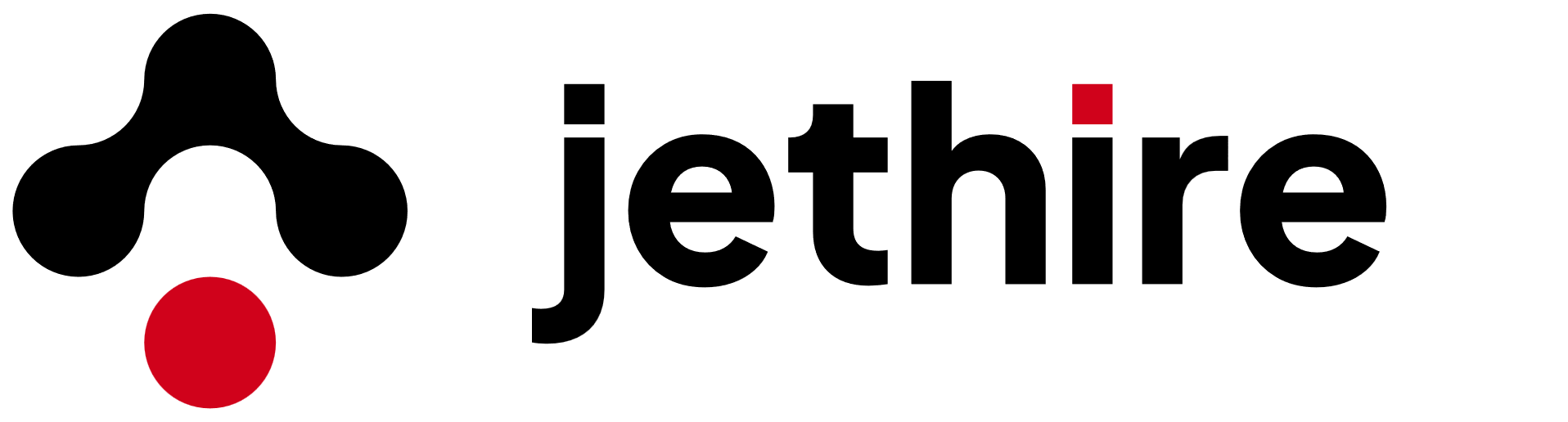


Técnico de Gestão e Programação de Sistemas Informáticos

Prova de Aptidão Profissional

**Jet Hire**



Cantanhede

Tomé Almeida, 12º TGPSI, Nº 15

2022 / 2025



Escola Técnico Profissional de Cantanhede

Técnico de Gestão e Programação de Sistemas Informáticos

Prova de Aptidão Profissional: Jet Hire

Equipa de Acompanhamento: Michael Teixeira, Elisabete Cavaleiro e Ana Marques

Cantanhede

2024/2025

CRIATIVIDADE É SÓ CONECTAR AS COISAS.

***Steve Jobs***

**Agradecimentos**

Gostaria de agradecer em primeiro lugar ao meu coordenador de curso, professor

Michael Teixeira, que se mostrou sempre disponível para me esclarecer dúvidas e que

me orientou nas diversas decisões que precisei de tomar ao longo da realização do

curso e da respetiva prova de aptidão profissional. Vejo nele uma inspiração e um

modelo a seguir.

**Índice Geral**

[Introdução (Português) 1](#_Toc200363092)

[Introduction (Inglês) 3](#_Toc200363093)

[Planeamento 5](#_Toc200363094)

[Fundamentação 5](#_Toc200363095)

[Recursos 6](#_Toc200363096)

[Hardware 6](#_Toc200363097)

[Software 6](#_Toc200363098)

[Metodologia 8](#_Toc200363099)

[Atividades 9](#_Toc200363100)

[Fase de Definição 9](#_Toc200363101)

[Fase de Desenvolvimento 9](#_Toc200363102)

[Fase de Manutenção 10](#_Toc200363103)

[Cronograma 10](#_Toc200363104)

[Instalações 11](#_Toc200363105)

[Visual Studio Code 11](#_Toc200363106)

[MongoDB Compass 12](#_Toc200363107)

[Node.js (node / npm / npx) 13](#_Toc200363108)

[Github Desktop 14](#_Toc200363109)

[Inkscape 16](#_Toc200363110)

[Krita 17](#_Toc200363111)

[Postman 18](#_Toc200363112)

[Sincronização entre dois computadores 20](#_Toc200363113)

[Criação do repositório 20](#_Toc200363114)

[Push 21](#_Toc200363115)

[Pull 23](#_Toc200363116)

[Sitemap 24](#_Toc200363117)

[Projeto 26](#_Toc200363118)

[Elementos gerais 26](#_Toc200363119)

[Download e preparação do template 26](#_Toc200363120)

[Instalação de dependências 26](#_Toc200363121)

[Execução do projeto 26](#_Toc200363122)

[Página principal 27](#_Toc200363123)

[Dashboard 27](#_Toc200363124)

[Conclusão 28](#_Toc200363125)

[Webgrafia 29](#_Toc200363126)

**Índice de Figuras**

[Figura 1 - Metodologia Kanban 8](#_Toc200106004)

[Figura 2 - Cronograma 10](#_Toc200106005)

[Figura 3 - Instalação Visual Studio Code: Passo 1 11](#_Toc200106006)

[Figura 4 - Instalação Visual Studio Code: Passo 2 11](#_Toc200106007)

[Figura 5 - Extensões 12](#_Toc200106008)

[Figura 6 - Instalação MongoDB Compass: Passo 1 12](#_Toc200106009)

[Figura 7 - Instalação MongoDB Compass: Passo 2 13](#_Toc200106010)

[Figura 8 - sudo apt install nodejs 13](#_Toc200106011)

[Figura 9 - sudo apt install npm 13](#_Toc200106012)

[Figura 10 - Verificação do npm 14](#_Toc200106013)

[Figura 11 - Instalação GitHub Desktop: Passo 1 15](#_Toc200106014)

[Figura 12 - Instalação GitHub Desktop: Passo 2 15](#_Toc200106015)

[Figura 13 - Instalação GitHub Desktop: Passo 3 16](#_Toc200106016)

[Figura 14 - Instalação do Inkscape: Passo 1 16](#_Toc200106017)

[Figura 15 - Instalação do Inkscape: Passo 2 17](#_Toc200106018)

[Figura 16 - Instalação do Krita: Passo 1 18](#_Toc200106019)

[Figura 17 - Instalação do Krita: Passo 2 18](#_Toc200106020)

[Figura 18 - Instalação do Postman: Parte 1 19](#_Toc200106021)

[Figura 19 - Instalação do Postman: Parte 2 19](#_Toc200106022)

[Figura 20 - Instalação do Postman: Parte 3 20](#_Toc200106023)

[Figura 21 - Sincronização entre dois computadores: Passo 1 21](#_Toc200106024)

[Figura 22 - Sincronização entre dois computadores: Passo 2 21](#_Toc200106025)

[Figura 23 - Sincronização entre dois computadores: Passo 3 22](#_Toc200106026)

[Figura 24 - Sincronização entre dois computadores: Passo 4 22](#_Toc200106027)

[Figura 25 - Sincronização entre dois computadores: Passo 5 23](#_Toc200106028)

[Figura 26 - Sincronização entre dois computadores: Passo 6 23](#_Toc200106029)

[Figura 27 - Sincronização entre dois computadores: Passo 7 24](#_Toc200106030)

[Figura 28 - Sincronização entre dois computadores: Passo 8 24](#_Toc200106031)

[Figura 29 – Sitemap principal 25](#_Toc200106032)

[Figura 30 - Download do template 26](#_Toc200106033)

[Figura 31 - Instalação dos node\_modules 26](#_Toc200106034)

[Figura 32 - npm run dev 27](#_Toc200106035)

[Figura 33 - Projeto executado 27](#_Toc200106036)

[Figura 34 - Logótipo 28](#_Toc200106037)

[Figura 35 - Escolha da cor principal 29](#_Toc200106038)

[Figura 36 - Escolha do tom claro 29](#_Toc200106039)

[Figura 37 - Escolha do tom escuro 29](#_Toc200106040)

[Figura 38 - Criação de um gradiente claro 30](#_Toc200106041)

[Figura 39 - Criação de um gradiente escuro 30](#_Toc200106042)

[Figura 40 - Vermelho 31](#_Toc200106043)

[Figura 41 - Amarelo 31](#_Toc200106044)

[Figura 42 - Azul 31](#_Toc200106045)

[Figura 43 - Verde 31](#_Toc200106046)

[Figura 44 - Cores neutras 32](#_Toc200106047)

[Figura 45 - Exemplo de variáveis globais 32](#_Toc200106048)

[Figura 46 - Rodapé inicial 33](#_Toc200106049)

[Figura 47 - Rodapé final 33](#_Toc200106050)

[Figura 48 - Barra de navegação inicial 33](#_Toc200106051)

[Figura 49 - Barra de navegação final 33](#_Toc200106052)

[Figura 50 - Destaque principal inicial 34](#_Toc200106053)

[Figura 51 - Destaque principal final 34](#_Toc200106054)

[Figura 52 - Slider de categorias inicial 37](#_Toc200106055)

[Figura 53 - Slider de categorias final 37](#_Toc200106056)

[Figura 54 - Como funciona inicial 38](#_Toc200106057)

[Figura 55 - Como funciona final 38](#_Toc200106058)

[Figura 56 - Ofertas mais recentes inicial 39](#_Toc200106059)

[Figura 57 - Ofertas mais recentes final 39](#_Toc200106060)

[Figura 58 - Procurar trabalho inicial 40](#_Toc200106061)

[Figura 59 - Procurar trabalho final 40](#_Toc200106062)

[Figura 60 - Candidatos recentes inicial 41](#_Toc200106063)

[Figura 61 - Candidatos recentes final 41](#_Toc200106064)

[Figura 62 - Cria o teu perfil inicial 41](#_Toc200106065)

[Figura 63 - Cria o teu perfil final 42](#_Toc200106066)

[Figura 64 - Cabeçalho da página de ofertas inicial 42](#_Toc200106067)

[Figura 65 - Cabeçalho da página de ofertas final 42](#_Toc200106068)

[Figura 66 - Filtro avançado: Área 43](#_Toc200106069)

[Figura 67 - Filtro avançado: Salário 43](#_Toc200106070)

[Figura 68 - Filtro avançado: Modalidade 43](#_Toc200106071)

[Figura 69 - Filtro avançado: Experiência 43](#_Toc200106072)

[Figura 70 - Filtro avançado: Oferta Publicada 43](#_Toc200106073)

[Figura 71 - Filtro avançado: Tipo de Trabalho 43](#_Toc200106074)

[Figura 72 - Oferta de emprego inicial 44](#_Toc200106075)

[Figura 73 - Oferta de emprego final 44](#_Toc200106076)

[Figura 74 - Apresentação de empresas inicial 44](#_Toc200106077)

[Figura 75 - Apresentação de empresas final 45](#_Toc200106078)

[Figura 76 - Página de início de sessão inicial 45](#_Toc200106079)

[Figura 77 - Página de início de sessão final 46](#_Toc200106080)

[Figura 78 - Página de registo inicial 46](#_Toc200106081)

[Figura 79 - Página de registo final 47](#_Toc200106082)

[Figura 80 - Banner inicial 47](#_Toc200106083)

[Figura 81 - Banner final 48](#_Toc200106084)

[Figura 82 - Descrição final 48](#_Toc200106085)

[Figura 83 - Competências iniciais 48](#_Toc200106086)

[Figura 84 - Competências finais 49](#_Toc200106087)

[Figura 85 - Experiência inicial 49](#_Toc200106088)

[Figura 86 - Experiência final 49](#_Toc200106089)

[Figura 87 - Educação inicial 50](#_Toc200106090)

[Figura 88 - Educação final 50](#_Toc200106091)

[Figura 89 - Portfólio normal 50](#_Toc200106092)

[Figura 90 - Portfólio com hover 50](#_Toc200106093)

[Figura 91 - Barra lateral inicial 51](#_Toc200106094)

[Figura 92 - Barra lateral final 51](#_Toc200106095)

[Figura 93 - Primeira secção de edição de perfil 52](#_Toc200106096)

[Figura 94 - Segunda secção de edição de perfil 52](#_Toc200106097)

[Figura 95 - Terceira secção da edição de perfil 53](#_Toc200106098)

[Figura 96 - Quarta secção do registo de perfil 55](#_Toc200106099)

[Figura 97 - Exemplo de adição de dados no registo de perfil 60](#_Toc200106100)

[Figura 98 - Portfólio na edição de perfil 61](#_Toc200106101)

[Figura 99 - Exemplo de portfólio na edição de perfil 61](#_Toc200106102)

[Figura 100 - Quinta secção do registo de perfil 61](#_Toc200106103)

[Figura 101 - Página inicial MongoDB 63](#_Toc200106104)

[Figura 102 - Criação de projeto no MongoDB 63](#_Toc200106105)

[Figura 103 - Criação de um cluster no MongoDB 64](#_Toc200106106)

[Figura 104 - Criação de administrador no MongoDB 65](#_Toc200106107)

[Figura 105 - Escolha de aplicação no MongoDB 65](#_Toc200106108)

[Figura 106 - Ligação ao cluster no MongoDB 66](#_Toc200106109)

[Figura 107 - Página inicial MongoDB Compass 67](#_Toc200106110)

[Figura 108 - Interface do MongoDB Compass 67](#_Toc200106111)

[Figura 109 - Dependências Mongoose 68](#_Toc200106112)

[Figura 110 - Exemplo de erro no registo 77](#_Toc200106113)

[Figura 111 - Criação de projeto na GCC 80](#_Toc200106114)

[Figura 112 - Dashboard da GCC 81](#_Toc200106115)

[Figura 113 - Criação do token OAuth: Parte 1 81](#_Toc200106116)

[Figura 114 - Criação do token OAuth: Parte 2 82](#_Toc200106117)

[Figura 115 - Criação do token OAuth: Parte 3 82](#_Toc200106118)

[Figura 116 - Criação do token OAuth: Parte 4 83](#_Toc200106119)

[Figura 117 - Criação do token OAuth: Parte 5 83](#_Toc200106120)

[Figura 118 - Criação do token OAuth: Parte 6 84](#_Toc200106121)

[Figura 119 - Barra de navegação sem sessão 90](#_Toc200106122)

[Figura 120 - Barra de navegação com sessão 90](#_Toc200106123)

[Figura 121 - Verificação de perfil completa 95](#_Toc200106124)

[Figura 122 - Verificação de perfil por concluir 95](#_Toc200106125)

[Figura 123 - Exemplo de perfil por preencher 95](#_Toc200106126)

[Figura 124 - Exemplo de perfil completo 96](#_Toc200106127)

[Figura 125 - Criação de conta no Cloudinary 98](#_Toc200106128)

[Figura 126 - Página de "API Keys" 99](#_Toc200106129)

[Figura 127 - Exemplo de "A carregar candidatos" 104](#_Toc200106130)

[Figura 128 - Exemplo de contador de apresentação de utilizadores 105](#_Toc200106131)

[Figura 129 - Exemplo de "dropdown" na aplicação 106](#_Toc200106132)

[Figura 130 - Candidatos ordenados por "Mais Recente" 106](#_Toc200106133)

[Figura 131 - Candidatos ordenados por "Mais Antigo" 107](#_Toc200106134)

[Figura 132 - Candidatos a carregar 107](#_Toc200106135)

[Figura 133 - Sem resultados 108](#_Toc200106136)

[Figura 134 - Resultado da pesquisa de candidatos 108](#_Toc200106137)

[Figura 135 - Placeholder do formulário do registo de empresas 109](#_Toc200106138)

[Figura 136 - Formulário de registo de empresas 109](#_Toc200106139)

[Figura 137 - Cabeçalho da página de empresas 110](#_Toc200106140)

[Figura 138 - Secção 1 da barra lateral da página de empresas 110](#_Toc200106141)

[Figura 139 - Secção 2 da barra lateral da página de empresas 110](#_Toc200106142)

[Figura 140 - Apresentação das empresas 111](#_Toc200106143)

[Figura 141 - Página de início de sessão da dashboard inicial 119](#_Toc200106144)

[Figura 142 - Página de início de sessão da dashboard final 119](#_Toc200106145)

[Figura 143 - Exemplo de cartão de registo de empresa 123](#_Toc200106146)

[Figura 144 - Domínio verificado no Resend 124](#_Toc200106147)

[Figura 145 - API key do Resend 124](#_Toc200106148)

[Figura 146 - Exemplo de email de token de acesso 125](#_Toc200106149)

[Figura 147 - Interface de registo de empresa 131](#_Toc200106150)

[Figura 148 - Exemplo de email de registo de empresa 133](#_Toc200106151)

**Índice de Tabelas**

# Introdução (Português)

Para a minha Prova de Aptidão Profissional (PAP), escolhi desenvolver a “Jet Hire”, uma plataforma online para oferta e procura de emprego. A escolha deste projeto foi inspirada numa experiência pessoal que vivi durante as férias de verão. Na altura, precisei de procurar um trabalho para juntar dinheiro, mas encontrei várias dificuldades em encontrar vagas de forma prática e acessível. Isso fez-me perceber a importância de uma ferramenta que torne este processo mais simples tanto para quem procura como para quem oferece emprego.

