Universidade do Minho
Escola de Engenharia

Mestrado em Engenharia Informática
Requisitos e Arquiteturas de Software

Projeto PictuRAS - Fase 3

Janeiro 2025

pg57511	**Benjamim Rodrigues**
pg57868	**Bruno Gião**
pg57549	**Hugo Gomes**
pg55956	**João Magalhães**
pg57565	**João Ribeiro**
pg55972	**José Pacheco**
pg57884	**Lara Pereira**
pg57885	**Lingyun Zhu**
pg57582	**Luís Ferreira**
pg55992	**Pedro Lopes**

Índice

1 Introdução e Objetivos 2

2 Análise Crítica 3
2.1 Levantamento de requisitos 3
2.2 Arquitetura do sistema 5
2.3 Implementação 6
2.3.1 Mudanças em relação à solução arquitetural proposta 6
2.3.2 Endpoints 6
2.3.2.1 Login 6
2.3.2.2 Edição do Projeto 7
2.3.2.3 Página de Projetos 7
2.3.2.4 Página de Registo 7
2.3.2.5 Página de Subscrições 7
2.3.3 Tecnologias Utilizadas 8

3 Microsserviços 9

4 Problemas e implementações para o futuro 10

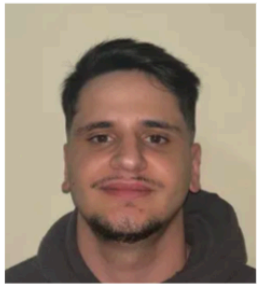
5 Conclusão 11

PG57549

PG55956

PG55992

PG55972

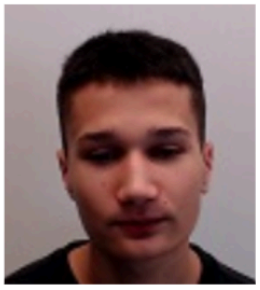


PG57565

PG57868

PG57884

PG57582



PG57511

PG57885

1 Introdução e Objetivos

A presente documentação visa descrever detalhadamente a implementação do *Minimum Viable Product* (MVP) do projeto pictuRAS, desenvolvido no âmbito da Unidade Curricular de Requisitos e Arquiteturas de Software. O objetivo desta fase é entregar uma solução funcional que permita aos utilizadores executar as seguintes funcionalidades fundamentais:

- Criação e listagem de projetos.
- Carregamento de imagens para os projetos.
- Adição de ferramentas de edição aos projetos.
- Pré-visualização e aplicação de ferramentas de edição.
- Descarregamento dos resultados gerados.

Além destas funcionalidades, esta fase também inclui a integração de um microsserviço de *watermark* disponibilizado pela equipa docente. O desenvolvimento foi orientado pela arquitetura especificada na fase anterior disponibilizada e inclui alterações para acomodar os desafios encontrados durante a implementação.

2 Análise Crítica

2.1 Levantamento de requisitos

Em relação aos requisitos sugeridos pela solução arquitetural proposta para esta fase, depois de algumas dificuldades com tempo e implementação das tecnologias, decidimos ter algum cuidado em relação à escolha dos requisitos a seguir, em vez de focarmo-nos em fazer todos.

Dito isto, encontram-se abaixo os requisitos sugeridos que não foram implementados na nossa solução:

Requisitos Funcionais:

Requisito	**Descrição**	**Contexto**	**Prioridade**	**Estado**
Req1	O utilizador autentica-se utilizando as suas credenciais.	2	Must	Completo
Req2	O utilizador anónimo regista-se.	1	Must	Completo
Req3 e Req4	O utilizador escolhe o plano de subscrição (Gratuito ou Premium).	1	Must	Completo
Req6	O utilizador cria um projeto.	3	Must	Completo
Req7	O utilizador lista os seus projetos.	3	Must	Completo
Req8	O utilizador acede à área de edição de um projeto.	3	Must	Completo
Req9	O utilizador carrega imagens para um projeto.	3	Must	Completo
Req11	O utilizador adiciona uma ferramenta de edição ao projeto.	4	Must	Completo
Req12	O utilizador desencadeia o processamento de um projeto.	4	Must	Completo
Req13	O utilizador transfere o resultado de um projeto para o dispositivo local	4	Must	Incompleto
Req18	O utilizador registado acede às suas informações de perfil.	1	Should	Incompleto
Req21	O utilizador *premium* cancela a sua subscrição *premium*.	1	Should	Completo
Req22	O utilizador registado termina a sua sessão.	2	Must	Completo

