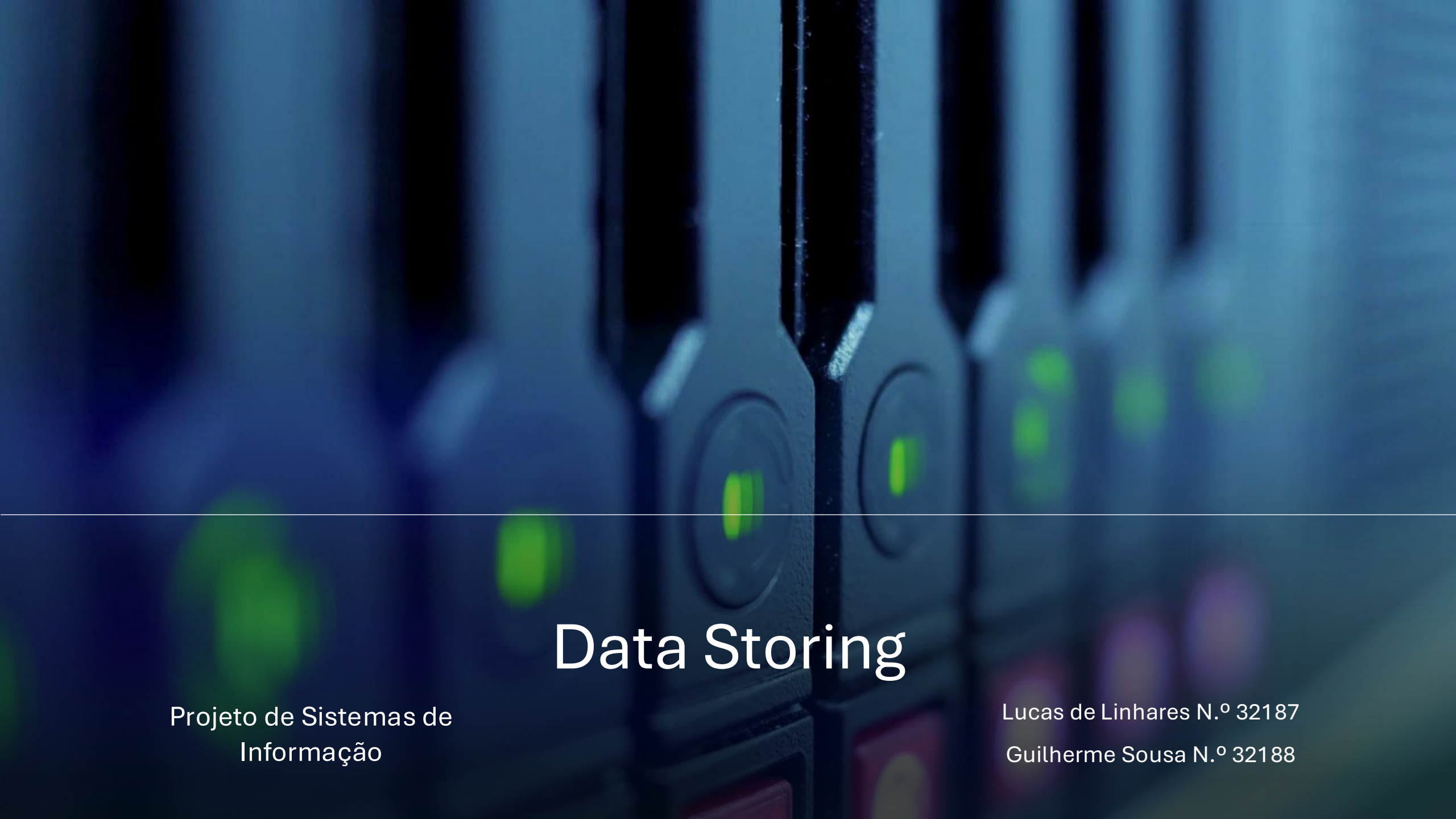


Data Storing

Projeto de Sistemas de Informação

Lucas de Linhares N.º 32187

Guilherme Sousa N.º 32188

Índice


Enquadramento do tema
Objetivos
Requisitos
Modelo de casos de uso
Modelo de dados
Arquitetura geral do projeto
Tecnologias envolvidas
Mockups
Estudos
Estudo Password Hashing
Estudo Primitivas de Sincronização
Build System
Sistema de Internacionalização
Tudo o Resto
Futuras Melhorias
Mostrar projeto

Enquadramento

- Este projeto tem como objetivo o desenvolvimento de uma plataforma web que permite aos utilizadores carregar e categorizar imagens de peças de roupa. O sistema utiliza essas informações e sugere outfits personalizados com base nas condições meteorológicas.
- A plataforma utiliza algoritmos de machine learning que, a partir dos dados fornecidos pelos utilizadores, gera combinações de vestuário apropriadas para diferentes estações e temperaturas, otimizando a escolha de roupas de acordo com o clima e preferências pessoais.