Durante o desenvolvimento do projeto, optei por direcionar a plataforma apenas para a área de TI, visto que, é a área que frequento e que pretendo seguir e ao ter noção desta problemática no mundo real, decidi tornar este projeto uma espécie de solução para um problema atual.

A “Jet Hire” será uma aplicação prática e moderna, que permitirá a ligação entre pessoas à procura de trabalho e empresas com vagas disponíveis. Além disso, esta plataforma incluirá funcionalidades úteis, como a possibilidade de adicionar anúncios para trabalhos temporários, incluindo vagas de verão para jovens maiores de 16 anos, garantindo que possam candidatar-se com o apoio de um responsável.

A decisão de desenvolver este projeto com tecnologias como **Next.js**, **MongoDB**, **Bootstrap**, **Mongoose** e **NextAuth** deve-se ao facto de querer explorar ferramentas que são muito utilizadas no mercado atual. Para mim, esta é uma excelente oportunidade de consolidar os conhecimentos que adquiri ao longo do curso e ao mesmo tempo criar uma solução que pode ajudar outras pessoas na mesma situação que eu enfrentei.

A plataforma terá várias funcionalidades para os dois tipos principais de utilizadores. Para quem procura emprego, será possível criar um perfil detalhado com competências e experiências, publicar anúncios com os serviços que oferece e responder diretamente a vagas publicadas pelas empresas. Para as empresas, estas poderão criar anúncios de emprego e até formular questionários personalizados para facilitar o processo de seleção. Adicionalmente, será criado um sistema de backoffice para que moderadores possam supervisionar e garantir a segurança e a qualidade do conteúdo da plataforma.

O projeto será desenvolvido em três fases. Na primeira fase, fiz uma pesquisa detalhada sobre as ferramentas e funcionalidades que seriam necessárias para criar a aplicação. Depois disso, planeei as principais funcionalidades e desenhei *mockups* simples para organizar as ideias. Na segunda fase, comecei a construir o projeto, implementando as funções principais e corrigindo erros à medida que surgiam. Na terceira e última fase, ajustarei os detalhes finais, realizarei testes com utilizadores reais e recolherei feedback para melhorar a plataforma antes da sua entrega.

A “Jet Hire” não é apenas um projeto técnico, mas também um reflexo do meu crescimento ao longo do curso e da minha vontade de criar algo útil e inovador. Acredito que este trabalho não só mostra o que aprendi, mas também demonstra como pretendo usar esses conhecimentos no futuro.

# Introduction (Inglês)

For my Professional Aptitude Project (PAP), I decided to develop “Jet Hire,” an online platform for finding and offering jobs. The idea for this project came from a personal experience during the summer holidays. At that time, I needed to find a job to save money, but I had a hard time finding opportunities in an easy and practical way. This made me realise how important it is to have a tool that makes the process simpler for both job seekers and employers.

While working on the project, I chose to focus the platform on the IT area, as it is the field I am studying and want to pursue in the future. Knowing this is a real problem for many people, I decided to make this project a way to solve it.

“Jet Hire” will be a modern and easy-to-use application that connects people looking for jobs with companies offering positions. The platform will also include helpful features, such as allowing temporary job postings, including summer jobs for people aged 16 and older, with parental approval if needed.

I decided to develop this project using technologies like Next.js, MongoDB, Bootstrap, Mongoose, and NextAuth because these tools are widely used in the job market. For me, this is a great chance to practise what I have learned during my course while creating a solution that could help others who are in the same situation I was in.

The platform will have different features for its two main user groups. Job seekers can create detailed profiles with their skills and experience, post ads about the services they offer, and apply directly to job postings from companies.

Companies, on the other hand, will be able to create job postings and design customised questionnaires to help with the hiring process. A back-office system will also be created so that moderators can monitor the platform and ensure everything is running smoothly and safely.

The project will be developed in three main stages. In the first stage, I researched the tools and features needed for the application and planned the main functions. I also created simple mockups to organise my ideas. In the second stage, I started building the project, adding the main features and fixing any problems that came up. In the third and final stage, I will focus on making improvements, testing the platform with real users, and collecting feedback to make it even better before delivering it.

“Jet Hire” is not just a technical project. It also shows how much I have grown during my course and reflects my passion for creating something useful and innovative. I believe this project not only demonstrates what I have learned but also shows how I plan to use this knowledge in the future.

# Planeamento

## Fundamentação

Razões que levaram à escolha do tema.

|  |  |
| --- | --- |
| **Objetivos** | |
| **Gerais** | **Específicos** |
| Terminar o Curso | * Ter boa nota na PAP; * Terminar o curso com uma boa média. |
| Desenvolvimento da PAP | * Aumentar os meus conhecimentos em Programação; * Melhorar as minhas competências de concretização de projetos; * Adquirir conhecimentos na Linguagens (Quais?) * Aplicar os conhecimentos adquiridos ao longo dos três anos do curso. * Explorar novas tecnologias utilizadas para o desenvolvimento da PAP. * Coloca a aplicação na Play Store. * Colocar o site online para que seja visível por muitos utilizadores. * Desenvolver competências na área de desenvolvimento de projetos de software. |

## Recursos

### Hardware

Desktop pessoal

- AMD Ryzen 5 7600X @ 4.7Ghz

- RAM 32Gb DDR5 5600Mhz

- Radeon RX 6700 10Gb

- Disco SSD 512Gb

- Disco HDD 2Tb

- Linux Ubuntu 22.04.1

Portátil pessoal

- Apple Silicon M3 Chip

- RAM 16Gb

- Disco SSD 512Gb

- macOS Sequoia 15.01

### Software

- Microsoft Office 2021 Pro Plus

Requisitos Mínimos

- Sistema Operativo – Windows 10+ ou macOS 11.6+

- Processador – 1.6 GHz ou superior

- Memória RAM – 4Gb (64 bits)

- Disco – 4Gb

- Resolução Mínima – 1280px X 768px

- Visual Studio Code

Requisitos Mínimos

- Processador – 1.6 GHz ou superior

- Memória RAM – 1Gb

- Github Desktop

Requisitos Mínimos

- Sistema Operativo – Windows 10+ (64 bits) ou macOS 10.5+

- Arc

Requisitos Mínimos

- Sistema Operativo – Windows 10 19H1+ ou macOS 13+

- Google Chrome

Requisitos Mínimos

- Sistema Operativo – Windows 7+ ou macOS 10.9+

- Processador – 2 GHz ou superior

- Memória RAM – 2Gb

- Krita

Requisitos Mínimos

- Sistema Operativo – Windows 8.1+

- Processador – 1.6 GHz ou superior

- Memória RAM – 4Gb

- Disco – 1Gb

- MongoDB Compass

Requisitos Mínimos

- Sistema Operativo – Windows 10+ ou macOS 11+ ou Ubuntu 16.04+

## Metodologia

Para desenvolver a Jet Hire optei por usar a metodologia Kanban. Esta metodologia é muito visual e facilita o controlo da gestão de tarefas tanto como garante um fluxo contínuo do desenvolvimento.

A metodologia Kanban consiste na utilização de um quadro de tarefas para monitorizar o fluxo de trabalho. As tarefas são categorizadas e assim distribuídas respetivamente.

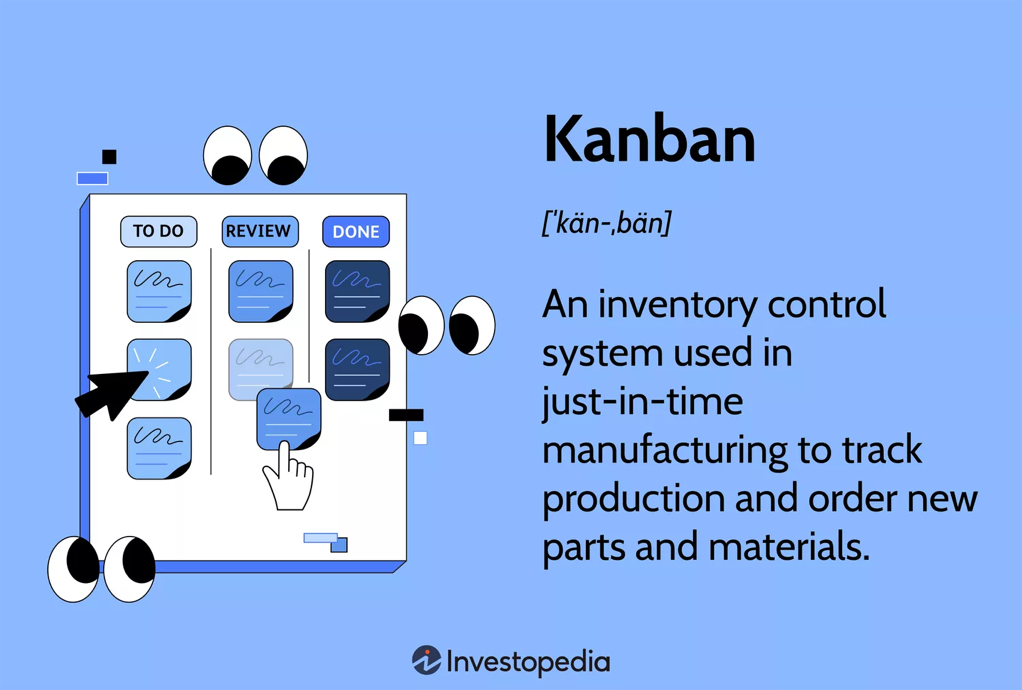


Figura 1 - Metodologia Kanban

*Fonte:* [*https://www.investopedia.com/terms/k/kanban.asp*](https://www.investopedia.com/terms/k/kanban.asp)

Escolhi esta metodologia porque ajuda a adaptar o trabalho conforme as necessidades mudam e dá uma visão clara do que se está a fazer, do que já se fez e do que se vai fazer. Como os requisitos da aplicação podem evoluir ao longo do tempo, o Kanban permite fazer estes ajustes de maneira fácil. Além disso, esta metodologia ajuda a identificar e resolver problemas rapidamente, garantindo que o projeto avança de forma organizada e eficiente.

## Atividades

A Prova de Aptidão Profissional irá ser desenvolvida em três fases. Cada fase é constituída por várias atividades. A primeira fase, a Fase de Definição tem como principais atividades: pesquisa bibliográfica, análise do sistema, análise dos requisitos e planeamento do projeto. A segunda fase, a Fase de Desenvolvimento é constituída pelas seguintes atividades, desenho, codificação e testes. A terceira e última fase, a Fase de Manutenção é onde encontramos as atividades de correção, adaptação e evolução.

### Fase de Definição

► Pesquisa Bibliográfica – etapa inicial do trabalho com o objetivo de reunir todas as informações necessárias para o desenvolvimento do projeto.

► Análise do Sistema – atividade onde se realizou o levantamento de todas as funcionalidades do projeto e a forma como elas vão funcionar. Nesta atividade foi também realizada a escolha das tecnologias para a realização do mesmo.

► Análise dos Requisitos – nesta atividade foi necessário fazer a verificação dos requisitos mínimos do Software necessário para implementação do projeto.

► Planeamento do Projeto – com esta atividade foi desenvolvido o cronograma em que se estabelece as balizas temporais para cada uma das fases do projeto.

### Fase de Desenvolvimento

► Desenho – nesta atividade faz-se o levantamento dos dados necessários para criação da Base de Dados utilizada no projeto, implementando o Diagrama Entidade Relacionamento e o Modelo de Dados. Faz-se ainda a estruturação da navegação do website e do backoffice.

► Codificação – atividade em que toda a parte de codificação é realizada.

► Testes – ao longo desta atividade realiza-se os testes às funcionalidades implementadas por forma a garantir que nesta fase as mesmas estão a funcionar de acordo com o que foi planeado.

### Fase de Manutenção

► Correção – é nesta atividade em que irei realizar correções ao projeto. Estas correções tanto podem ser à interface gráfica ou ao código realizado até ao momento. Se for necessário também se poderá fazer alterações estruturais à base de dados.

► Adaptação – depois de executado todos os testes e correções, é nesta atividade que vão ser realizadas adaptações ao projeto. Estas adaptações podem ser ao nível da interface gráfica ou ao nível da otimização de código. Estas adaptações surgem da necessidade do projeto ser testado por outras pessoas.

► Evolução – depois de ter os objetivos iniciais concluídos, é nesta fase, havendo tempo que irei tentar implementar novas funcionalidades que foram surgindo ao longo do desenvolvimento do projeto e que inicialmente não estavam previstas.

## Cronograma

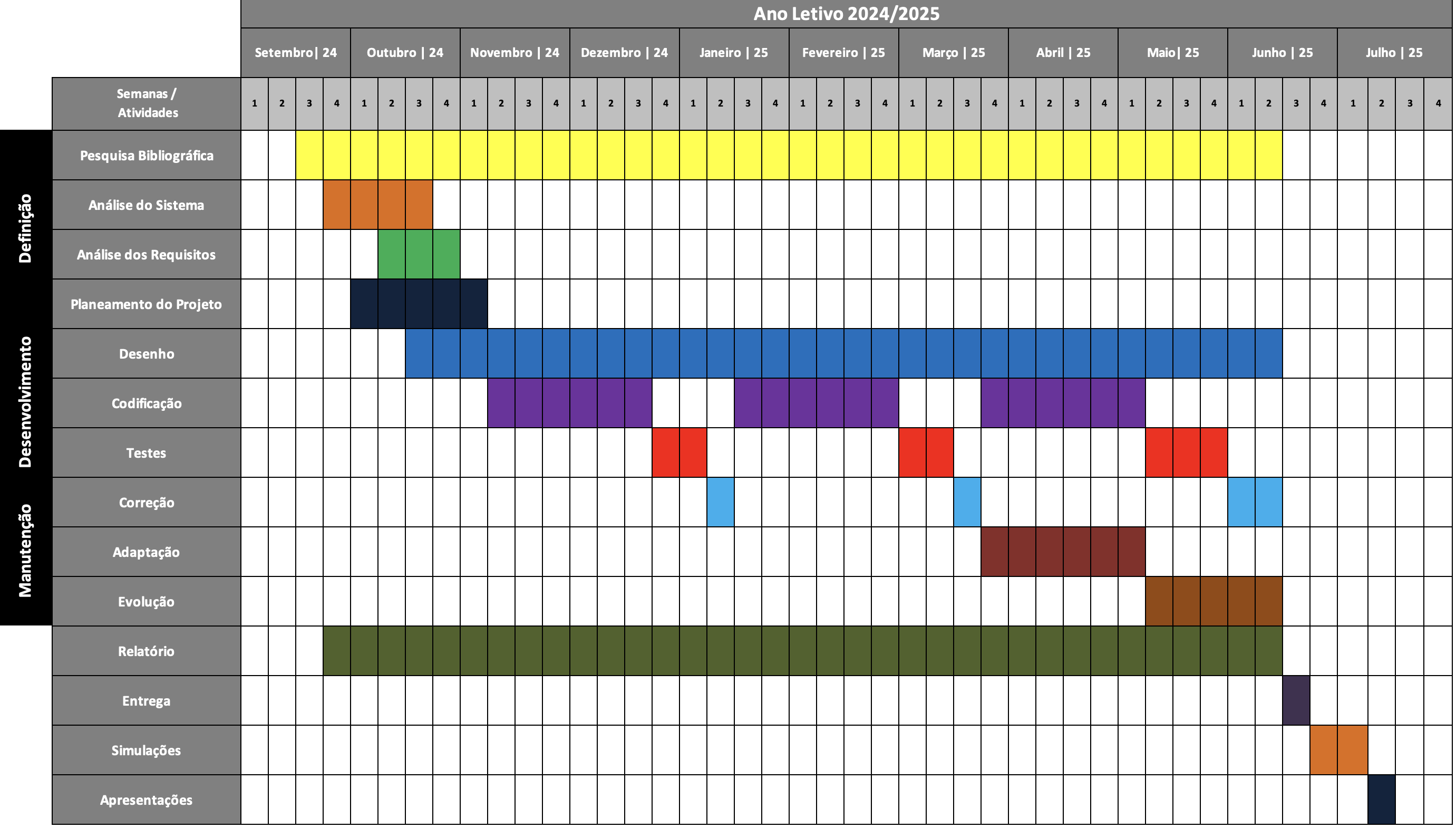


Figura 2 - Cronograma

## Instalações

### Visual Studio Code

Para começar a instalação do Visual Studio Code é preciso fazer *download* do instalador através do [site oficial](https://code.visualstudio.com/).

