de um projeto.	4	Must
Req16	O utilizador altera os parâmetros das ferramentas.	4	Must

Req17	O utilizador cancela o processamento de um projeto durante a sua execução.	4	Should
Req18	O utilizador registado acede às suas informações de perfil.	1	Should
Req19	O utilizador edita o seu perfil.	1	Should
Req20	O utilizador com subscrição Premium altera a recorrência de pagamento entre mensal ou anual.	1	Could
Req21	O utilizador *premium* cancela a sua subscrição *premium*.	1	Should
Req22	O utilizador registado termina a sua sessão.	2	Must
Req23	O utilizador registado remove a sua conta e dados.	1	Must
Req24	O utilizador partilha o resultado da edição de uma imagem diretamente nas redes sociais.	4	Must
Req25	O utilizador recorta manualmente imagens.	4	Must
Req26	O utilizador escala imagens para dimensões específicas.	4	Must
Req27	O utilizador adiciona borda a imagens.	4	Must
Req28	O utilizador altera a saturação a imagens.	4	Must
Req29	O utilizador ajusta o brilho a imagens.	4	Must
Req30	O utilizador ajusta o contraste a imagens.	4	Must
Req31	O utilizador binariza imagens.	4	Should
Req32	O utilizador roda imagens.	4	Must
Req33	O utilizador aplica um algoritmo de recorte automático de imagens com base no seu conteúdo.	4	Must
Req34	O utilizador aplica ajustes automáticos de otimização das imagens com base no conteúdo.	4	Could
Req35	O utilizador remove o fundo da imagem, mantendo apenas o objeto principal.	4	Must
Req36	O utilizador extrai texto de imagens.	4	Should
Req37	O utilizador aplica um algoritmo de reconhecimento de objetos em imagens.	4	Must
Req38	O utilizador aplica um algoritmo de contagem de pessoas em imagens.	4	Should

Tabela 20: Lista tabular dos requisitos funcionais prioritários

No total, foram satisfeitos 38 dos 38 requisitos funcionais, correspondendo a um grau de cobertura de cerca de 100%.

13.2. Requisitos Não Funcionais

No que diz respeito aos requisitos não funcionais, o documento de referência apresenta um total de 36. Destes, a equipa considera ter cumprido os seguintes:

Requisito	Descrição	Prioridade
Req1	A aplicação deve conter uma interface simples e clara.	Must
Req2	A aplicação deve conter ícones e botões que sejam claros e intuitivos.	Must
Req3	A aplicação deve utilizar cores suaves e neutras.	Could
Req4	A aplicação deve ser fácil de usar.	Must
Req5	A página de um projeto distingue visualmente entre as ferramentas básicas e avançadas.	Must
Req6	O utilizador vê todas as funcionalidades da aplicação mesmo que o seu perfil não lhes confira acesso.	Must
Req7	O utilizador cria um projeto implicitamente ao arrastar ficheiros para o dashboard da aplicação.	Must
Req8	O utilizador carrega imagens arquivadas num único ficheiro .zip.	Must
Req9	A listagem dos projetos é ordenada pela data do último acesso: mais recentes primeiro.	Should
Req10	A listagem dos projetos é pesquisável por nome.	Could
Req11	O utilizador é mantido informado acerca do estado de processamento de um projeto em tempo real.	Must
Req12	A aplicação fornece ajudas contextuais em várias páginas da aplicação, oferecendo informações guiadoras para o fluxo e a experiência do utilizador.	Could
Req13	A aplicação apresenta a pré-visualização do resultado da aplicação da sequência de ferramentas do projeto na imagem atualmente selecionada.	Must
Req14	A visualização de imagens grandes ou pequenas é auxiliada pelo utiliário zoom	Must
Req15	A visualização de imagens permite que se navegue em todas as duas direções arrastando o rato.	Must
Req16	A aplicação ajuda o utilizador a selecionar apenas ferramentas compatíveis.	Should
Req17	A aplicação fornece tutoriais ou dicas integradas.	Could
Req18	A aplicação é compatível com diferentes plataformas e browsers, incluindo de dispositivos móveis e desktop.	Must
Req19	A aplicação fornece feedback visual claro e imediato ao utilizador em caso de erro ou falha durante um procedimento demorado.	Should