OBJETIVOS

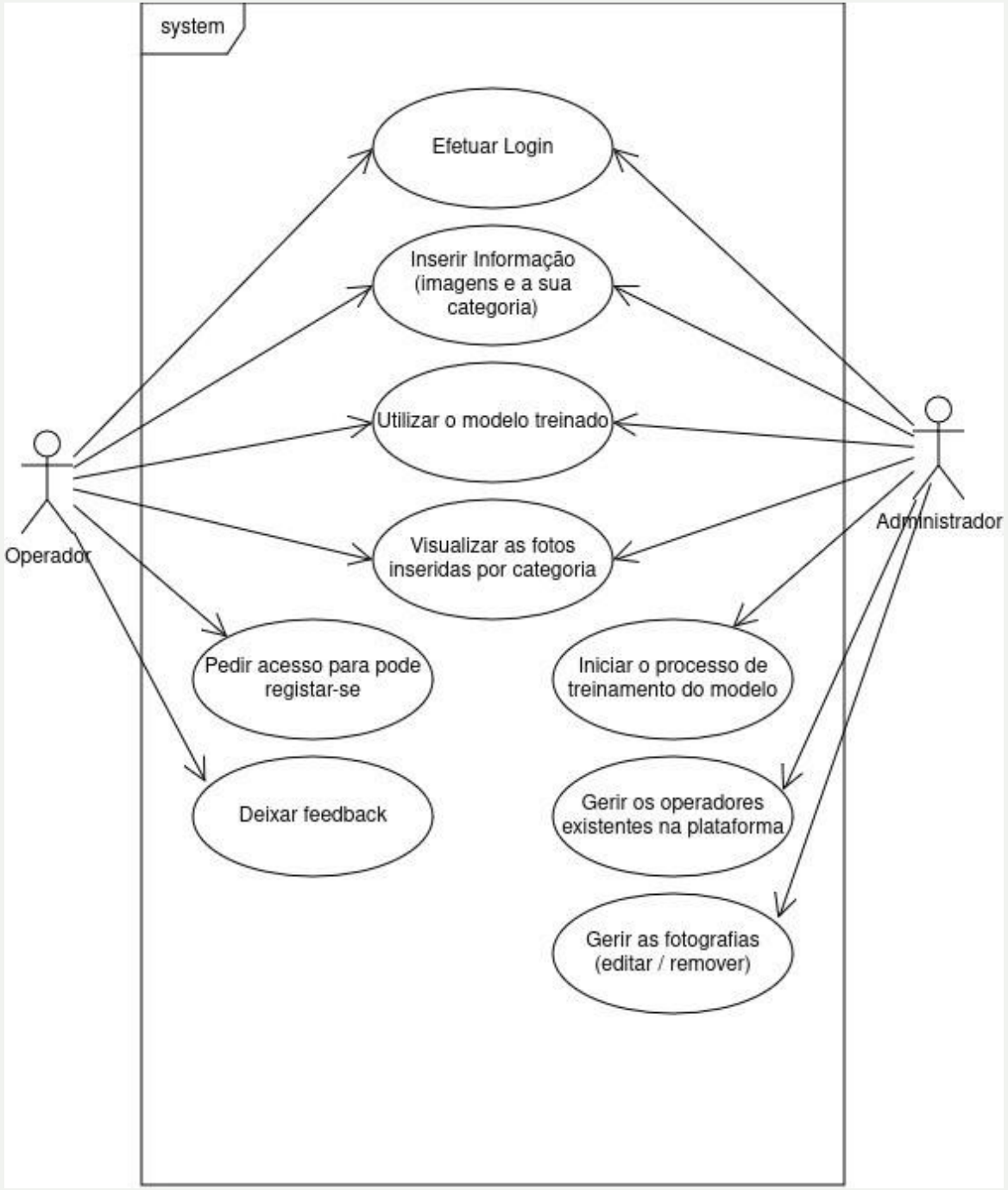
- Desenvolver uma aplicação multiplataforma (Web, Windows, Linux e Android) que permita aos utilizadores gerir e inserir imagens de roupa, classificando-as detalhadamente.
- O dataset gerado será utilizado para treinar modelos de machine learning, focados em sugerir outfits personalizados.
- O projeto priorizou performance, segurança e qualidade de código, com escolhas tecnológicas feitas com base em benchmarks e estudos detalhados.

Requisitos

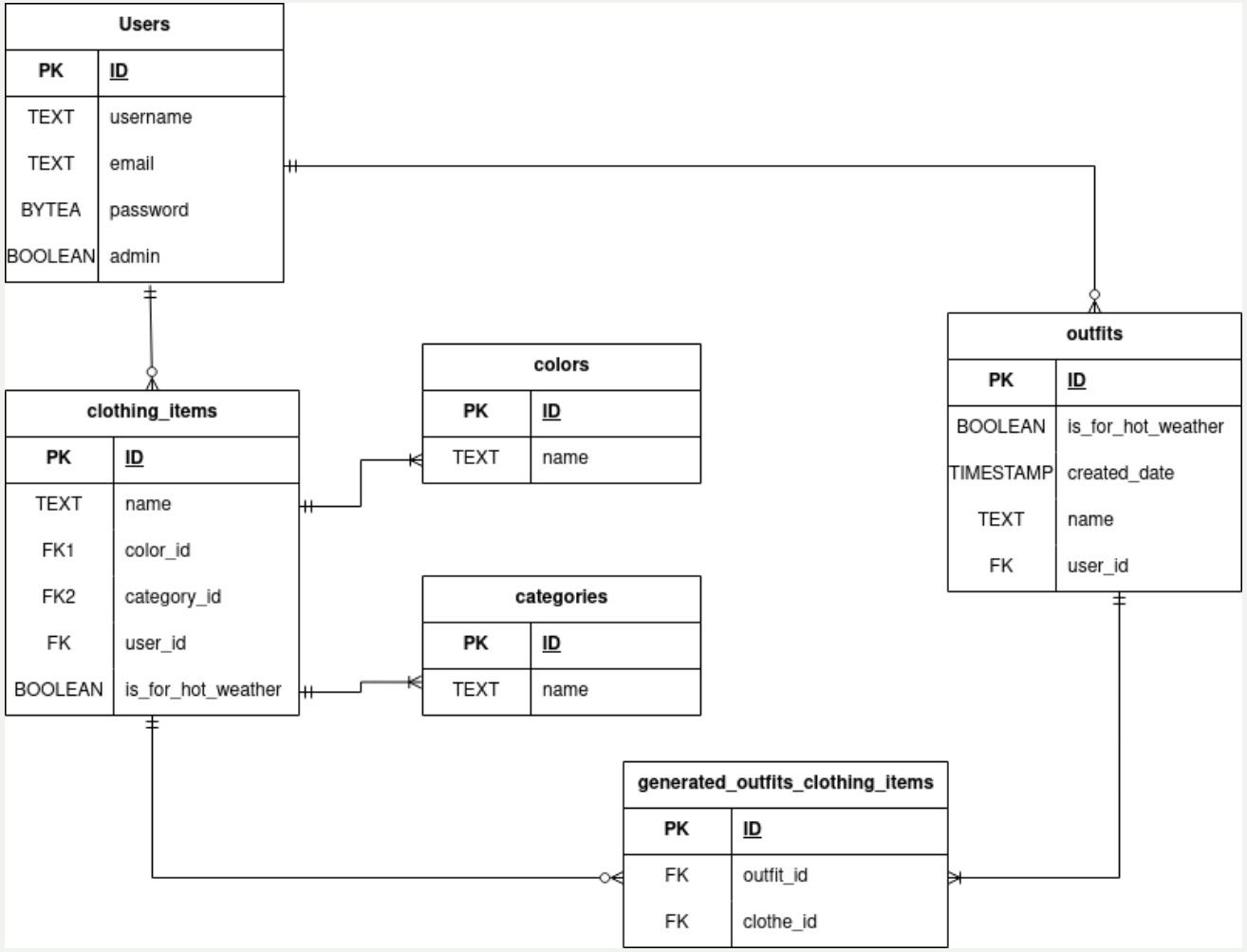
Nº	Descrição	Perfil	Prioridade
RF1	Conseguir efetuar login	Administrador Operador	Alta
RF2	Inserir informação (imagens e a sua categorização)	Administrador Operador	Alta
RF3	Pedir acesso para poder se registar na plataforma	Operador	Alta
RF4	Conseguir usar o modelo treinado	Administrador Operador	Alta
RF5	Conseguir iniciar o processo de treinamento do modelo	Administrador	Alta
RF6	Conseguir gerir operadores (aceitar/remover operadores e editar as suas informações)	Administrador	Alta
RF7	Conseguir visualizar as fotos inseridas por categoria	Administrador Operador	Média
RF8	Conseguir gerir uma dada fotografia (editar as suas informações ou removê-la)	Administrador	Média
RF9	Conseguir deixar feedback/reportar bugs	Operador	Baixa