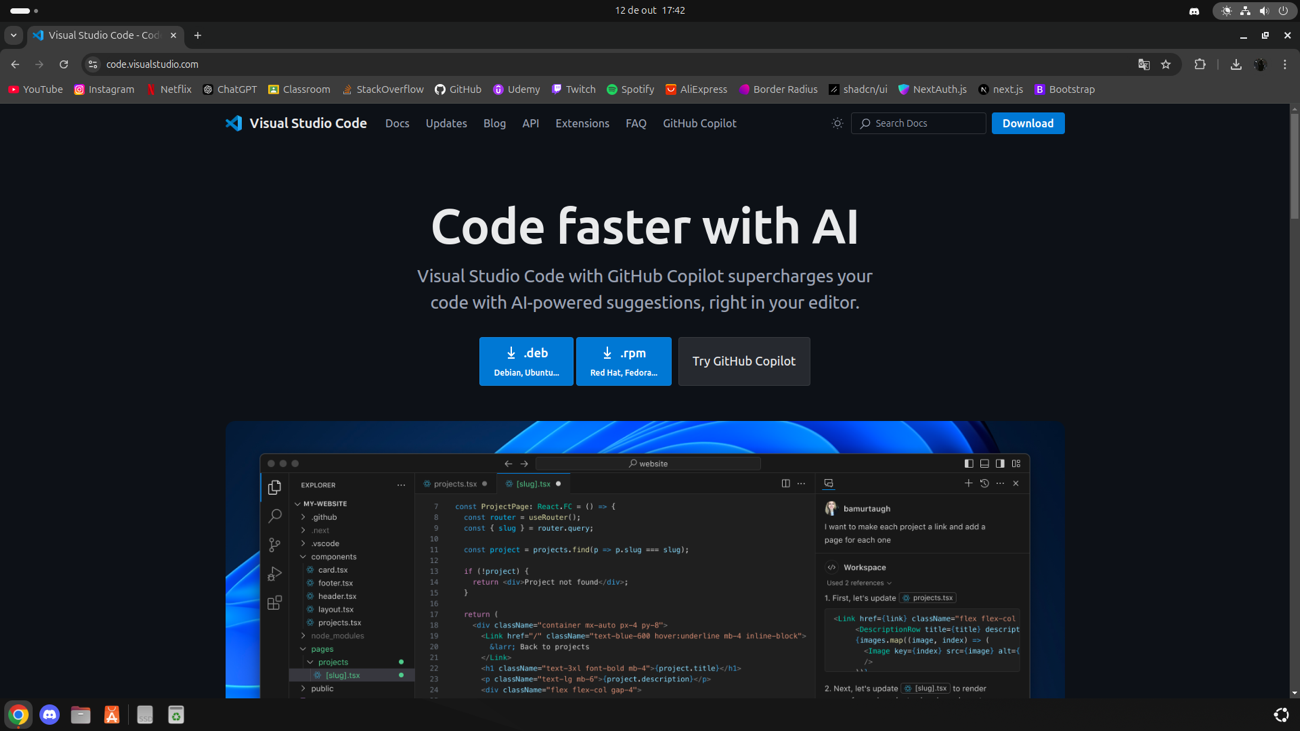


Figura 3 - Instalação Visual Studio Code: Passo 1

Depois de executar o instalador e a instalação estiver pronta, a página inicial abre.

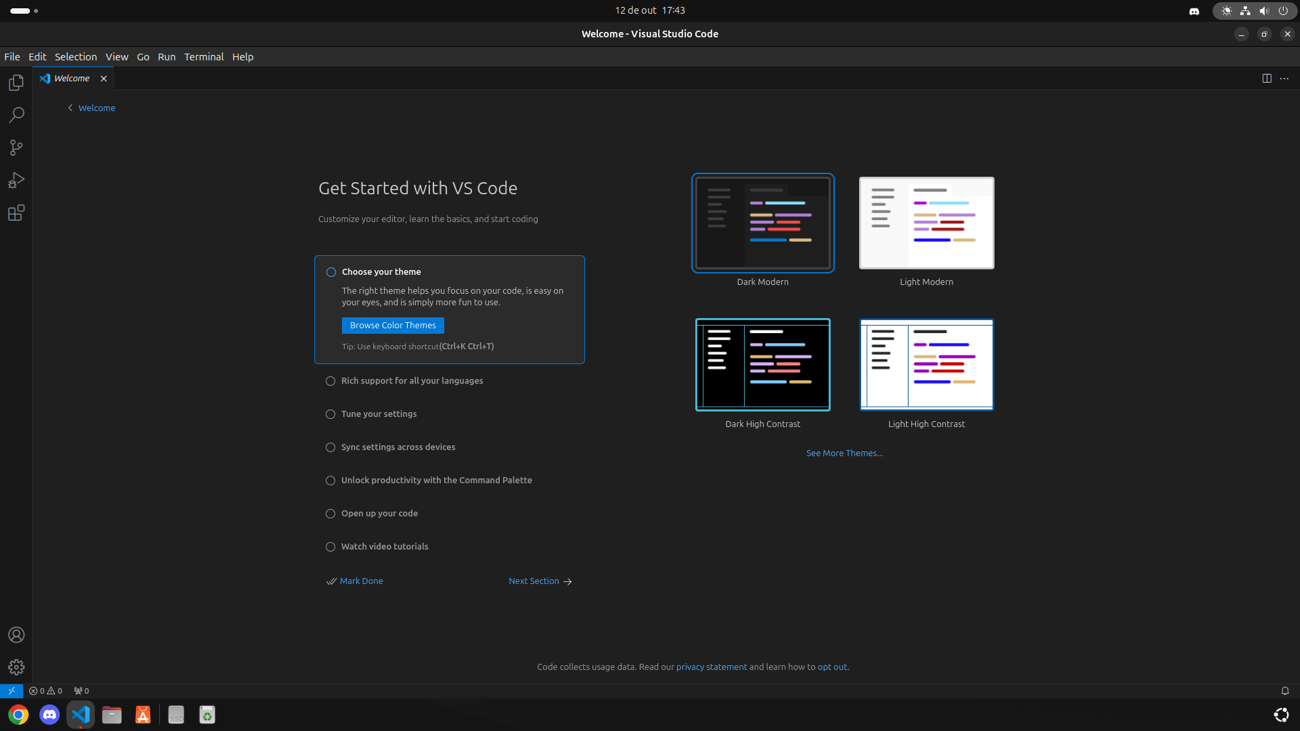


Figura 4 - Instalação Visual Studio Code: Passo 2

Para personalizar o Visual Studio Code a meu gosto e de acordo com as minhas necessidades adicionei as seguintes extensões:

|  |  |
| --- | --- |
| * **Auto Rename Tag** * **Error Lens** * **Min Theme** * **MongoDB for VS Code** * **Symbols** | Figura 5 - Extensões |

### MongoDB Compass

Tal como o VS Code é necessário fazer *download* do instalador através do [site oficial](https://www.mongodb.com/try/download/compass).

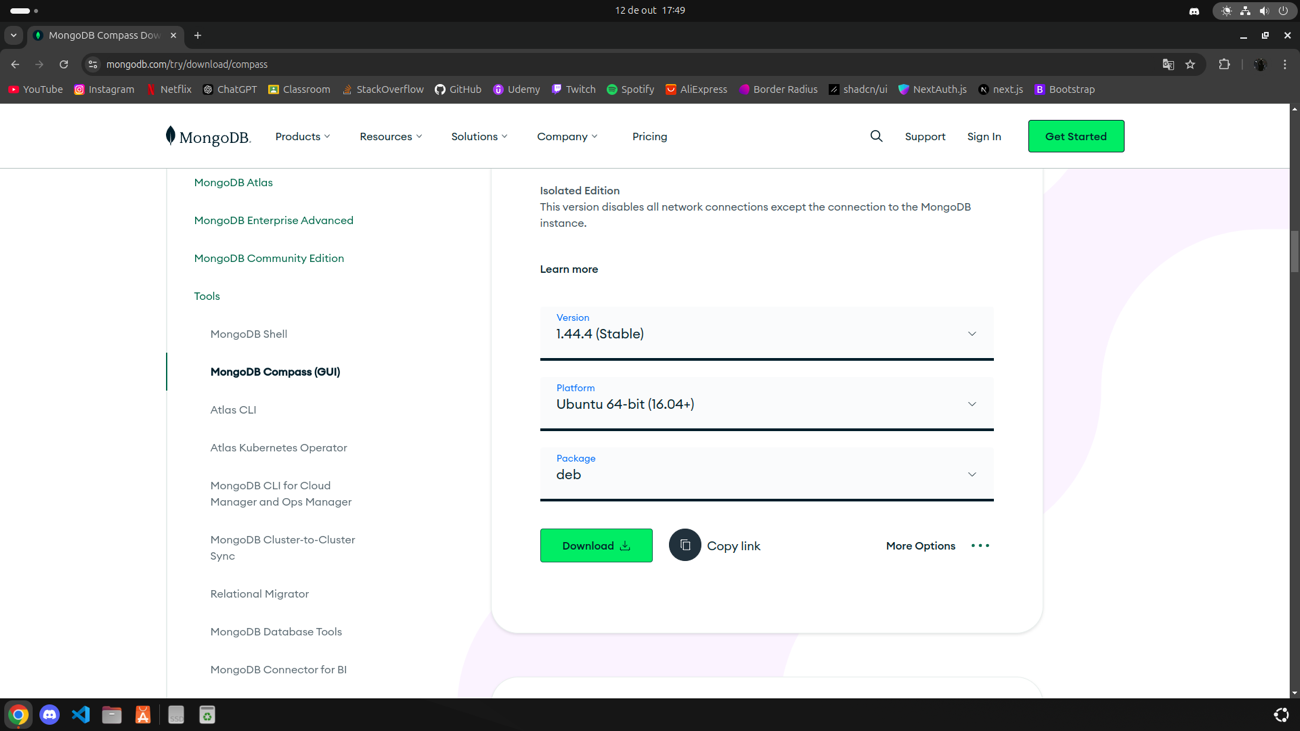


Figura 6 - Instalação MongoDB Compass: Passo 1

De seguida, é só executar o instalador.

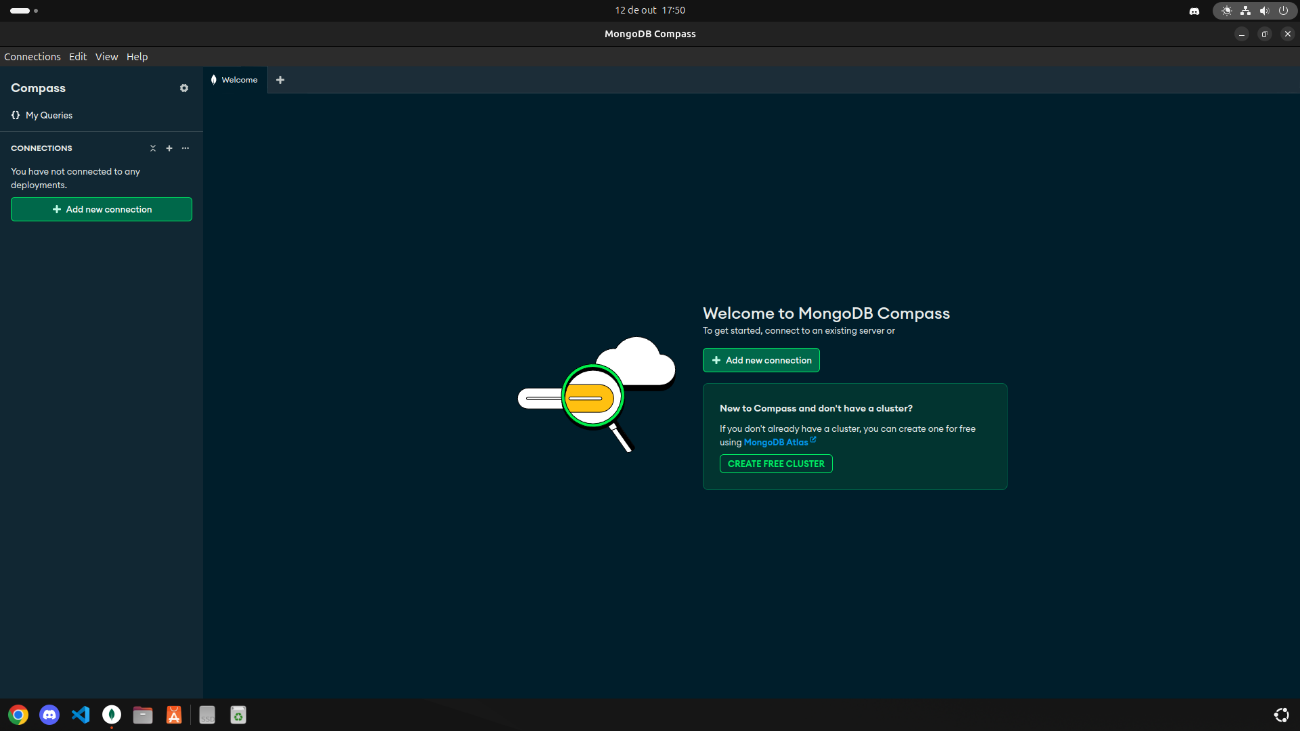


Figura 7 - Instalação MongoDB Compass: Passo 2

### Node.js (node / npm / npx)

No terminal do Ubuntu apenas é preciso utilizar os seguintes comandos para instalar o node e o npm:

|  |  |
| --- | --- |
| ***sudo apt install nodejs*** | ***sudo apt install npm*** |
| Figura 8 - sudo apt install nodejs | Figura 9 - sudo apt install npm |

Para verificar se foi instalado corretamente executei os comandos para verificar a respetiva versão:

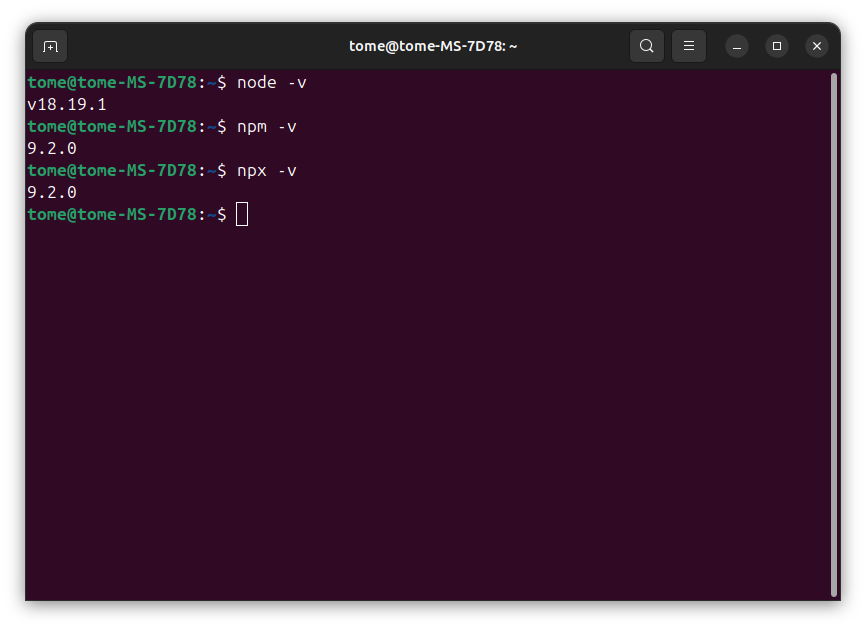


Figura 10 - Verificação do npm

### Github Desktop

Para instalar o Github Desktop no Ubuntu é necessário fazer *download* através do terminal, visto que não existe instalador no site oficial. Para isso é preciso utilizar o seguinte comando:

*wget https://github.com/shiftkey/desktop/releases/download/release-3.1.7-linux1/GitHubDesktop-linux-3.1.7-linux1.deb*

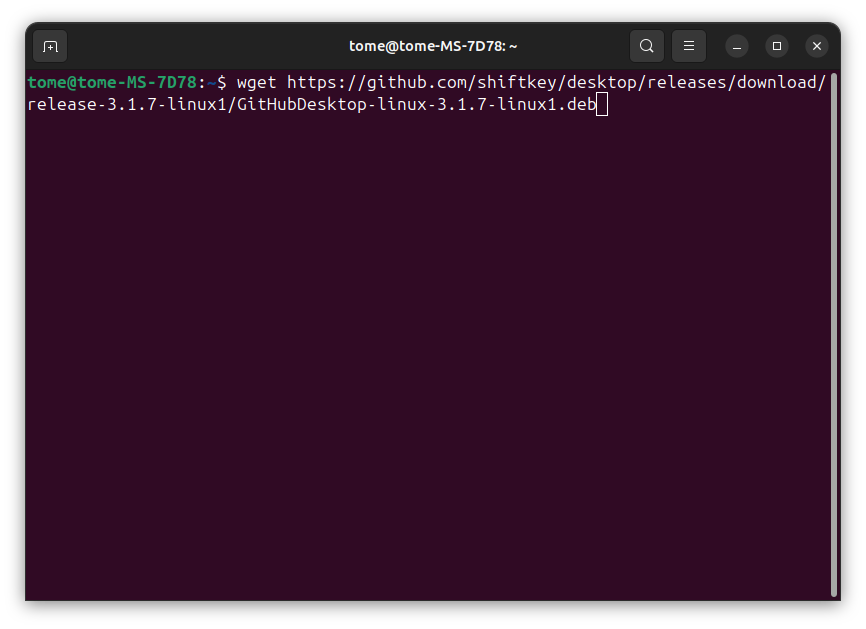


Figura 11 - Instalação GitHub Desktop: Passo 1

De seguida, é só executar o instalador com o comando:

*sudo apt install -f ./GitHubDesktop-linux-3.1.7-linux1.deb*

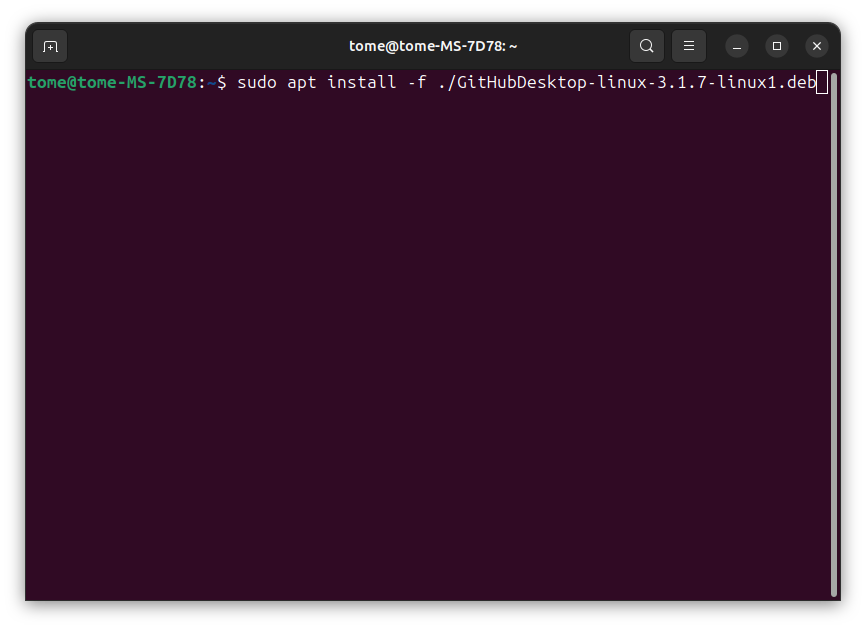


Figura 12 - Instalação GitHub Desktop: Passo 2

No fim da execução do instalador só falta abrir a aplicação.