Requisito	Descrição	Contexto	Prioridade	Estado
Req25	O utilizador recorta manualmente imagens.	4	Must	Completo
Req26	O utilizador escala imagens para dimensões específicas.	4	Must	Completo
Req27	O utilizador adiciona borda a imagens.	4	Must	Completo
Req29	O utilizador ajusta o brilho a imagens.	4	Must	Completo
Req30	O utilizador ajusta o contraste a imagens.	4	Must	Completo
Req32	O utilizador roda imagens.	4	Must	Completo
Req35	O utilizador aplica um algoritmo de recorte automático de imagens com base no seu conteúdo.	4	Must	Completo

Requisitos Não Funcionais:

Requisito	Descrição	Contexto	Prioridade	Estado
RNF5	A página de um projeto distingue visualmente entre as ferramentas básicas e avançadas.	Usabilidade	Must	Completo
RNF7	O utilizador cria um projeto implicitamente ao arrastar ficheiros para o dashboard da aplicação.	Usabilidade	Must	Completo
RNF8	O utilizador carrega imagens arquivadas num único ficheiro .zip.	Usabilidade	Must	Completo
RNF9	O utilizador é mantido informado acerca do estado de processamento de um projeto em tempo real.	Usabilidade	Must	Incompleto
RNF14	A visualização de imagens grandes ou pequenas é auxiliada pelo utilitário zoom.	Usabilidade	Must	Incompleto
RNF15	A visualização de imagens permite que se navegue em todas as duas direções arrastando o rato.	Usabilidade	Must	Incompleto

Requisito	Descrição	Contexto	Prioridade	Estado
RNF18	A aplicação é compatível com diferentes plataformas e browsers, incluindo de dispositivos móveis e desktop	Usabilidade	Must	Incompleto
RNF19	A aplicação fornece feedback visual claro e imediato ao utilizador em caso de erro ou falha durante um procedimento demorado.	Usabilidade	Should	Completo?
RNF22	O sistema deve processar até 100 imagens ao mesmo tempo, sem quebras percetíveis no desempenho.	Escalabilidade e Elasticidade	Must	Incompleto
RNF23	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.	Escalabilidade e Elasticidade	Must	Incompleto
RNF25	A aplicação deve ser integrável com outras plataformas e serviços de terceiros.	Extensibilidade	Must	Completo
RNF28	A aplicação deve ser facilmente estendida com novas ferramentas de edição.	Extensibilidade	Must	Completo
RNF32	O sistema deve ser projetado para facilitar a execução de testes.	Manutenção	Must	Ambíguo
RNF33	A aplicação deve realizar backups automáticos dos dados e imagens dos utilizadores.	Manutenção	Should	Incompleto

2.2 Arquitetura do sistema

Em relação à Arquitetura do Sistema, tentamos segui-la o máximo possível, mas houve certos pontos onde nos diferenciamos:

- Em relação ao componente de Stripe, que iria tratar da gestão financeira, deixamos essa implementação para a responsabilidade de outra equipa, de forma a podermos nos focar nos requisitos mais importantes e essenciais para o funcionamento da aplicação.
- Quanto à autenticação, consideramos desenvolver uma API específica com cookies para tratar da autenticação, mas com o número de tarefas a aumentar e as implementações com Docker a complicar, decidimos-nos focar no que consideramos mais importante, ou seja, a

implementação e chamada dos Containers que contêm os microsserviços. Tendo feito isto, a autenticação acabou por ser feita fazendo apenas um POST à BD, se existir algum utilizador com as mesmas credenciais.

2.3 Implementação

2.3.1 Mudanças em relação à solução arquitetural proposta

- **Falta da implementação do Stripe:** Como um dos nossos principais objetivos desta fase foi implementar a complicada comunicação entre os diferentes microsserviços implementados, acabamos por deixar a implementação da parte financeira da aplicação para outra equipa.
- **Utilizadores não são questionados com a opção de pagar a subscrição Premium durante o registo:** Um dos requisitos com o qual acabamos por discordar mais foi com a opção dos utilizadores poderem comprar a Subscrição Premium durante o registo da conta, visto que achamos que era um pouco invasivo e "direto" à carteira do utilizador, o que não dá muito boa imagem. Devido a isso, deixamos a opção de mudança de subscrições apenas quando o utilizador já se registou e encontra-se autenticado.
- **Mudança dos tipos de modelos das entidades da Base de Dados:** Em relação a este tópico, como decidimos utilizar MongoDB como o nosso serviço de Base de Dados ou, em outras palavras, um serviço de Bases de Dados Não-Relacionais, como os modelos sugeridos na solução arquitetural seguiam a ideologia de uma Base de Dados Relacional, nós acabamos por adaptar os modelos da melhor forma à nossa implementação, fazendo mudanças a modelos existentes e deixando alguns de lado também.