Nº	Descrição	Prioridade
RNF1	Ter o software disponível em vários idiomas (português, inglês)	Baixa
RNF2	Ter o software disponível em todas as principais plataformas (web, mobile e desktop)	Média
RNF3	A plataforma tem de ter uma boa performance	Alta
RNF4	A plataforma tem de ser segura	Alta
RNF5	A plataforma deve estar disponível 24 horas por dia, 7 dias por semana.	Alta
RNF6	A plataforma deve ter um design responsivo	Alta

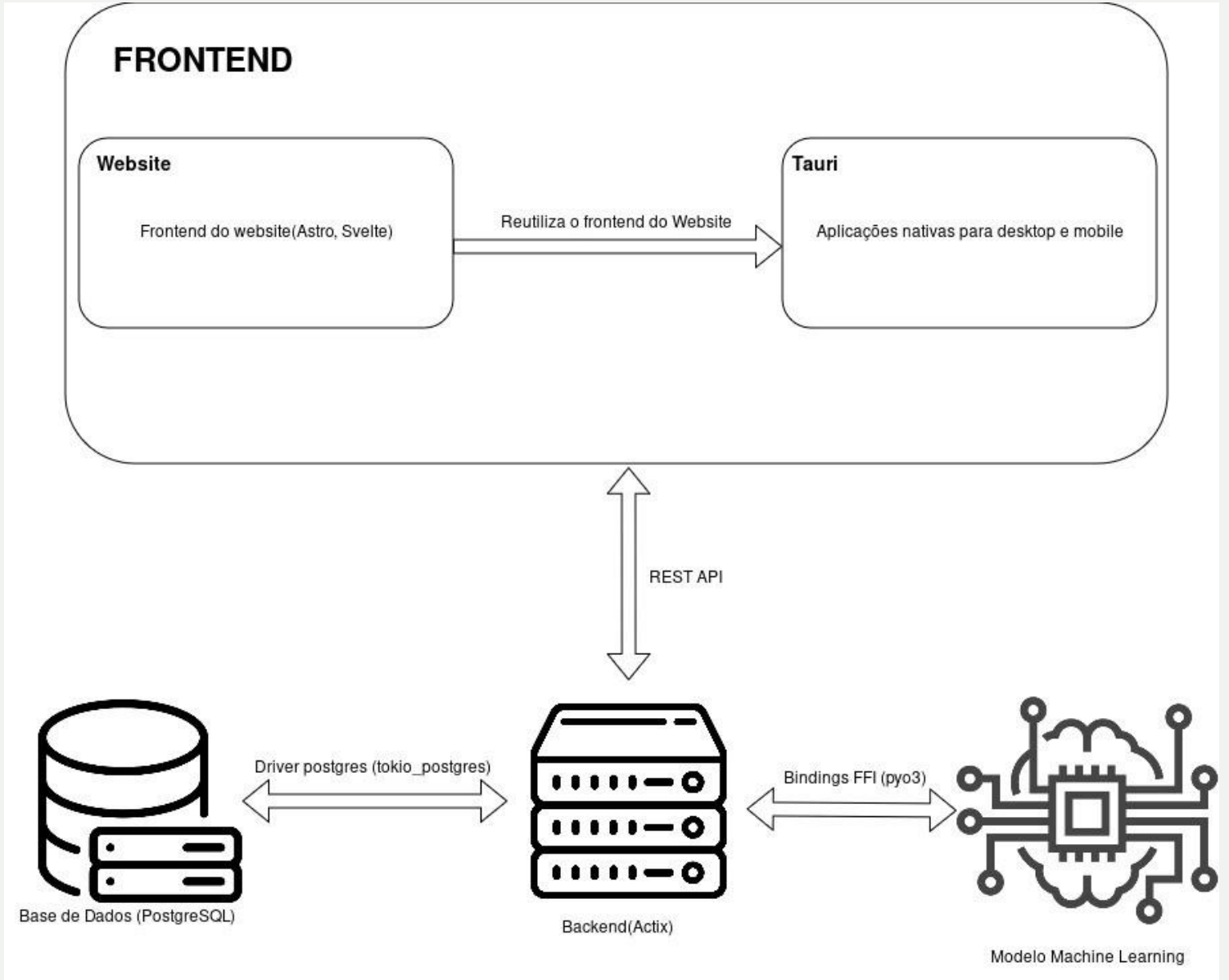
Modelo de casos de uso



Modelo de dados (modelo relacional)