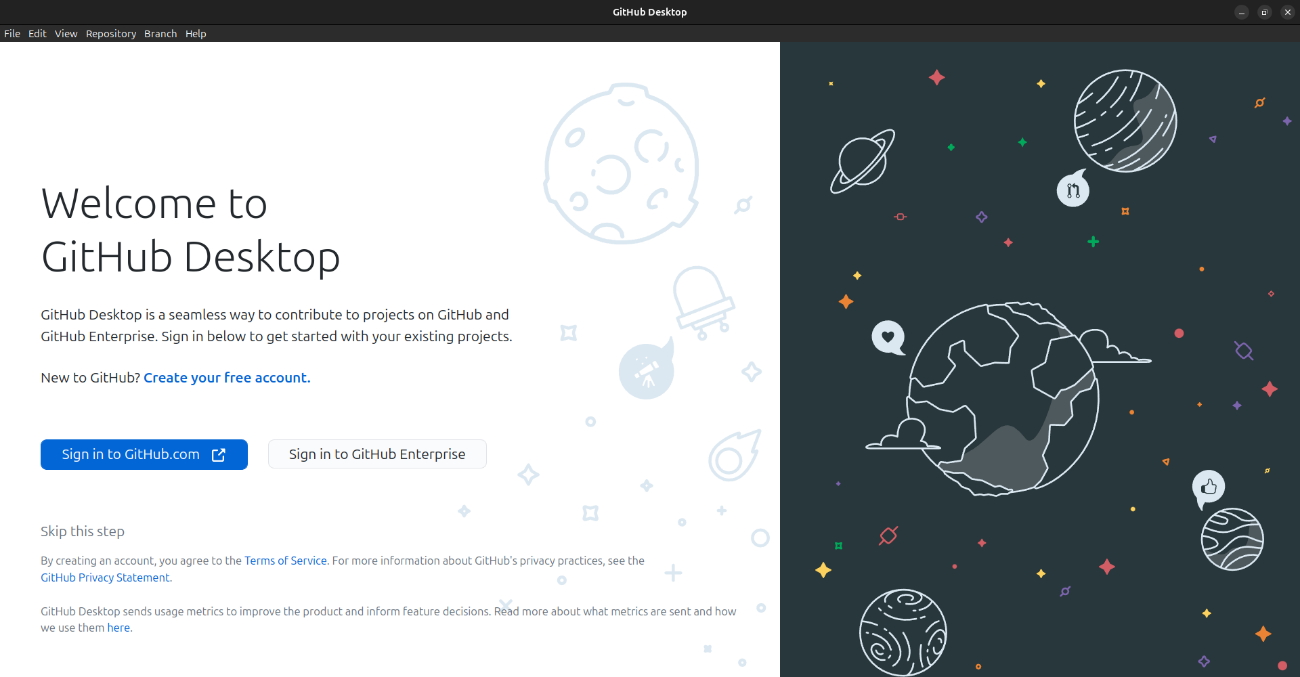


Figura 13 - Instalação GitHub Desktop: Passo 3

### Inkscape

Para a instalação do Inkscape só foi preciso utilizar o comando *sudo apt install inkscape* no terminal.

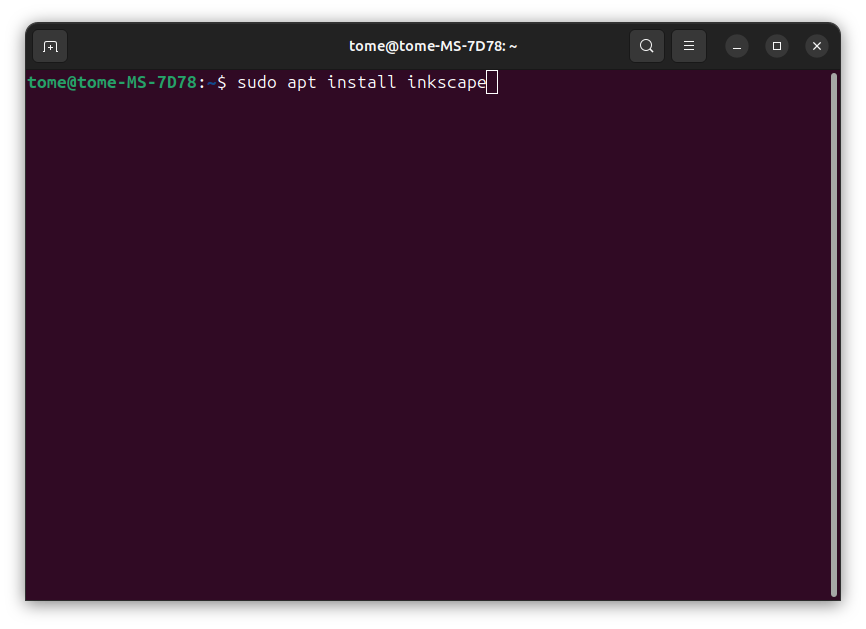


Figura 14 - Instalação do Inkscape: Passo 1

A instalação está concluída e pronta a ser utilizada.

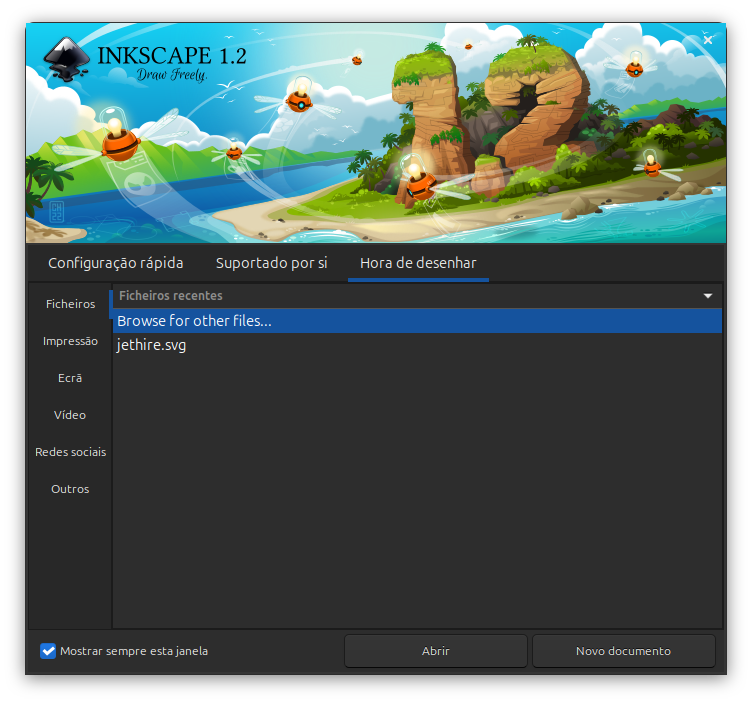


Figura 15 - Instalação do Inkscape: Passo 2

### Krita

Como o Krita está disponível no Centro de Aplicações do Ubuntu apenas foi preciso instalar através do mesmo.

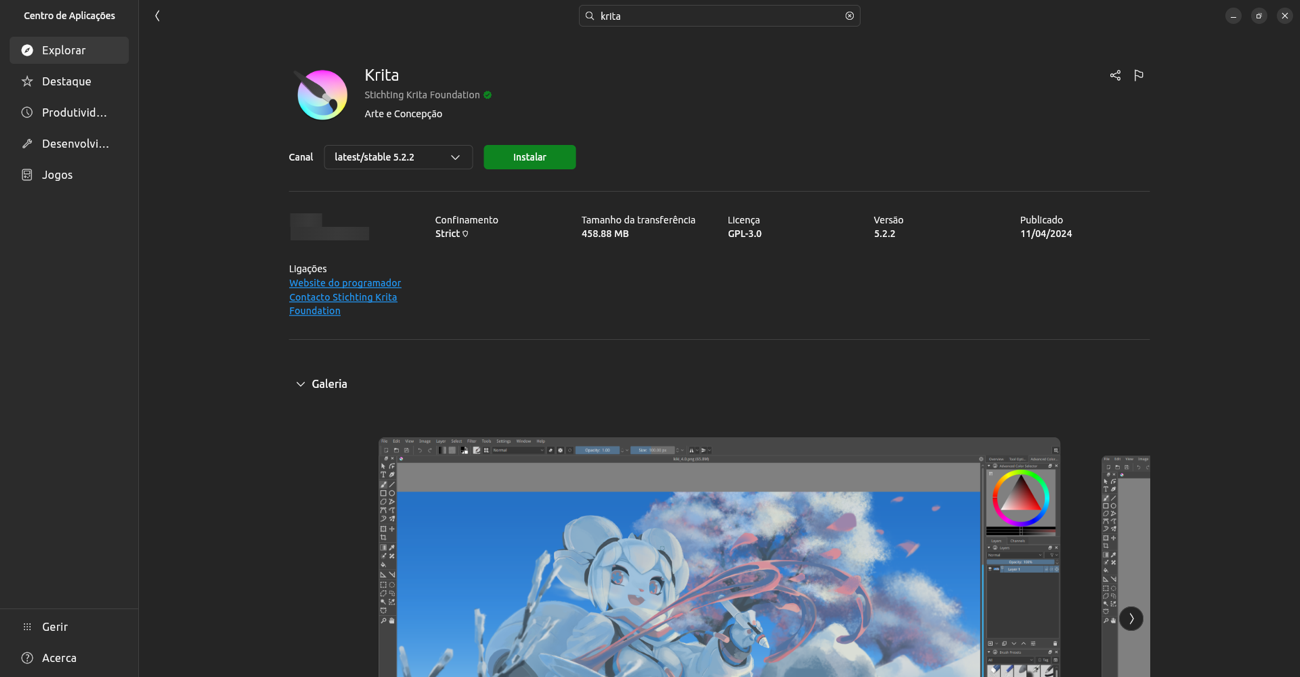


Figura 16 - Instalação do Krita: Passo 1

A aplicação está pronta a ser utilizada.

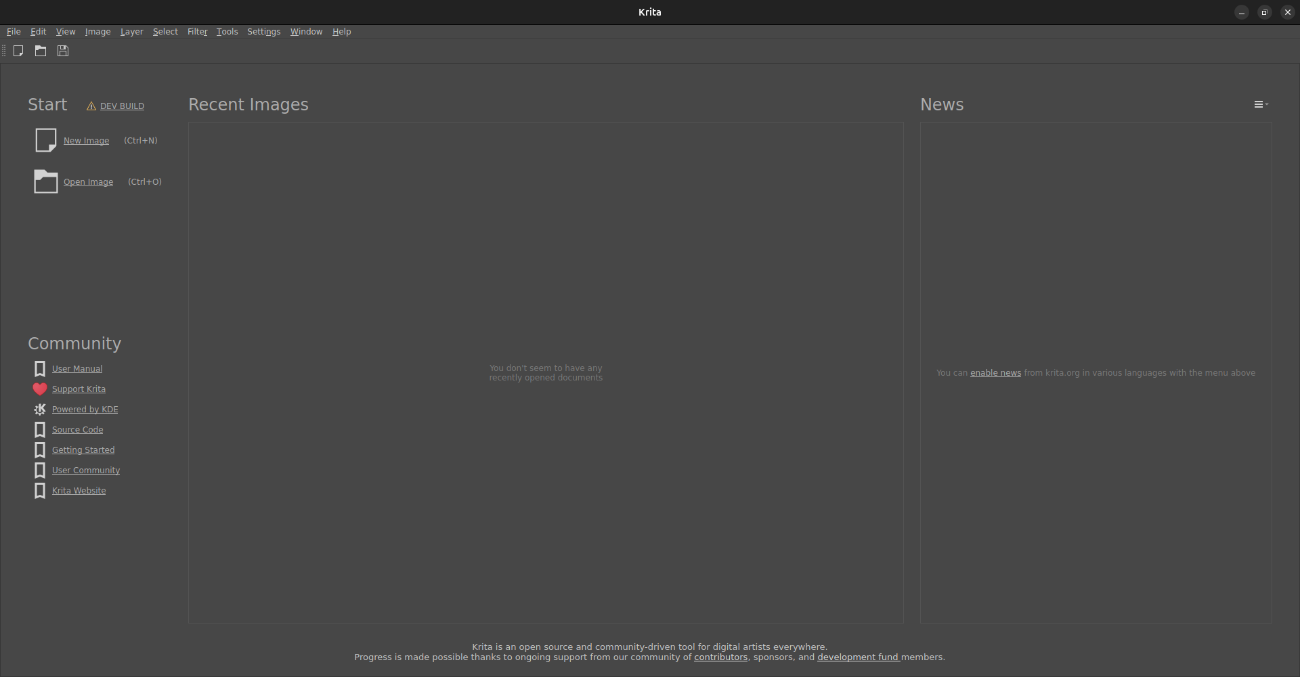


Figura 17 - Instalação do Krita: Passo 2

### Postman

Para instalar o Postman, apenas é necessário transferir o executável do site oficial da empresa.

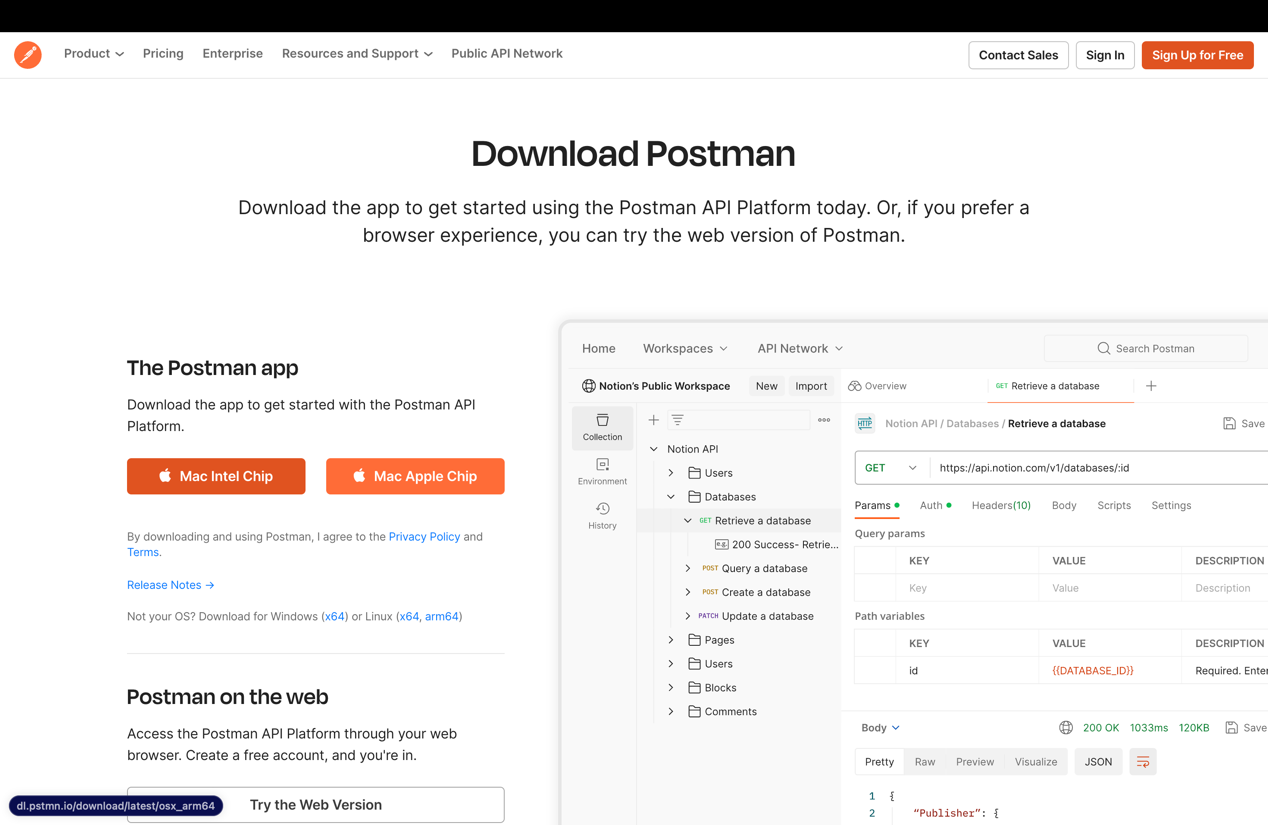


Figura 18 - Instalação do Postman: Parte 1

Após a instalação do executável, apenas é preciso criar conta e o Postman está pronto a ser utilizado.