Req20	O utilizador copia um link que aponta para a página do projeto.	Should
Req21	As ferramentas de edição de imagem devem ser processadas rapidamente.	Should
Req22	O sistema deve processar até 100 imagens ao mesmo tempo, sem quebras percetíveis no desempenho.	Must
Req23	A aplicação deve ser capaz de escalar horizontalmente de forma elástica para suportar o aumento de utilizadores e volume de processamentos, mantendo o desempenho mas também os custos controlados.	Must
Req24	A aplicação deve estar disponível 24 horas por dia, todos os dias.	Should
Req25	A aplicação deve ser integrável com outras plataformas e serviços de terceiros.	Should
Req26	A aplicação deve otimizar o uso de recursos computacionais do cliente (e.g., CPU, memória)	Should
Req27	A aplicação deve suportar os principais formatos de imagem, como JPEG, PNG, BMP e TIFF.	Should
Req28	A aplicação deve ser facilmente estendida com novas ferramentas de edição.	Must
Req29	A manutenção programada deve originar, no máximo, 10 minutos de downtime por mês, sendo realizada fora dos horários de maior utilização.	Should
Req30	O suporte ao cliente deve estar disponível por chat ou email, com tempos de resposta rápidos.	Should
Req31	As imagens devem ser cifradas durante o envio e quando são armazenadas no sistema.	Should
Req32	O sistema deve ser projetado para facilitar a execução de testes.	Must
Req33	A aplicação deve realizar backups automáticos dos dados e imagens dos utilizadores.	Should
Req34	O sistema garante que apenas os utilizadores que carregam as imagens têm acesso a elas.	Must
Req35	O sistema deve usar MFA para aumentar a segurança dos utilizadores registados.	Could
Req36	O sistema deve garantir que apenas utilizadores com permissões adequadas acedem a funcionalidades restritas, aplicando políticas de controlo de acesso claras e seguras.	Should

Req37	As sessões dos utilizadores devem ter um logout automático após longos períodos de inatividade.	Could
Req38	O sistema deve cumprir o RGPD.	Should
Req39	A aplicação deve estar disponível em vários idiomas.	Should
Req40	A aplicação deve ajustar-se aos diferentes formatos de data e hora.	Should
Req41	Os Termos e Condições da aplicação devem ser claros e acessíveis.	Should

Tabela 23: Lista tabular dos requisitos funcionais prioritários

No total, foram satisfeitos 40 dos 41 requisitos não funcionais, o que representa um grau de cobertura de 97.5%. À exceção do Req39, todos os requisitos foram cumpridos.

14. Conclusão e Trabalho Futuro

Este trabalho possibilitou a criação de um sistema complexo, fundamentado numa arquitetura distribuída que assegura a escalabilidade, flexibilidade e eficiência necessárias para um ambiente dinâmico e em constante evolução. A estrutura modular adotada permite que diferentes componentes do sistema operem de forma independente, facilitando a adaptação e evolução do sistema sem impactar significativamente outras partes. A capacidade de escalar conforme a demanda e de integrar novos recursos de forma ágil foi um dos pontos chave na conceção do sistema, garantindo que ele possa crescer de acordo com as necessidades futuras.

Face à complexidade do desafio, acreditamos ter cumprido a maior parte dos requisitos e objetivos estipulados para a realização deste trabalho. No entanto, reconhecemos que nem todos os requisitos foram implementados na totalidade. Algumas funcionalidades ficaram fora do escopo devido à limitação de tempo e à necessidade de priorizar as tarefas mais críticas para o funcionamento do sistema. Apesar disso, o trabalho desenvolvido correspondeu amplamente às expectativas iniciais e foi conduzido com rigor, sempre com o objetivo de entregar uma solução robusta e eficiente.

Este trabalho foi extremamente importante, pois proporcionou a experiência do desenvolvimento completo de uma aplicação baseada em microsserviços, envolvendo a integração de diversas tecnologias e ferramentas. A experiência adquirida ao longo deste projeto contribuiu significativamente para o nosso crescimento profissional, permitindo-nos entender de forma prática os desafios que surgem na implementação de sistemas distribuídos e escaláveis. Contudo, também consideramos que foi um desafio altamente complexo, que exigiu muito mais do que os conhecimentos adquiridos nas aulas e ao longo da licenciatura. Para além disso, foi necessária uma boa gestão de tempo e recursos, bem como uma colaboração eficiente entre os membros da equipa.