Arquitetura geral da solução



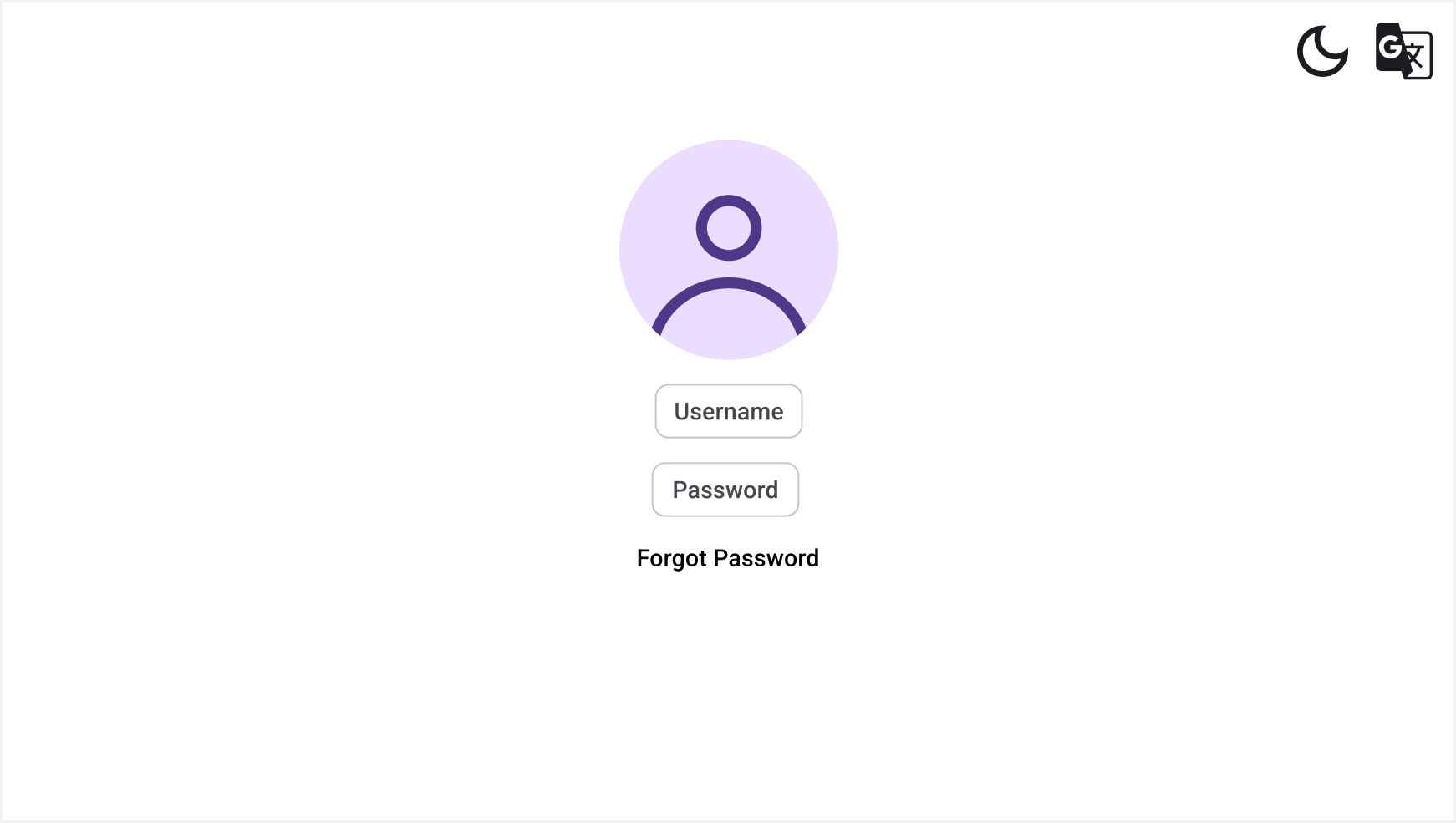
Tecnologias envolvidas

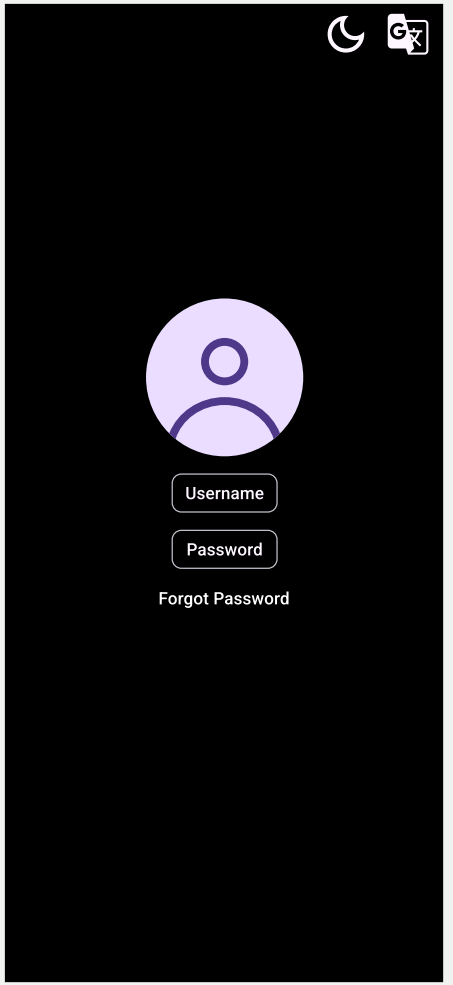
- Frontend
  - Astro – Metaframework moderna
  - Svelte – UI framework moderna
  - BunJS – Runtime de JS rápido e moderno
  - TypeScript – Utilizado para reduzir erros de runtime
  - Tauri – Desenvolvimento das aplicações para cada plataforma
- Backend
  - Rust – Linguagem rápida, segura e moderna
  - Actix – Framework com ótima performance
  - PostgreSQL (tokio-postgres)