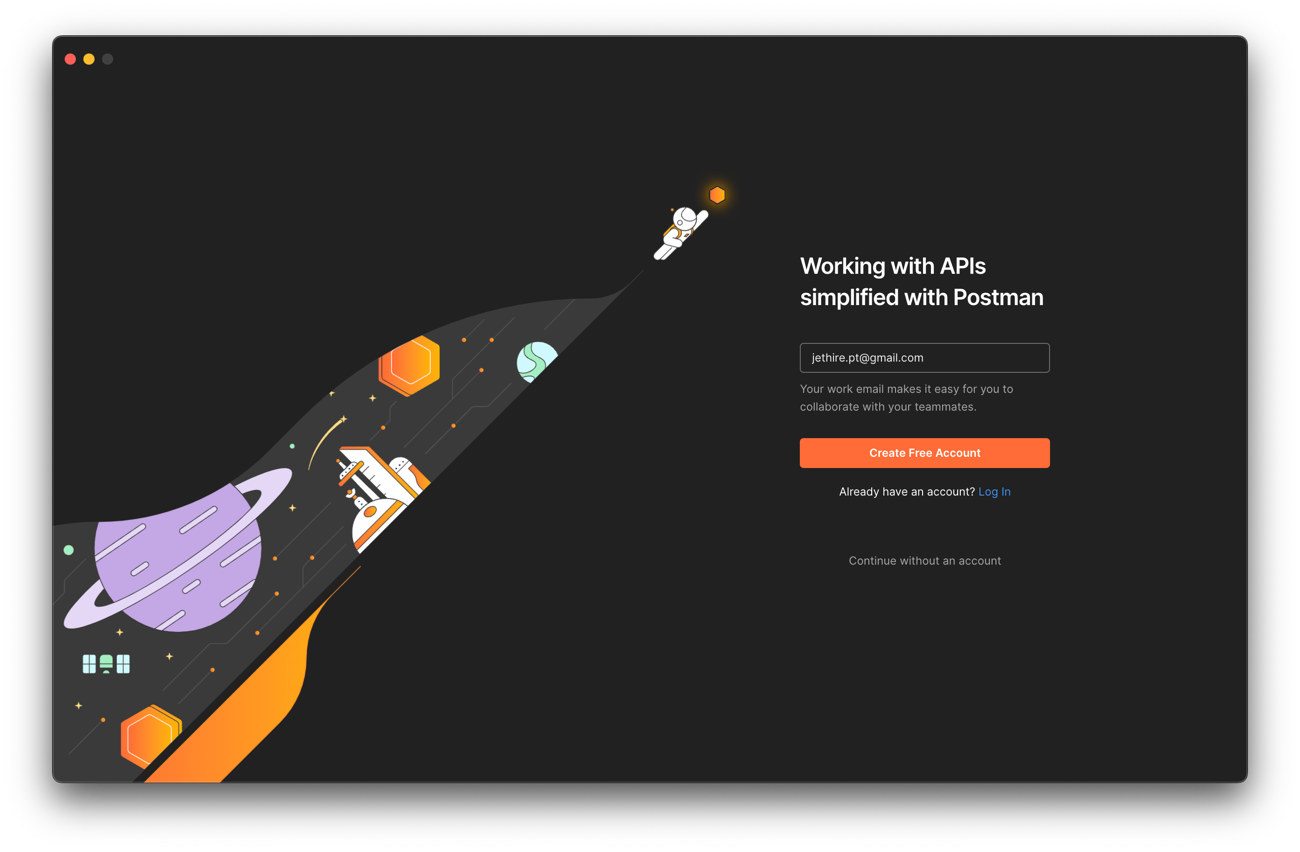


Figura 19 - Instalação do Postman: Parte 2

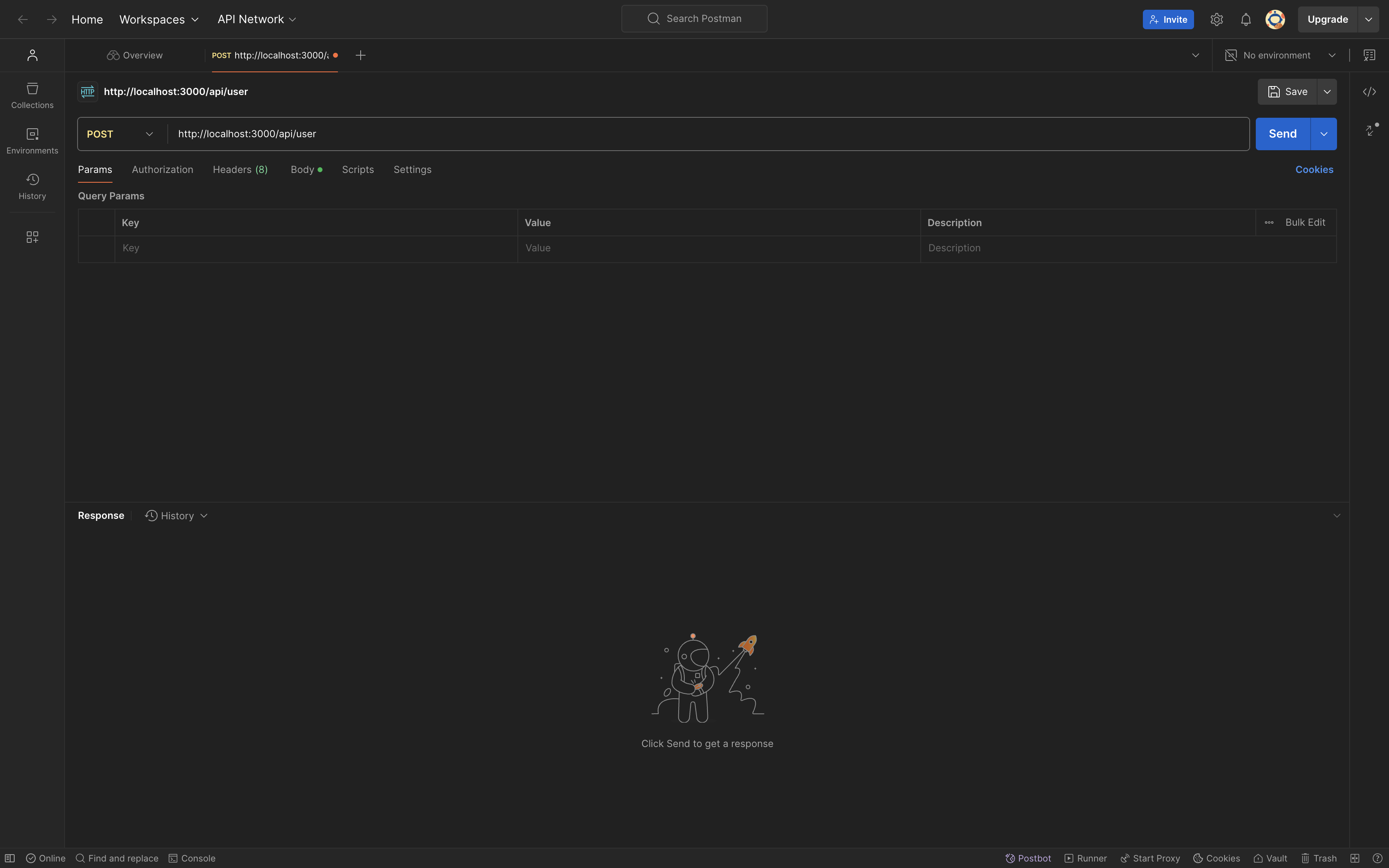


Figura 20 - Instalação do Postman: Parte 3

## Sincronização entre dois computadores

Como utilizei dois computadores para a realização projeto utilizei o GitHub Desktop para fazer esta sincronização.

### Criação do repositório

Para começar, é preciso criar o respetivo repositório onde se vai guardar todos os ficheiros:

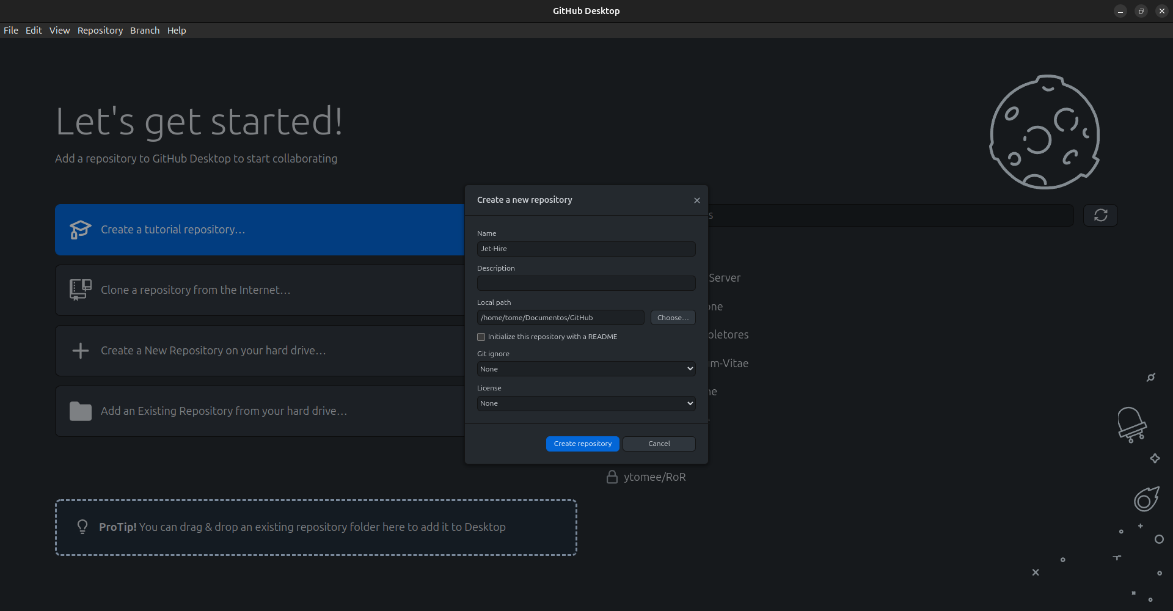


Figura 21 - Sincronização entre dois computadores: Passo 1

Depois da criação do repositório é preciso publicá-lo:

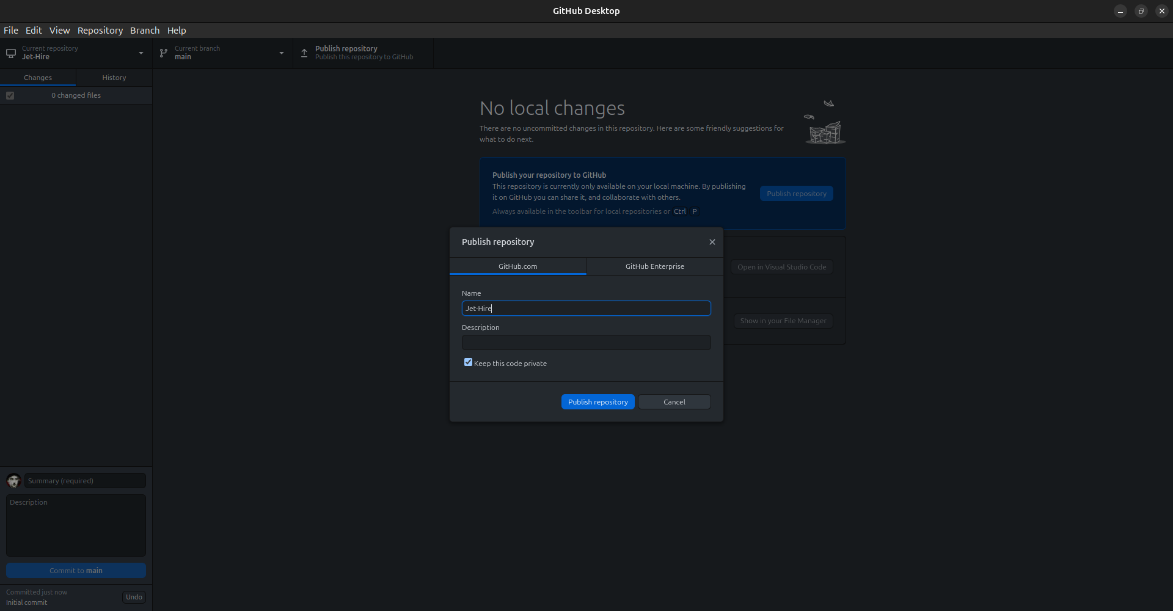


Figura 22 - Sincronização entre dois computadores: Passo 2

### Push

O Github vai criar uma pasta dentro da pasta Documentos do computador e é só preciso passar os ficheiros para a mesma. De seguida, volta-se á aplicação e verifica-se as alterações. No fim, dá-se *commit*, que vai fazer todas as alterações na *cloud.*

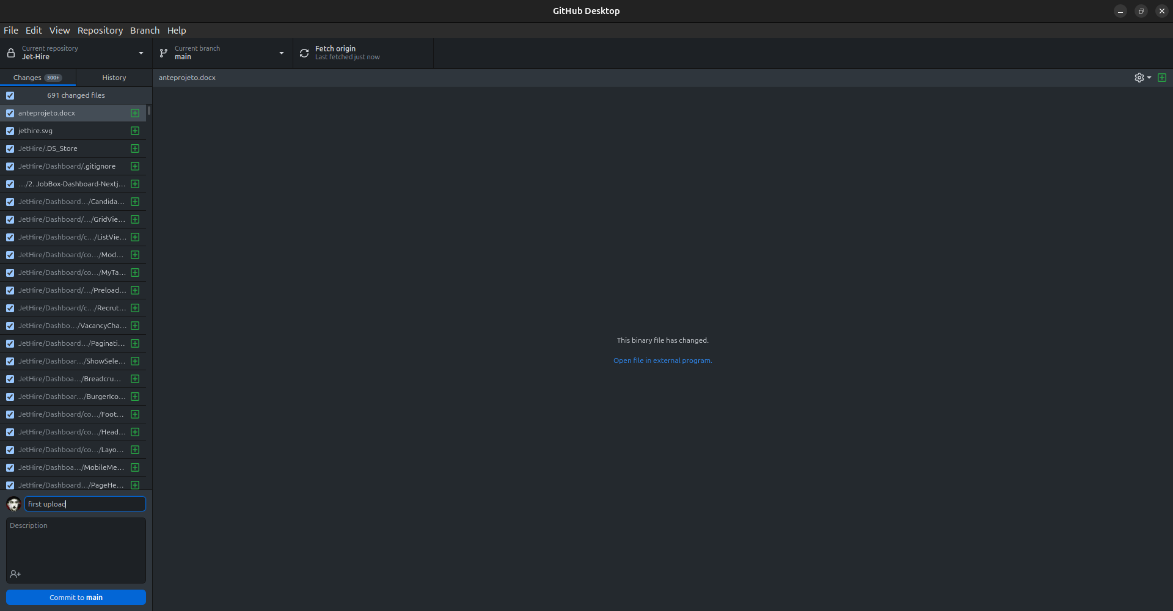
****

Figura 23 - Sincronização entre dois computadores: Passo 3

Depois de dar *commit*, é preciso dar *push*, ou seja, o pedido final para a alteração dos dados na *cloud*:

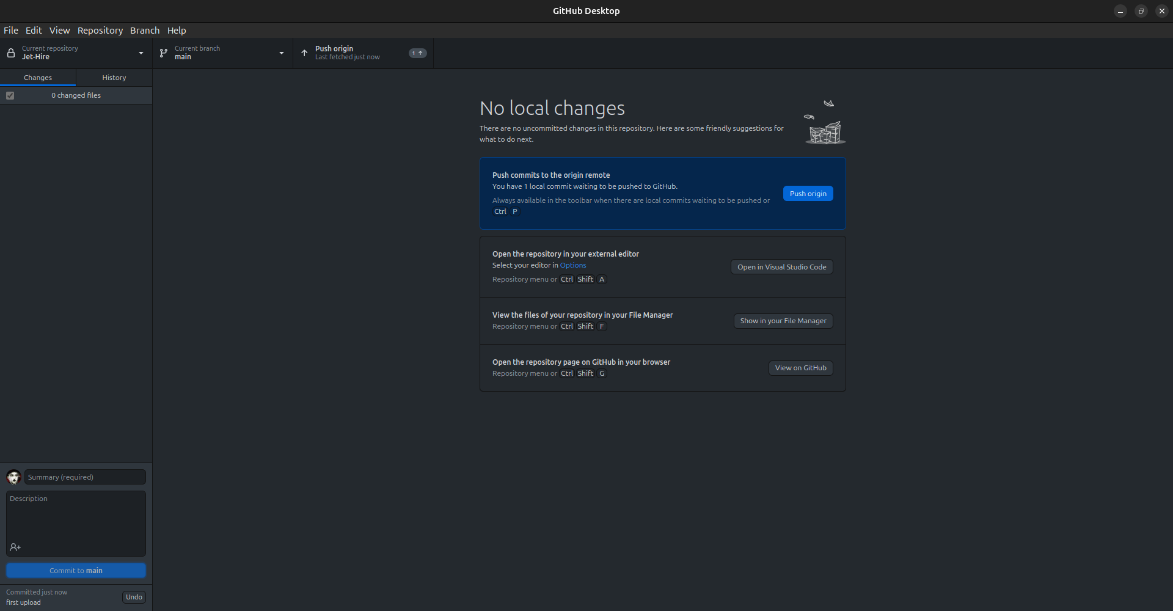


Figura 24 - Sincronização entre dois computadores: Passo 4

Para confirmar se o repositório foi criado corretamente pode-se confirmar pelo browser:

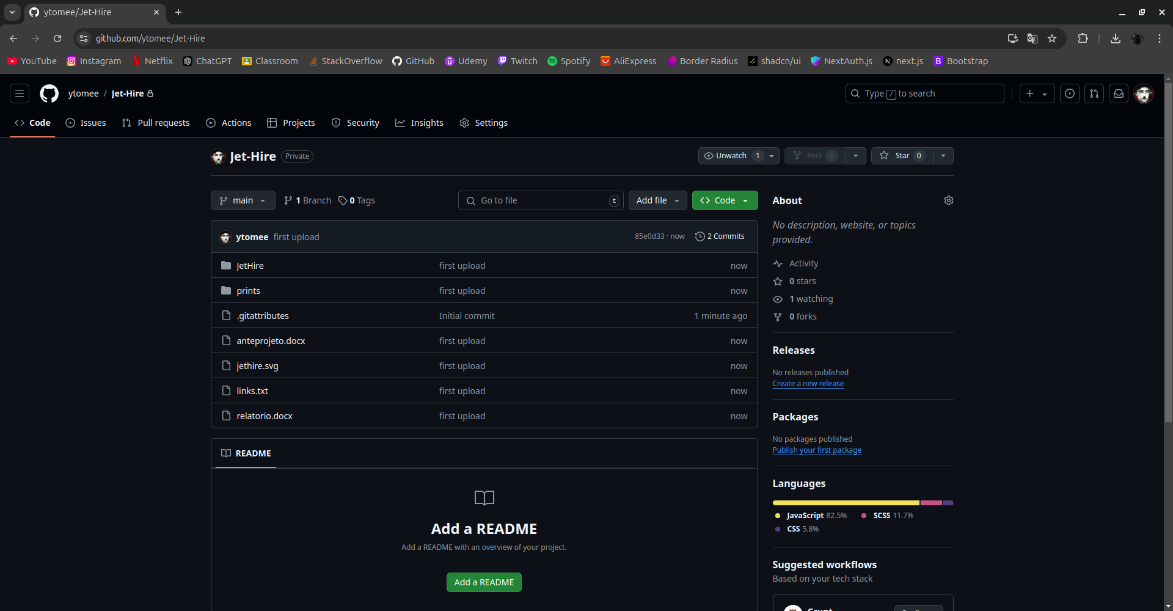


Figura 25 - Sincronização entre dois computadores: Passo 5

### Pull

Caso o repositório ainda não exista no computador, tem que se clonar o mesmo.

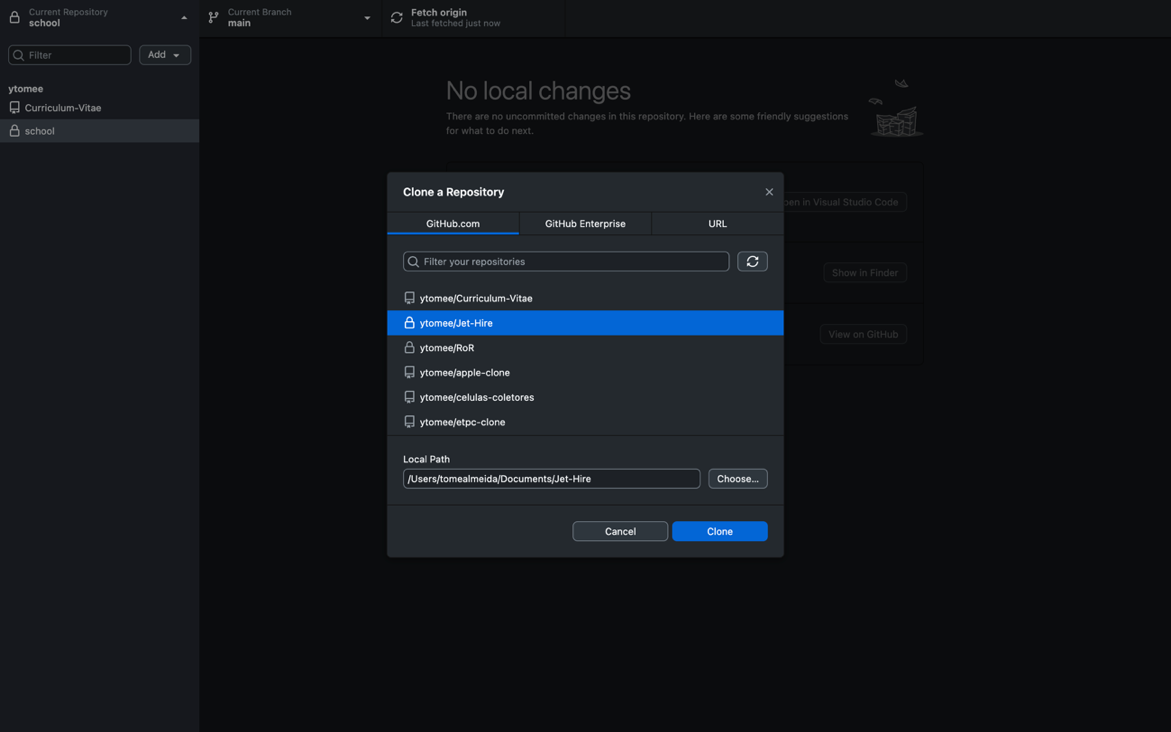


Figura 26 - Sincronização entre dois computadores: Passo 6

Se o repositório já existir, dá-se *fetch* para verificar se existem alterações na pasta, caso existam, dá-se *push* para voltar a alterar os dados:

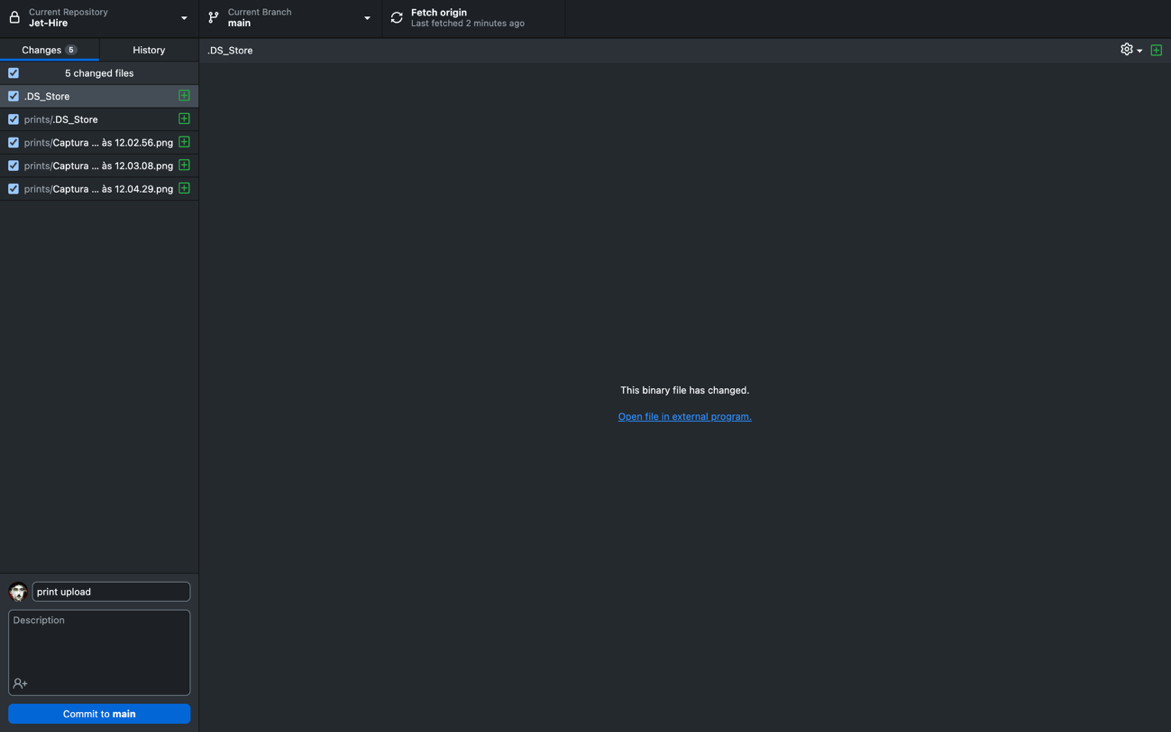


Figura 27 - Sincronização entre dois computadores: Passo 7

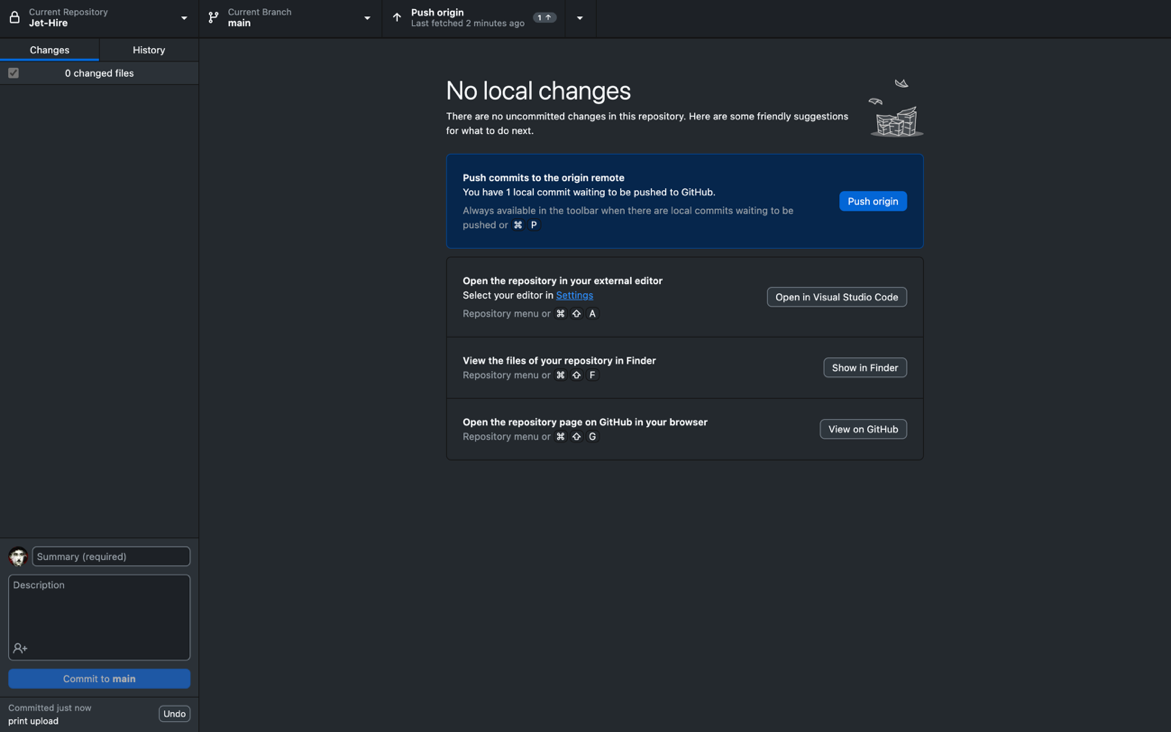


Figura 28 - Sincronização entre dois computadores: Passo 8

## Sitemap

Para o desenho do projeto decidi criar um *sitemap* na aplicação web FlowMapp. Optei por não criar um modelo DER visto que não vou utilizar uma base de dados relacional. Para uma melhor organização, optei por criar dois *sitemaps* separados, um para as páginas principais e outro para a *dashboard* geral.

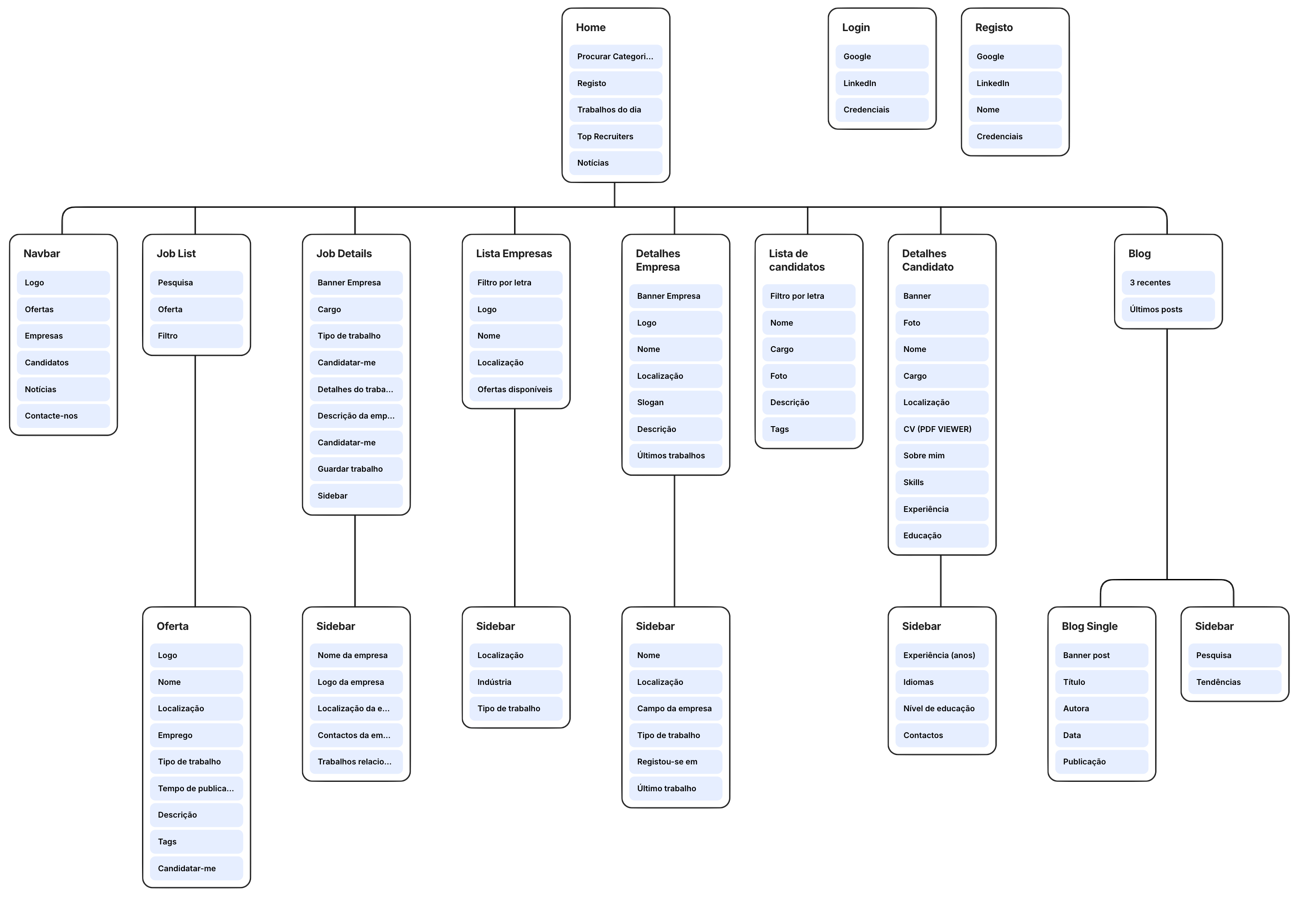


Figura 29 – Sitemap principal

# Projeto

## Elementos gerais

### Download e preparação do template

Após a escolha do **template base**, foi feito o **download dos ficheiros** necessários.  
O projeto foi então **aberto no Visual Studio Code**, onde se iniciou um terminal integrado e se acedeu à pasta do projeto com o comando:

cd [nome-da-pasta]

#### Instalação de dependências

Dentro da pasta, foi executado o comando:

npm install

Este comando é responsável por instalar todos os **módulos Node (node\_modules)** e gerar automaticamente o ficheiro package-lock.json, necessário para garantir que todas as dependências estão corretamente sincronizadas com a versão do projeto.