2.3.2 Endpoints

De forma a estabelecer as comunicações entre microsserviços e API do Backend, todos os endpoints nas páginas no Frontend comunicam primeiro com a API Gateway, que irá redirecionar os pedidos para os microsserviços necessários para a resolução do pedido.

De forma a clarificar a lógica das comunicações entre os diferentes componentes, segue-se abaixo uma explicação breve de todos os endpoints desenvolvidos na nossa implementação.

Para clarificar também que, de forma a diferenciar cada um dos microsserviços, utilizamos as seguintes portas para cada Microsserviço:

4000 - API Gateway com WS Gateway

4001 - Microsserviço de Utilizadores

4002 - Microsserviço de Projetos

4003 - Microsserviço de Subscrições

2.3.2.1 Login

- **axios.get('http://localhost:4000/login')** : Este endpoint envia o pedido de GET para o API Gateway, que vai verificar na BD se existe um user com as informações de login. Caso exista, o user é enviado para a página e armazenado no LocalStorage.

2.3.2.2 Edição do Projeto

- **axios.post('http://localhost:4000/addproject', storedProject)** : Este endpoint é responsável por adicionar o projeto armazenado no LocalStorage, ou seja, o projeto atualmente a ser editado na página, à BD, enviando um pedido POST até ao API Gateway.
- **axios.post('http://localhost:4000/user/:userid/project/:project_id/process')** : Este endpoint é o responsável por estabelecer a comunicação com o WS Gateway e começar o desencadeamento das ferramentas selecionadas nas imagem selecionada do projeto.

2.3.2.3 Página de Projetos

- **axios.get(`http://localhost:4000/projects/${userID}`)** : Este endpoint é responsável por ir buscar à Base de Dados todos os projetos do Utilizador que está autenticado na página. Isto é feito, outra vez, através de uma comunicação até ao API Gateway e, daí, até ao Microsserviço de Projetos.
- **axios.post('http://localhost:4000/changeProjectShow',{id:this.projects[index]._id,})** : Relativamente a este endpoint, sempre que é necessário apagar um projeto da BD, sentimos que seria melhor estabelecer essa responsabilidade a um Gestor de Base de Dados da aplicação. Dito isto, de forma a seguir este raciocínio, atribuímos uma flag, que indica se o projeto é admitido para ser demonstrado ao utilizador ou não. Ou seja, quando um utilizador "apaga" um projeto, este vai ficar apenas escondido do utilizador, até o Gestor da Base de Dados apagar todas as entradas de projetos com esta flag.

2.3.2.4 Página de Registo

- **axios.post('http://localhost:4000/register', {name:this.profileData.name,email: this.profileData.email,password: this.profileData.password})** : Este endpoint, tal como se pode imaginar pela keyword, trata do registo de um utilizador, enviando a informação até ao microsserviço dos Utilizadores, que vai verificar se o utilizador já existe na BD ou não e, caso não exista, então o utilizador é registado e passa para a página de edição de projetos, com a sua conta autenticada.

2.3.2.5 Página de Subscrições

- **axios.post('http://localhost:4000/changeSubscription', {id: this.storedUser._id,subscription: this.storedUser.plan.type,state : this.storedUser.plan.state,date_payment : this.storedUser.plan.Date_payment});** : Finalmente, faltando apenas referenciar o microsserviço das subscrições, este endpoint é responsável pela mudança do plano do utilizador, dependendo do plano que ele seleciona, passando este a ter uma subscrição Premium Ativa, Anual ou Mensal.

É importante de apontar também que, de forma a haver a passagem de pedidos do API Gateway para os microsserviços necessários para a resolução destes pedidos, foram implementados mais endpoints no API Gateway, de forma a fazer a passagem necessária para os outros microsserviços, como podemos reparar que todos os endpoints explicados acima têm como destino o API Gateway. Feito isto, o API Gateway, mal possa, envia de volta para o Frontend a resposta enviada por um dos microsserviços.

2.3.3 Tecnologias Utilizadas

As tecnologias utilizadas no projeto, dividem-se em quatro áreas principais onde incluem diversas ferramentas e linguagens.