- Modelo Machine Learning
  - Python – Linguagem mais utilizada para AI

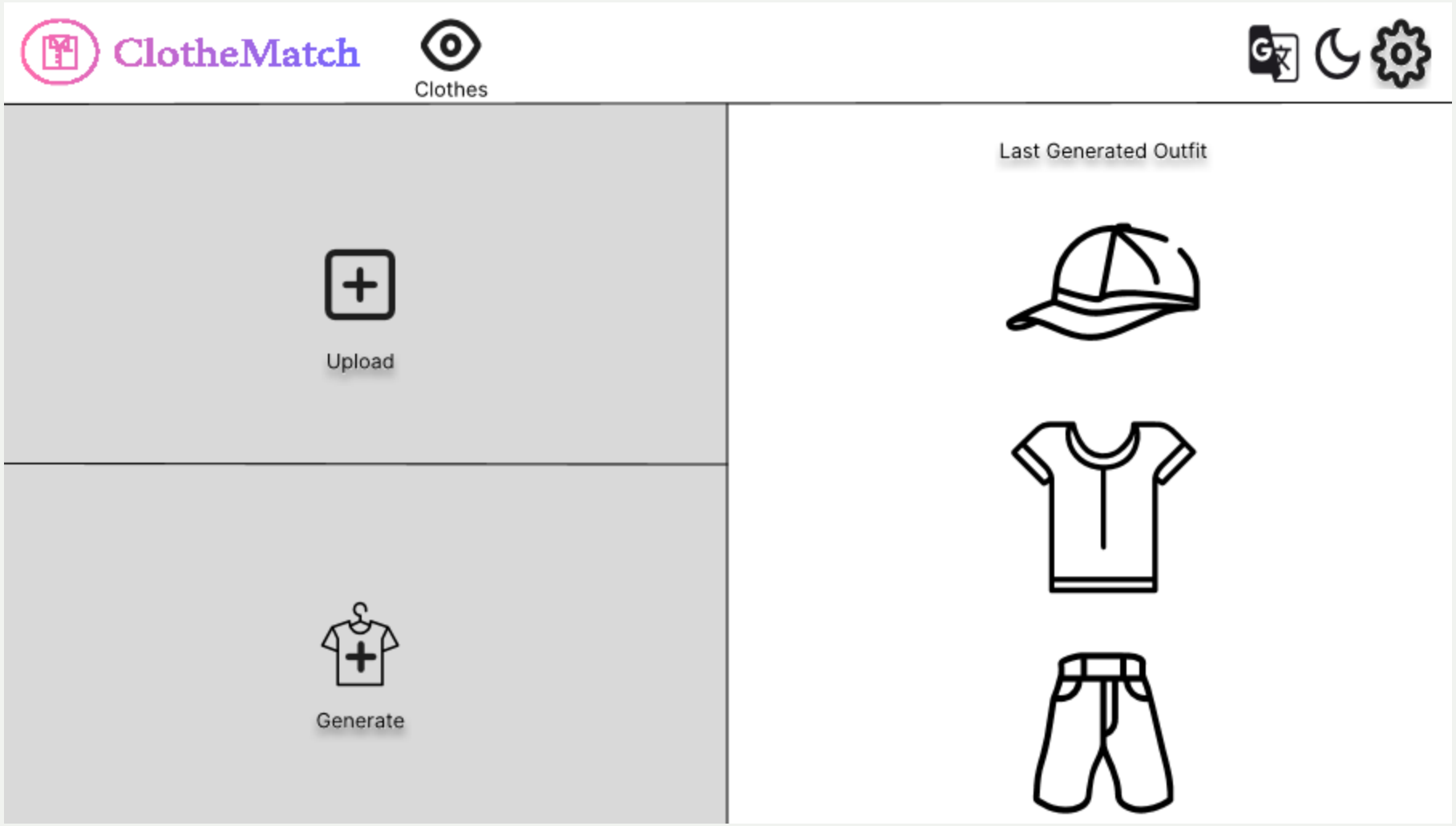
- Ferramentas usadas no Desenvolvimento
  - Postman – Testar o API
  - Datagrip – Gerir a base de dados e fornece uma consola para a mesma
  - Visual Studio Code – IDE utilizado com as extensões para as tecnologias usadas
  - GitHub – Git Provider