#### Execução do projeto

Com a instalação concluída, o projeto ficou pronto a ser executado com o comando:

npm run dev

Este comando inicia o servidor de desenvolvimento. Ao aceder ao **link apresentado no terminal**, é possível visualizar o projeto a correr **em tempo real no navegador**.

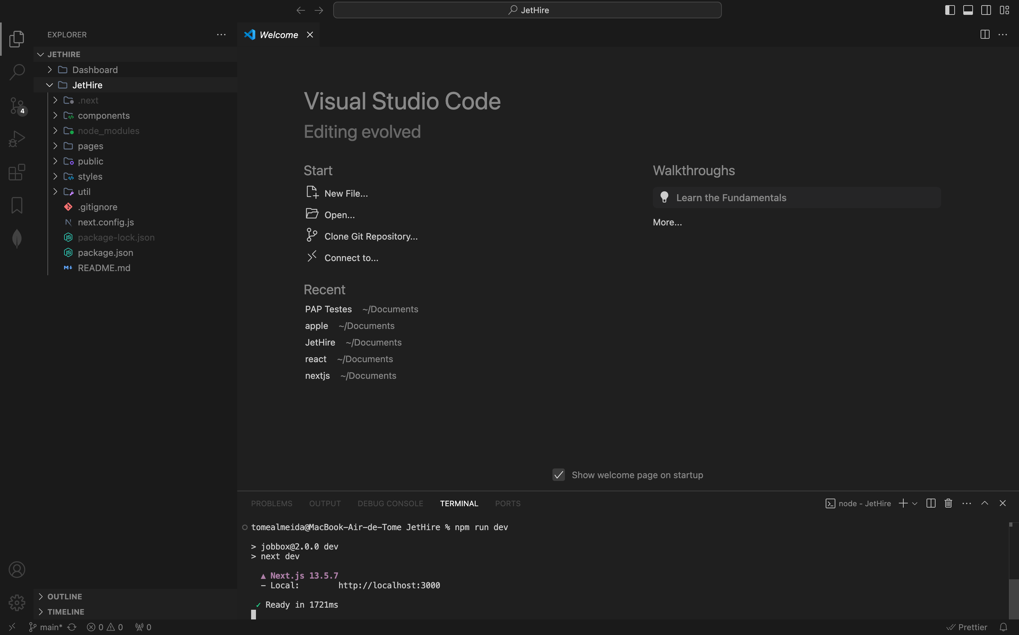


Figura 30 - npm run dev

### Logótipo

|  |  |
| --- | --- |
| O logótipo foi desenvolvido de forma a representar o conceito de uma **estrutura empresarial dinâmica**, destacando **quatro elementos modulares**. Para este projeto, um dos elementos é destacado individualmente — simbolizando o **novo membro** que se integra na equipa da empresa, tal como acontece no fluxo real da aplicação. | Figura 31 - Logótipo |

### Escolha da paleta de cores

#### Cor principal

A escolha da cor principal foi realizada com base na teoria de design de interfaces. Utilizou-se o modelo **HSB (Hue, Saturation, Brightness)** para gerar tons coerentes.

A cor principal escolhida foi o **vermelho**, por ser uma cor marcante, associada à ação e visibilidade.

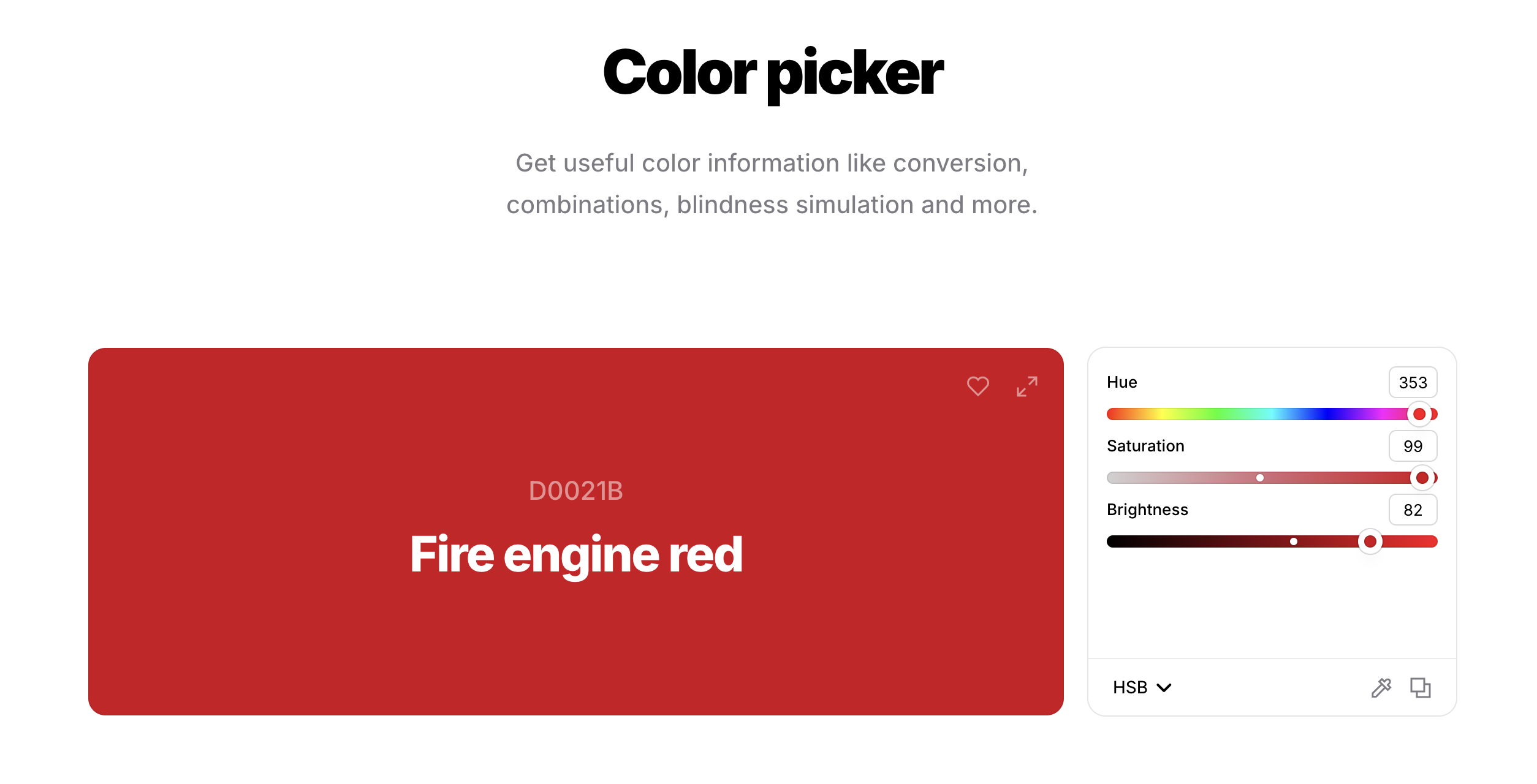


Figura 32 - Cor principal

**Definição de tons**

Para criar uma paleta equilibrada, o processo consistiu em:

* Gerar o **tom mais claro** aumentando a **luminosidade** e reduzindo a **saturação**;

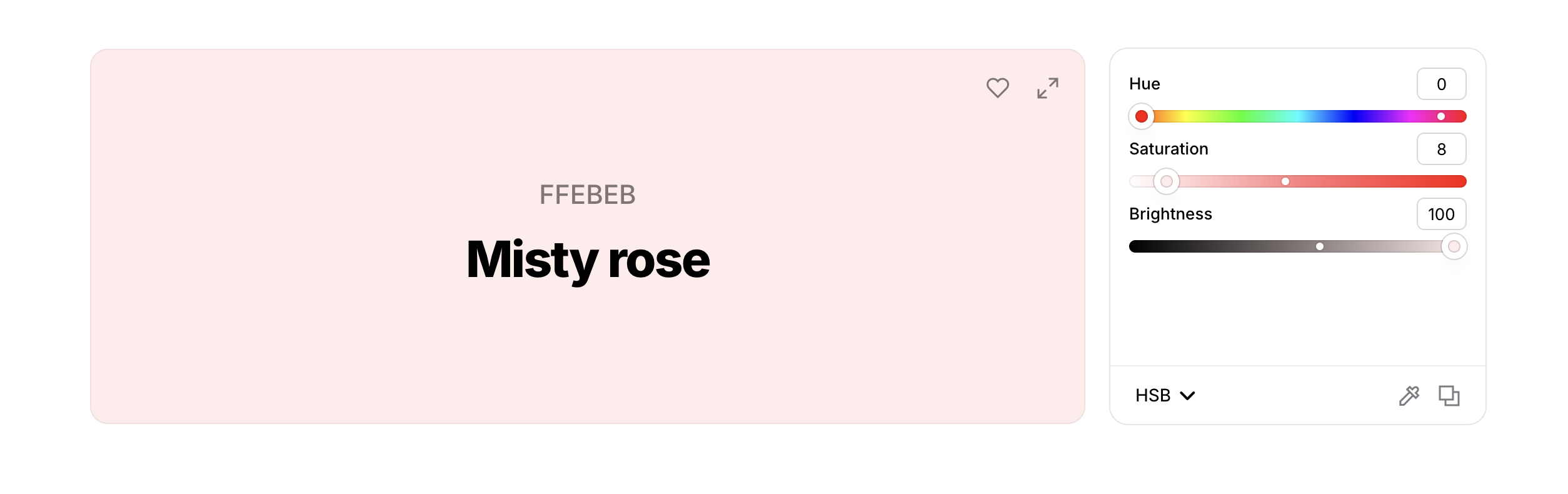


Figura 33 - Escolha de um tom claro

* Gerar o **tom mais escuro** diminuindo a **luminosidade** e aumentando a **saturação**.

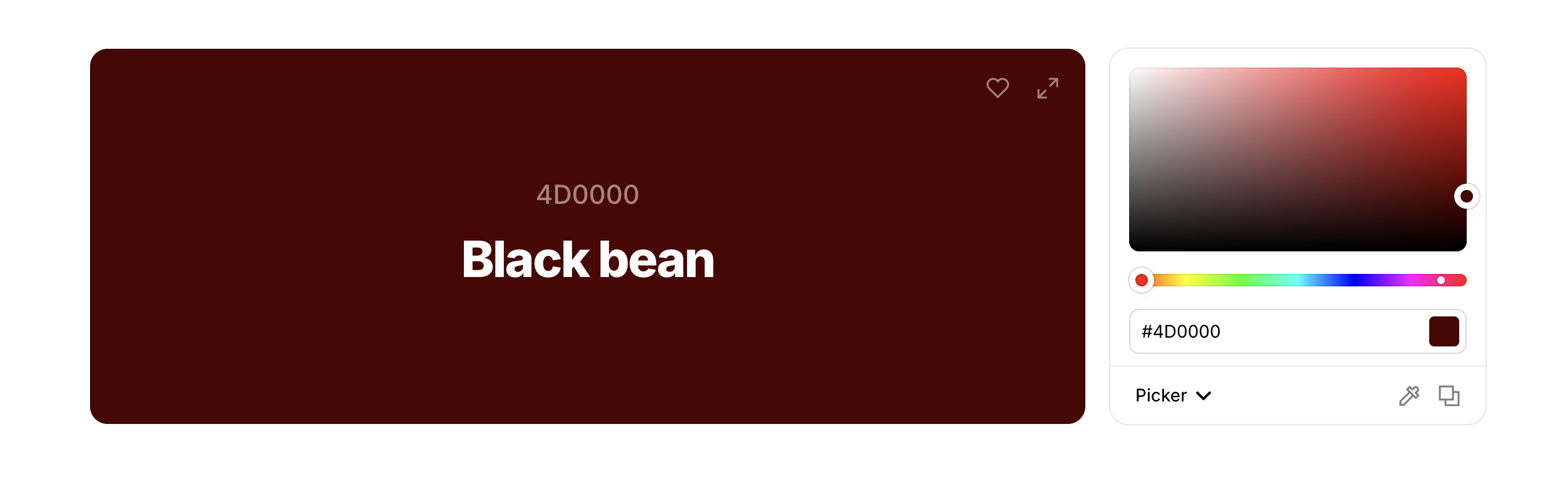


Figura 34 - Escolha de um tom escuro

**Criação dos gradientes**

Com os três tons definidos, foram gerados gradientes:

* **Gradiente claro**: do tom mais claro até à cor principal.

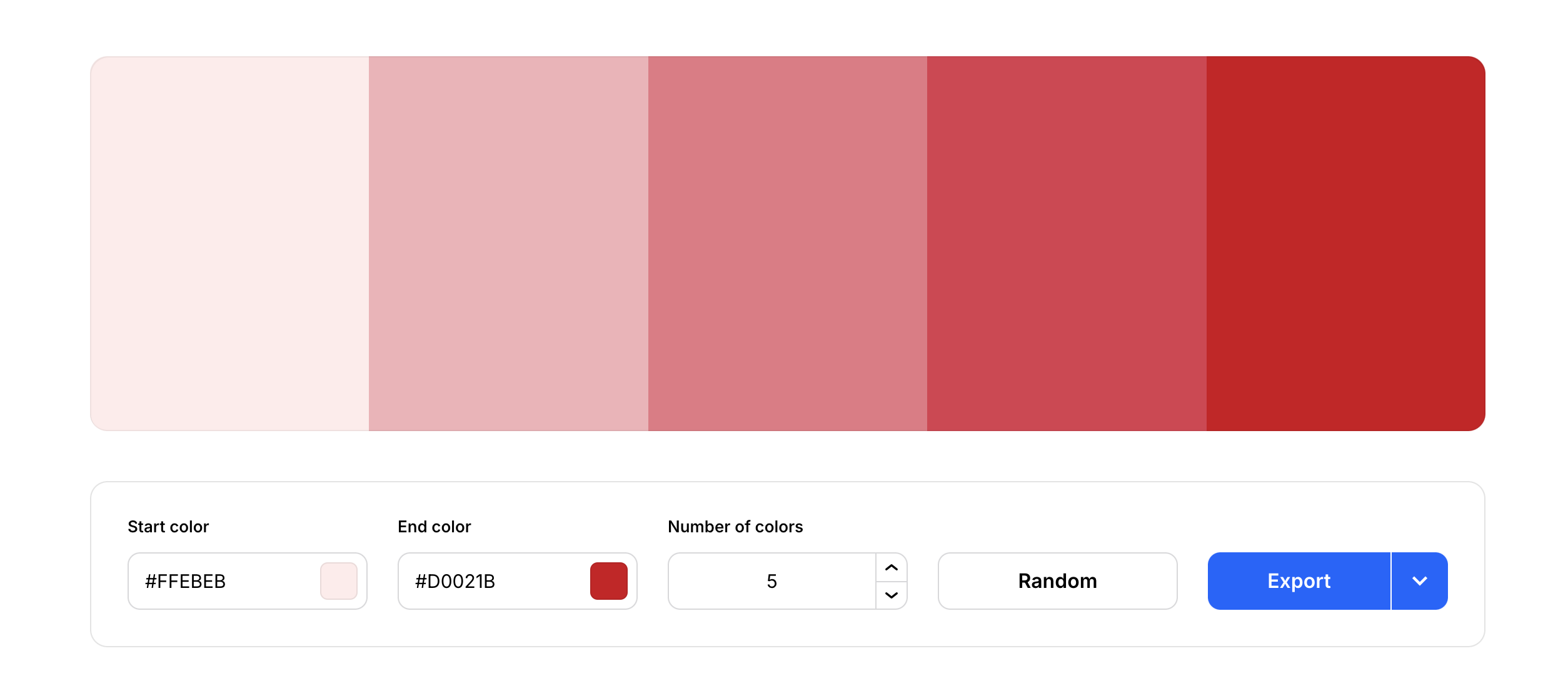


Figura 35 - Criação de um gradiente claro

* **Gradiente escuro**: da cor principal até ao tom mais escuro.