- **Backend**: No backend Utiliza-se *Python* nos microsserviços de ferramentas e JavaScript com Node.js para a API, WebSockets e microsserviços relacionados à gestão de utilizadores, projetos e subscrições.
- **Frontend**: Para o frontend Recorre-se ao framework *Vue.js* combinado com JavaScript e Vite, garantindo uma interface de utilizador eficiente.
- **Base de Dados**: A base de dados é gerida com *MongoDB*, utilizando Mongoose para facilitar o armazenamento de dados em formato não relacional.
- **Comunicação**: Na comunicação, RabbitMQ é usado para a gestão de mensagens, enquanto as APIs HTTP são implementadas com Express no backend e Axios no frontend para assegurar a integração entre a interface e os microsserviços de gestão. Além disso, os WebSockets permitem atualizações em tempo real do estado de execução de um projeto diretamente no frontend.

3 Microsserviços

Cada membro da equipa foi responsável pela implementação de um microsserviço. A distribuição foi organizada da seguinte forma:

Microserviço	**Respnsável**	**Descrição**
Ajuste de Brilho	pg57868	Permite modificar a intensidade luminosa de uma imagem, tornando-a mais clara ou mais escura. Ideal para corrigir imagens subexpostas ou superexpostas.
Ajuste de Contraste	pg57582	Altera o nível de contraste da imagem, enfatizando a diferença entre áreas claras e escuras. Útil para destacar detalhes em imagens ou para ajustá-las de acordo com preferências estéticas.
Borda de Imagem	pg55956	Adiciona bordas personalizadas às imagens, podendo ser utilizadas para criar margens decorativas ou destacar.
Rotação de Imagens	pg55992	Gira imagens em ângulos personalizados, permitindo correções de orientação ou ajustes criativos na composição.
Recorte de Imagens	pg55972	Permite selecionar e manter apenas uma área específica da imagem, removendo partes indesejadas.
Recorte Automático	pg57884	Utiliza algoritmos para identificar e recortar automaticamente áreas de interesse na imagem.
Redimensionamento	pg57565	Ajusta as dimensões da imagem para valores específicos, mantendo as proporções originais.
Extração de Texto	pg57549	Utilizando o Tesseract OCR, é possível identificar e extrair texto presente em imagens de forma eficiente. A classe TextExtractionTool exemplifica como combinar o Tesseract OCR com OpenCV para pré-processar imagens e melhorar a precisão da extração de texto.
Contagem de Pessoas	pg57885	Identificar e conta o número de pessoas em uma imagem. Ferramenta útil para análises em ambientes públicos ou eventos.

4 Problemas e implementações para o futuro

Apesar das ferramentas funcionarem, ao passar para um ambiente de Produção, problemas com a conexão à base de dados, e, principalmente, com conetividade entre Backend e Frontend (misturar http com https em produção), impediram a nossa capacidade de testar a aplicação de forma devida, problemas quais nos comprometemos a completar, corrigir e assegurar o bom funcionamento antes da defesa da aplicação.

Ora, de notar, que a conetividade entre Backend e Frontend poderá ser resolvida ao assegurar que o Backend também usa https, o MongoDB poderá funcionar se cada microsserviço tiver o seu próprio Container de base de dados.

5 Conclusão

A implementação do MVP do projeto pictuRAS representou um desafio significativo, mas permitiu o desenvolvimento de uma solução funcional que cumpre a maioria dos requisitos essenciais definidos. As principais funcionalidades, como a criação e gestão de projetos, carregamentos de imagens, aplicação de ferramentas de edição e descarregamento dos resultados, foram implementadas com sucesso. Adicionalmente, a integração de um microsserviço de watermark, conforme solicitado, demonstra a capacidade de estender a aplicação com novos serviços.

Apesar dos avanços, alguns requisitos funcionais e não funcionais não foram completamente implementados, como o suporte para visualização de imagens grandes com zoom ou a escalabilidade horizontal do sistema e, também, tal como se pode ver no capítulo 4, encontramos alguns problemas com a conexão à base de dados, quando testamos num ambiente de Produção, o que nos impediu de testar as principais funcionalidades no ambiente correto. Contudo, as escolhas estratégicas realizadas, como priorizar a comunicação entre microsserviços e a integração dos principais componentes, foram determinantes para garantir a entrega de um produto utilizável nesta fase.

Por fim, este projeto estabelece uma base sólida para futuras melhorias e ampliações, permitindo a evolução do pictuRAS para uma solução ainda mais completa e eficiente.