Mockups


Username
Password
Forgot Password

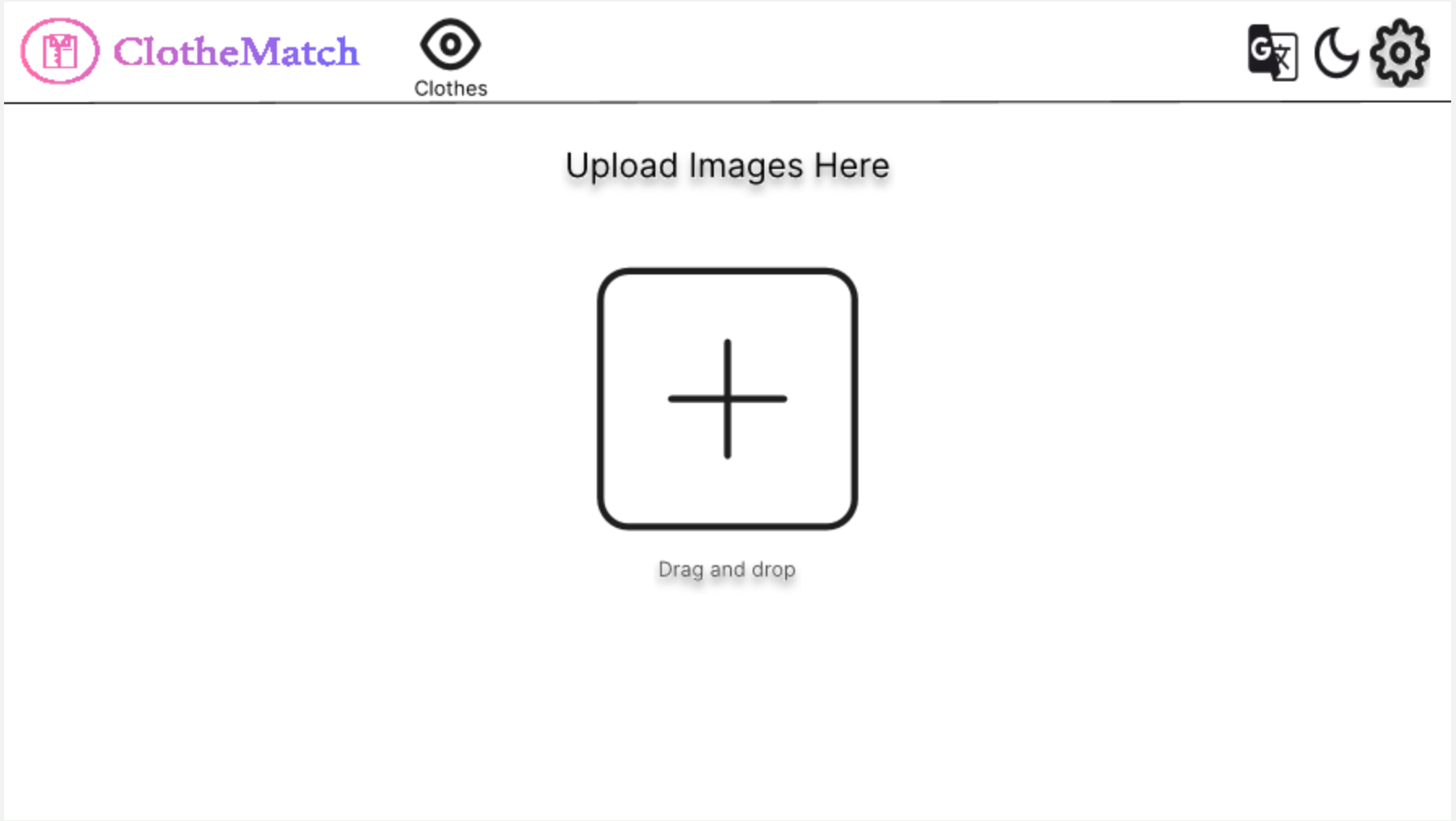

Username
Password
Forgot Password

Mockups


ClotheMatch
Clothes
Last Generated Outfit
Upload
Generate

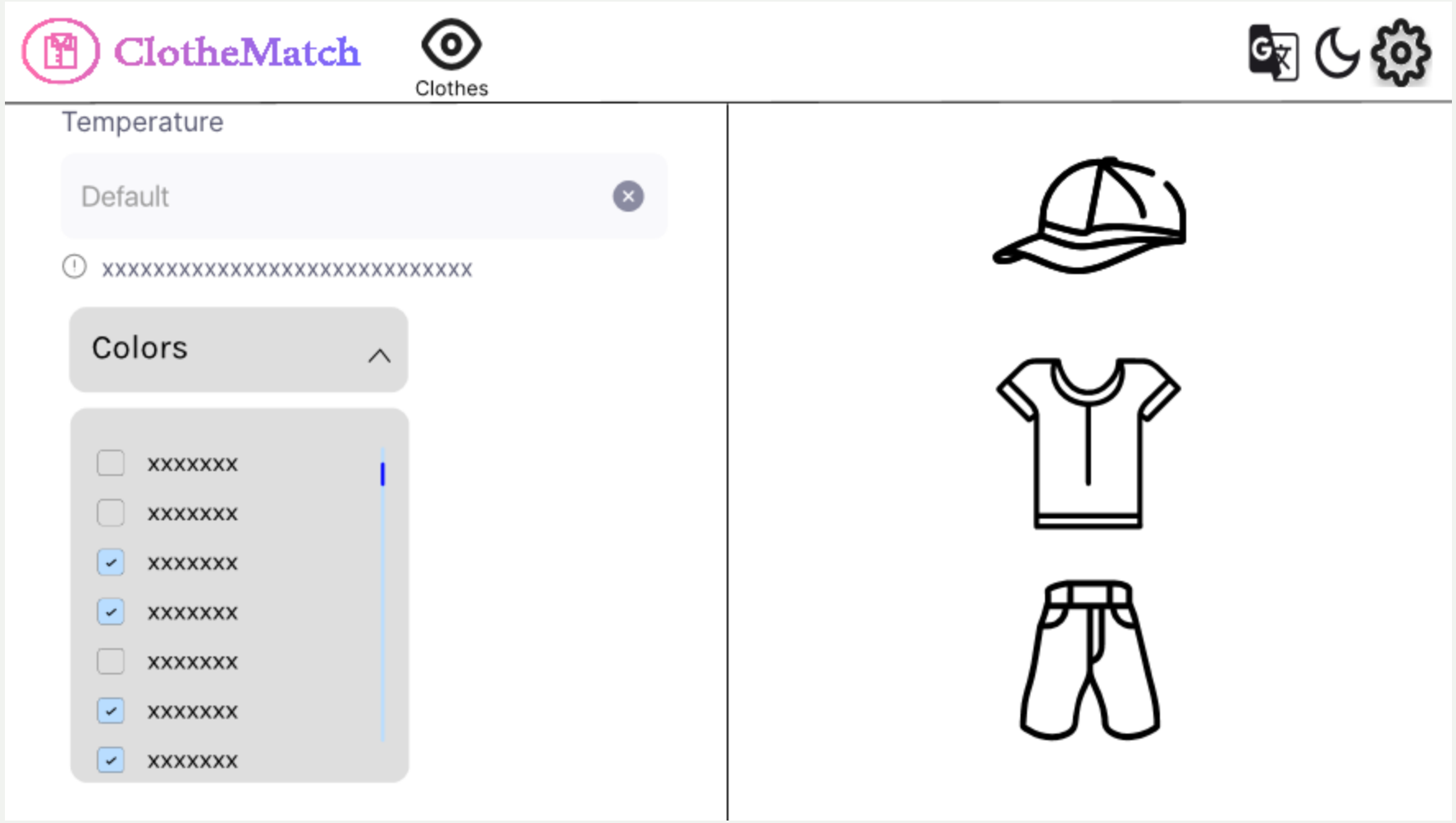

ClotheMatch
Ultimo outfit criado
Carregar
Criar

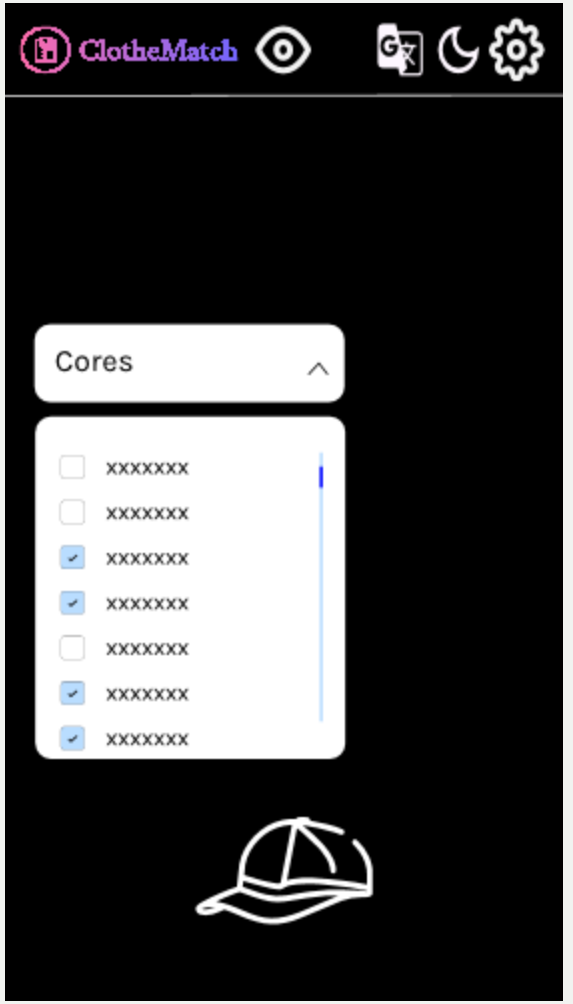
Mockups


ClotheMatch
Clothes
Upload Images Here
Drag and drop

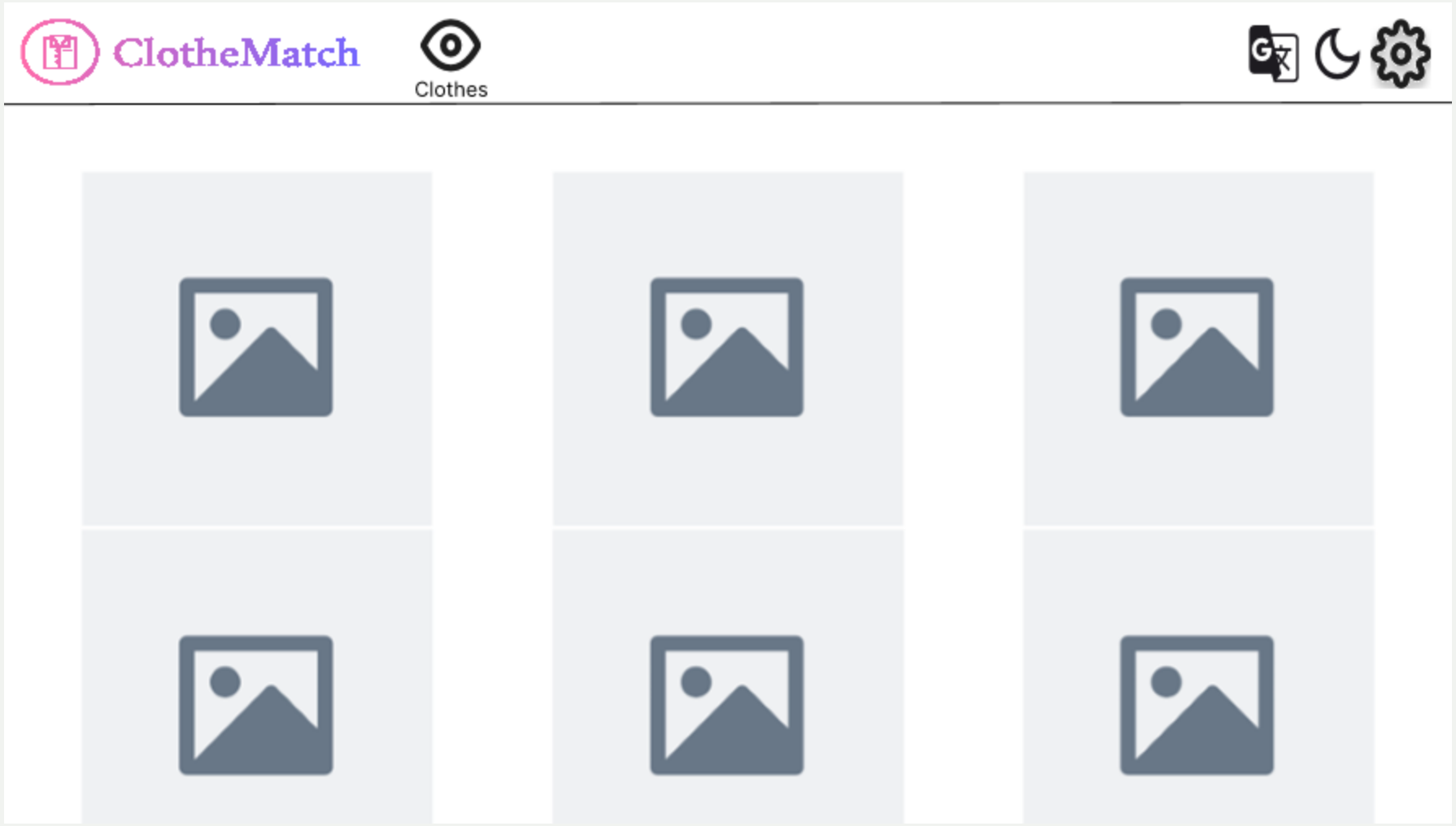

ClotheMatch
Carregar imagens aqui
Arraste aqui

Mockups


ClotheMatch
Clothes
Temperature
Default
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Colors
xxxxxxx
xxxxxxx
xxxxxxx
xxxxxxx
xxxxxxx
xxxxxxx
xxxxxxx


ClotheMatch
Cores
xxxxxxx
xxxxxxx
xxxxxxx
xxxxxxx
xxxxxxx
xxxxxxx
xxxxxxx

Mockups



ESTUDOS REALIZADOS

- Durante o desenvolvimento deste projeto, realizámos diversos estudos para garantir as melhores escolhas tecnológicas e de implementação. Estes incluíram:
- **Estudo de Dependências:** Seleção cuidadosa das tecnologias e bibliotecas utilizadas, priorizando performance, segurança e qualidade de código.
- **Estudo de Otimizações:** Investigação de técnicas para melhorar o tempo de compilação e a eficiência em runtime.
- **Estudo de Password Hashing em Rust:** Comparação de algoritmos e implementações, avaliando desempenho e segurança.