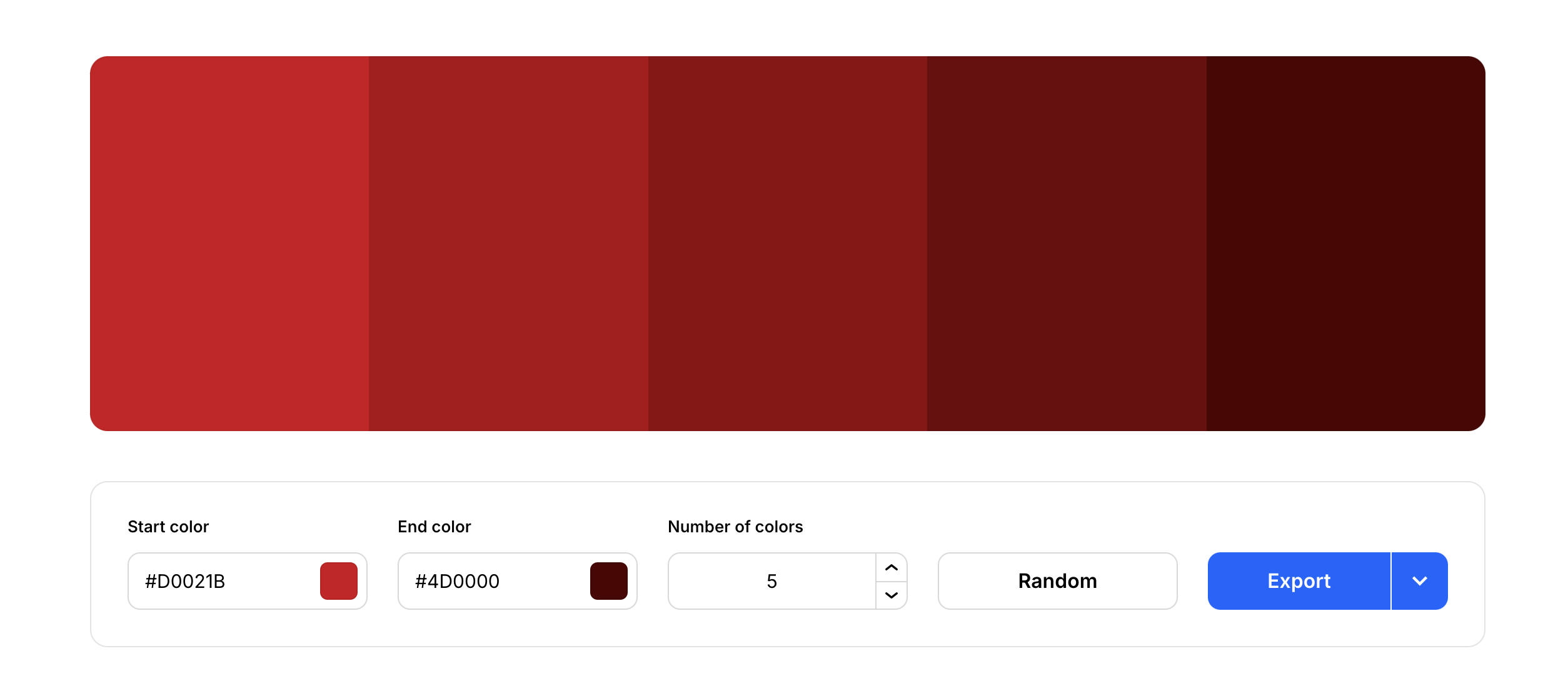


Figura 36 - Criação de um gradiente escuro

#### Cores secundárias

As cores secundárias foram definidas para representar diferentes **estados de notificação** no sistema:

|  |  |  |  |
| --- | --- | --- | --- |
| **Amarelo** - avisos | **Azul** – informações | **Verde** - confirmações | **Vermelho** - erros |

|  |  |  |
| --- | --- | --- |
| Figura 37 - Amarelo | Figura 38 - Azul | Figura 39 - Verde |

Cada uma destas cores passou pelo mesmo processo de **definição de tons** e **criação de gradientes**.

#### Cores neutras

A paleta de cores neutras acompanha a principal e é usada em fundos, textos e elementos complementares. Inclui tons que vão do **branco ao preto**, com variantes de cinzentos.



Figura 40 - Cores neutras

#### Implementação da paleta de cores no projeto

Para tornar estas cores acessíveis a todo o projeto, foram definidas como **variáveis globais** no CSS, dentro da pseudo-classe **:root**:

:root {

--red-900: #7b0000;

--red-500: #d40000;

--red-100: #ffb3b3;

...

}

Estas variáveis podem ser reutilizadas facilmente em qualquer parte do CSS:

a {

color: var(--red-900);

}

### Criação de um ambiente seguro

Num projeto **Next.js**, o ficheiro .env é utilizado para armazenar **informações sensíveis** de forma segura, como **palavras-passe**, **chaves de API** ou **URLs de ligação a bases de dados**. Estas variáveis não ficam visíveis no código-fonte e são acessíveis apenas dentro do ambiente da aplicação, o que ajuda a manter a segurança e organização do projeto.

Por padrão, o ficheiro .env é adicionado ao .gitignore, o que impede que seja incluído em repositórios públicos ou partilhado por engano, evitando a exposição de dados confidenciais.

As variáveis definidas neste ficheiro podem ser acedidas no código através de process.env, como ilustrado no exemplo:

GOOGLE\_SECRET\_ID: process.env.GOOGLE\_SECRET\_ID;

Este mecanismo permite centralizar e proteger valores críticos da aplicação de forma prática, tornando o ficheiro .env uma **ferramenta essencial para a segurança e manutenção do projeto**.

### Ligação á base de dados com Mongoose

#### O que é o MongoDB?

O **MongoDB** é uma base de dados **NoSQL**, orientada a documentos. Ao contrário das bases de dados relacionais (como MySQL ou PostgreSQL), onde os dados são organizados em **tabelas e linhas**, o MongoDB armazena os dados em **coleções de documentos**, semelhantes a objetos **JSON**.

Cada documento pode ter uma estrutura diferente, o que torna o MongoDB especialmente útil em projetos com dados variados ou dinâmicos — como aplicações web modernas. É uma base de dados **flexível, escalável e rápida**, muito utilizada em conjunto com aplicações feitas em **JavaScript/TypeScript**, como é o caso deste projeto.

#### O que é o Mongoose?

O **Mongoose** é uma biblioteca para **Node.js** que facilita a ligação e o uso do MongoDB. Com o Mongoose, é possível:

* Definir **schemas** que estruturam os dados de forma consistente;
* Validar automaticamente os dados inseridos;
* Executar operações de **CRUD** (criar, ler, atualizar, eliminar);
* Gerir relações entre dados;
* Usar middlewares e métodos personalizados.

#### Instalação

A instalação do Mongoose é feita com o seguinte comando no terminal:

npm install mongoose

Se a instalação for bem-sucedida, a dependência será adicionada à pasta **node\_modules**.

#### Ligação à base de dados

Para estabelecer a ligação, foi criado um ficheiro chamado **mongodb.js** dentro da pasta **/lib**, com o seguinte código:

import mongoose from "mongoose";

export const connectMongoDB = async () => {

try {

await mongoose.connect(process.env.MONGODB\_URL);

console.log("Conectado a MongoDB");

} catch (error) {

console.log("Erro a conectar:", error);

}

};

A função **connectMongoDB** tenta ligar-se ao URL definido no ficheiro **.env**. Se for bem-sucedida, é apresentada a mensagem de sucesso. Caso contrário, é mostrado o **erro** no terminal.

### Envio de dados para a base de dados

O envio de dados segue sempre uma **estrutura lógica** e sequencial:

* **Frontend:** onde o utilizador insere os dados (ex.: formulário).
* **API:** recebe os dados do frontend e processa-os.
* **Base de dados:** armazena a informação permanentemente através do modelo Mongoose.

#### Exemplo com fetch

No frontend, usa-se a função fetch para enviar dados para uma API:

await fetch("/api/users/add", {

method: "POST",

headers: { "Content-Type": "application/json" },

body: JSON.stringify({

name: "Ana",

email: "ana@email.com",

}),

});

* O método **POST** indica que estamos a enviar dados.
* O conteúdo é convertido para **JSON**.
* O destino é uma rota interna, que trata a inserção dos dados.

#### API (Back-End)

A API serve como **ponte entre o frontend e a base de dados**. Abaixo está um exemplo de API em Next.js com Mongoose:

import dbConnect from "@/lib/dbConnect";

import User from "@/models/User";

export default async function handler(req, res) {

await dbConnect();

if (req.method === "POST") {

const user = new User(req.body);

await user.save();

res.status(201).json({ success: true });

}

}

* **dbConnect()** estabelece ligação à base de dados.
* Criamos um novo objeto User com os dados recebidos.
* Por fim, utilizamos **.save()** para guardar na MongoDB.

### Cloudinary — Gestão de imagens e documentos

Para armazenar imagens e documentos — como **logótipos**, **fotografias de perfil**, **banners** e **CVs** — utilizei a plataforma **Cloudinary**.

Esta ferramenta permite **fazer o upload e gestão de ficheiros na nuvem**, poupando espaço e **descarregando a base de dados de armazenar ficheiros pesados**. Em vez de guardar o conteúdo de imagens diretamente, **apenas é guardada uma string (URL)** na base de dados, o que torna o acesso mais rápido e a gestão mais leve.

#### Configuração no .env

As credenciais devem ser guardadas de forma segura no ficheiro .env:

CLOUDINARY\_URL=cloudinary://[API\_KEY]:[API\_SECRET]@[CLOUD\_NAME]

Este URL será utilizado no ficheiro de configuração.

#### Funções para Envio e Eliminação

No projeto, foram criadas duas funções principais no ficheiro cloudinary.ts:

**Upload de imagens**

async function uploadToCloudinary(buffer: Buffer) {

return new Promise<{ secure\_url: string; public\_id: string }>((resolve, reject) => {

cloudinary.uploader

.upload\_stream(

{

folder: "pfp",

resource\_type: "image",

overwrite: true,

},

(error, result) => {

if (error || !result) reject(error);

else resolve({ secure\_url: result.secure\_url, public\_id: result.public\_id });

}

)

.end(buffer);

});

}

Esta função:

* Recebe a imagem em **buffer**.
* Envia-a para a pasta **pfp** no Cloudinary.
* Retorna o **URL seguro** e o **ID público**, que são depois guardados na base de dados.

**Eliminação de imagens**

async function deleteFromCloudinary(publicId: string) {

try {

await cloudinary.uploader.destroy(publicId);

} catch (error) {

console.error("Erro ao apagar imagem:", error);

}

}

Esta função:

* Elimina a imagem através do seu **publicId**.
* É utilizada, por exemplo, quando um utilizador atualiza a sua foto de perfil.

### NextAuth – Sistema de autenticação com sessões seguras

A autenticação da aplicação foi implementada com a biblioteca **NextAuth**, usando o **provedor de credenciais** (Credentials Provider). Esta escolha permite um controlo completo sobre o processo de login, adaptado à estrutura da base de dados e aos diferentes tipos de utilizadores existentes: **administradores** e **membros de empresas**.

#### Autorização personalizada

A lógica de autenticação personalizada está definida dentro da função **authorize**, que é executada assim que o utilizador submete as suas credenciais. Esta função realiza três passos essenciais:

async authorize(credentials) {

await dbConnect();

const { user, pass } = credentials!;

const admin = await Administrator.findOne({ user });

...

}

1. **Ligação à base de dados** com dbConnect(), garantindo que a verificação será feita com dados atualizados.
2. **Pesquisa de um utilizador válido**, começando por procurar entre os administradores. Se não existir, é feita uma pesquisa entre os membros das empresas.
3. **Validação da palavra-passe**, utilizando bcrypt.compare() para verificar se a palavra-passe inserida corresponde à que está encriptada na base de dados.

Se as credenciais forem válidas, a função devolve um objeto com os dados do utilizador autenticado, como o seu id, nome, tipo de utilizador (admin ou empresa), e cargo, permitindo construir a sua sessão.

#### Criação e gestão de sessões com JWT

A aplicação está configurada para usar **JWT (JSON Web Tokens)** como estratégia de sessão. Esta abordagem permite guardar os dados da sessão de forma segura, **sem depender do servidor**, o que facilita a escalabilidade e a persistência da sessão mesmo após atualizações da página.

session: {

strategy: "jwt",

},

Os dados do utilizador autenticado são guardados dentro do token através do **callback jwt()**, que é executado logo após o login:

async jwt({ token, user }) {

if (user) {

token.id = user.id;

token.name = user.name;

token.type = user.type;

token.role = user.role;

token.company = user.company;

}

return token;

}

Este token funciona como uma **identidade digital assinada**: contém a informação essencial sobre o utilizador e é automaticamente incluído em todos os pedidos subsequentes da aplicação.

#### Mas o que é exatamente um JWT?

Um **JSON Web Token** é um objeto compacto, codificado em base64 e assinado digitalmente, que armazena informação de forma segura. No contexto da aplicação, ele funciona como um “cartão de identificação digital” que acompanha o utilizador ao navegar entre páginas.

#### Sessões protegidas e acessíveis no front-end

Para tornar os dados do token acessíveis ao front-end, é utilizado o **callback session()**, que traduz o conteúdo do JWT numa sessão visível no navegador:

async session({ session, token }) {

session.user.id = token.id;

session.user.name = token.name;

session.user.role = token.role;

session.user.type = token.type;

session.user.company = token.company;

return session;

}

Com esta estrutura, o front-end pode aceder a session.user para **identificar o utilizador ativo**, mostrar os seus dados, **proteger rotas específicas** (por exemplo, rotas apenas para administradores), e personalizar a experiência consoante o seu tipo.

#### Exemplo prático

Supondo que o utilizador **José Silva** (cargo: *recruiter*, empresa: *TechNova*) inicia sessão com sucesso, o token JWT gerado irá conter algo como:

{

"id": "664abec123",

"name": "José Silva",

"role": "recruiter",

"company": "TechNova"

}

Este token acompanha todos os pedidos do utilizador e permite à aplicação saber, a qualquer momento, **quem está autenticado**, **que permissões tem**, e **a que empresa pertence** — tudo sem necessidade de chamadas adicionais ao servidor.

## Página principal

## Dashboard

# Conclusão

O desenvolvimento da Jet Hire foi uma experiência extremamente enriquecedora, tanto a nível técnico como pessoal.

Ao longo deste projeto, aprofundei os meus conhecimentos em tecnologias modernas como Next.js, MongoDB, Mongoose e autenticação com NextAuth, além de consolidar boas práticas de desenvolvimento.

Este projeto permitiu-me aplicar, de forma prática, os conteúdos aprendidos durante o curso, desafiando-me a encontrar soluções para problemas reais. Além disso, tive a oportunidade de explorar ferramentas como o Cloudinary e a criação de formulários dinâmicos, que contribuíram para a construção de uma plataforma moderna, intuitiva e funcional.

Acredito que esta aplicação poderá ser útil para jovens e empresas da área das Tecnologias de Informação e, se for desenvolvida e mantida futuramente, poderá mesmo ser disponibilizada como solução real no mercado.

Com este trabalho, concluo a minha Prova de Aptidão Profissional com orgulho no percurso que trilhei e motivação para continuar a evoluir na área da programação.

# Webgrafia

**ZacsTech. 2023.** *How to install Github Desktop on Ubuntu 22.04.* Acedido em outubro de 2024: <https://www.youtube.com/watch?v=ki0MuWceGk4&ab_channel=ZacsTech>

**UXTools. 2023.** *How I make UI color palletes.* Acedido em novembro de 2024: <https://www.youtube.com/watch?v=yYwEnLYT55c&ab_channel=UXTools>

**Heroicons. 2024.** *Beautiful hand-crafted SVG icons, by the makers of Tailwind CSS.* Acedido em novembro de 2024: <https://heroicons.com/>

**GTCoding. 2023.** *Step-by-Step Guide: Adding Google Authentication with NextAuth in Next.js 13 and MongoDB.* Acedido em novembro de 2024: <https://www.youtube.com/watch?v=6N3Rumo-c3s>

**Clerk. 2024.** *Password-based authentication in Next.js.* Acedido em dezembro de 2024: <https://clerk.com/blog/password-based-authentication-nextjs>

**HeyNode. 2020.** *Salt and hash passwords with bcrypt.* Acedido em dezembro de 2024: <https://heynode.com/blog/2020-04/salt-and-hash-passwords-bcrypt/>

**Mongoose. 2025.** *Guide.* Acedido em janeiro de 2025: <https://mongoosejs.com/docs/guide.html>

**Mongoose. 2025.** *Subdocuments.* Acedido em janeiro de 2025: <https://mongoosejs.com/docs/subdocs.html>

**React. 2025.** *useState.* Acedido em janeiro de 2025: <https://react.dev/reference/react/useState>