- **Estudo de Primitivas de Sincronização em Rust:** Benchmarking de diferentes abordagens para gestão de concorrência.
- Estes estudos fundamentaram as decisões que moldaram o sucesso do projeto.

ESTUDO PASSWORD HASHING

- Realizámos um estudo aprofundado sobre algoritmos de password hashing em Rust, com o objetivo de garantir a segurança e a eficiência no processo de armazenamento de palavras-passes.
- Comparamos várias implementações em Rust, incluindo `argon2`, `rust-argon2`, e `argon2-kdf` (bindings para a implementação C original), bem como outras soluções como `scrypt` e `bcrypt`.
- Fizemos benchmarks para avaliar o desempenho de cada algoritmo, testamos parâmetros como memória, tempo e custo de computação, e comparamos os resultados com as implementações padrão em PHP (`PASSWORD_BCRYPT` e `PASSWORD_ARGON2ID`).
- Com base nos resultados, decidimos utilizar o `argon2-kdf`, que demonstrou ser a solução mais rápida e eficiente para o nosso caso. Além disso, contribuímos para um projeto open-source ao realizar um fork da biblioteca `argon2-kdf` e submeter melhorias que foram aceites pelo autor original.
- Este estudo garantiu uma abordagem robusta para a segurança do sistema, sem comprometer a performance.

ESTUDO PRIMITIVAS DE SINCRONIZAÇÃO

- Realizámos um estudo sobre primitivas de sincronização em Rust para escolher as melhores opções para garantir a concorrência eficiente no nosso projeto. Comparamos as primitivas de sincronização da biblioteca padrão (`std`) com as da biblioteca `parking_lot`, analisando o desempenho de mutexes, read-write locks e barreiras, tanto em cenários de thread única como multi-thread.
- Através de benchmarks, avaliámos o tempo de execução em diversas situações, incluindo cenários de contenção. Embora a `parking_lot` apresentasse algumas vantagens em termos de desempenho, os resultados mostraram que as primitivas da biblioteca padrão (`std`) tinham um desempenho similar e, como estas são mais amplamente utilizadas e bem suportadas, optámos por utilizá-las no nosso projeto.
- Este estudo permitiu garantir que a sincronização no nosso sistema fosse eficiente, sem sobrecargar o desempenho global.

BUILD SYSTEM

- Para simplificar o processo de construção e compilação de cada subprojeto, decidimos criar um build system personalizado. Como cada subprojeto possui o seu próprio sistema de build, com várias opções e configurações de compilação, seria demasiado complexo e ineficiente manter e decorar comandos extensos para cada um.
- Optámos por implementar este sistema em Python, utilizando um ficheiro `make.py` para cada subprojeto, juntamente com um ficheiro `make.py` na raiz do projeto. O ficheiro da raiz chama os scripts específicos de cada subprojeto, permitindo uma abordagem centralizada. Também é possível usar diretamente os ficheiros `make.py` de cada subprojeto individualmente.
- Cada ficheiro `make.py` implementa uma dataclass `Args`, que define os diferentes argumentos e modos de compilação para o projeto. Por exemplo, a dataclass do `make.py` da app inclui variáveis como `dev`, `release`, `mobile`, `clean`, entre outras, permitindo uma personalização detalhada da construção do projeto, conforme a necessidade.
- Este sistema modular e flexível facilita o processo de build e torna a gestão das configurações de compilação mais eficiente e intuitiva.

BUILD SYSTEM

```
# app
./make.py --help
# resultado
usage: make.py [-h] [-d | -r] [-m] [-bf] [-c] [-u] [-R] [-n] [--native] [-s] [-k]

App build script

options:
-h, --help show this help message and exit
-d, --dev Run in development mode
-r, --release Build in release mode
-m, --mobile Build the mobile version (requires Android NDK and SDK)
-bf, --build-frontend
Also build the frontend on release
-c, --clean Clean the target directory
-u, --upx Compress the binary with UPX
-R, --run Run the release application
-n, --nightly Build with nightly compiler for further optimizations
--native Build with native CPU target (not recommended for distribution)
-s, --smallest Build with optimizations for size (enables release, nightly and upx)
-k, --keys Create keystore and upload keys
# compilar app no modo mais otimizado para mobile
./make.py -m –s
# início da compilação
```

SISTEMA DE INTERNACIONALIZAÇÃO

- Neste projeto, o frontend é traduzido em português e em inglês. A abordagem utilizada permite gerar as páginas em ambas as línguas durante o processo de compilação, evitando traduções em tempo de execução.
- O sistema copia as páginas traduzidas para a pasta correspondente ao idioma, mantendo a versão em inglês na raiz e a versão em português na pasta "/pt". Além disso, criámos funções para detectar a língua da URL, realizar traduções dinâmicas e ajustar os caminhos das páginas conforme o idioma selecionado.
- Este sistema proporciona uma experiência de navegação fluida e eficiente para os usuários, com tradução automática e navegação correta entre as versões em diferentes idiomas.

SISTEMA DE INTERNACIONALIZAÇÃO

```python
def _copy_files() -> None:
    """This function copies all the pages to the /pt folder"""

    Colors.info("Copying files")
    start = perf_counter()

    entries = os.listdir(f"{CWD}/src/pages")
    for entry in entries:
        if os.path.isfile(f"{CWD}/src/pages/{entry}"):
            copyfile(f"{CWD}/src/pages/{entry}", f"{CWD}/src/pages/pt/{entry}")

    elapsed = perf_counter() - start
    Colors.success(f"Copied files in {elapsed:.2f} seconds")
```

SISTEMA DE INTERNACIONALIZAÇÃO

```ts
import { translations, defaultLang } from "./translations";

export function getLangFromUrl(url: URL) {
  const [, lang] = url.pathname.split("/");
  if (lang in translations) return lang as keyof typeof translations;
  return defaultLang;
}

export function useTranslations(lang: keyof typeof translations) {
  return function t(key: keyof (typeof translations)[typeof defaultLang]) {
    return translations[lang][key] || translations[defaultLang][key];
  };
}

export function useTranslatedPath(lang: keyof typeof translations) {
  return function translatePath(path: string, l: string = lang) {
    return l === defaultLang ? path : `/${l}${path}`;
  };
}
```

TUDO O RESTO

- O código completo do projeto está disponível de forma open source no nosso repositório no GitHub. Através deste repositório, é possível aceder a todos os detalhes da implementação, desde a estrutura do backend e frontend até a configuração dos sistemas e otimizações aplicadas.
- Podem consultar e explorar todo o código em:
- https://github.com/lucascompython/PDSDI-TP

FUTURAS MELHORIAS

- Embora tenhamos alcançado os principais objetivos do projeto, existem várias áreas com potencial de melhoria:
- **Modelo de Machine Learning Avançado**: Substituir o algoritmo simples por modelos de machine learning, como sistemas de recomendação ou redes neurais, para oferecer sugestões de outfits mais personalizadas e precisas.
- **Aprimoramento da Experiência do Utilizador (UX)**: Tornar o frontend mais dinâmico e intuitivo, com filtros avançados e maior responsividade.
- **Expansão dos Atributos das Peças de Roupa**: Permitir adicionar mais informações, como estação do ano ou ocasiões específicas, para enriquecer a base de dados e melhorar as sugestões.
- **Integração de Notificações Personalizadas**: Enviar recomendações ou lembretes baseados em preferências e condições climáticas.
- **Compilação para Mais Plataformas**: Expandir para plataformas como iOS e macOS, com pequenas alterações no sistema de build.
- **Conteinerização com Docker**: Implementar Docker para melhorar a portabilidade, garantir execução consistente e facilitar a escalabilidade do sistema.

VER PROJETO EM EXECUÇÃO