# **Documentação do Projeto Galapagos Capital - Project Onboarding Switzerland**

**Autor: Raul Lemelle**

**01/09/2024**

## **Índice**

1. **Visão Geral do Projeto**
2. **Arquitetura do Sistema**
3. **Instalação e Configuração**
4. **Uso**
5. **Estrutura do Projeto**
6. **Testes**
7. **Tecnologias Utilizadas**
8. **Considerações Finais**

## **1. Visão Geral do Projeto**

O **Project Onboarding Switzerland** da Galapagos Capital é uma aplicação web desenvolvida para facilitar o processo de Onboarding , permitindo que os bankers interajam através de uma interface simples e intuitiva. O sistema oferece funcionalidades de registro de novos clientes e consulta de status de formulários relacionados ao Onboarding.

## **2. Arquitetura do Sistema**

A aplicação segue a arquitetura MVC (Model-View-Controller) e é baseada no framework Flask. Os principais componentes incluem:

* **Rotas (routes.py)**: Controlam a navegação dentro da aplicação, mapeando URLs para funções específicas.
* **Templates (templates/)**: Arquivos HTML que definem a interface do usuário.
* **Serviços (services.py)**: Funções auxiliares que lidam com a lógica de negócio e interações com APIs externas.
* **Configurações (config.py)**: Define as configurações da aplicação, incluindo variáveis de ambiente e chaves da API.
* **Blueprints**: Modularizam a aplicação, permitindo que diferentes partes do sistema sejam organizadas de maneira separada.

## **3. Instalação e Configuração**

### **Pré-requisitos**

* Python 3.x
* Virtualenv

### **Passos para Instalação**

1. **Clone o repositório:**

bash

git clone <https://github.com/Raul-Lemelle/status_form_project_onboarding_switzerland.git>cd seu\_repositorio

1. **Crie e ative um ambiente virtual:**

bash

python -m venv venv  
source venv/bin/activate # Para Linux/MacOS  
venv\Scripts\activate # Para Windows

1. **Instale as dependências:**

bash

pip install -r requirements.txt

1. **Configuração:**

Configure as variáveis de ambiente necessárias no arquivo .env, incluindo chaves da API e configurações específicas do Flask:

makefile

PIPEFY\_API\_URL=https://api.pipefy.com/graphql

PIPEFY\_API\_TOKEN= <Token gerado no Pipefy>

REGISTER\_NEW\_URL= <URL do Star Form Initial Registration>

INITIAL\_REGISTRATION\_PIPE\_ID= <ID do Pipe>

BENEFICIARY\_KYC\_PIPE\_ID= <ID do Pipe>

RISK\_PROFILE\_PIPE\_ID= <ID do Pipe>

1. **Inicie a aplicação:**

bash

phyton run.py

A aplicação estará disponível em <http://127.0.0.1:5000>.

## **4. Uso**

A aplicação oferece as seguintes funcionalidades principais:

* **Página Inicial (index.html)**: Apresenta opções para registrar um novo cliente ou consultar o status de formulários.
* **Página de Status (status.html)**: Permite a seleção de um "Banker" e de um "Client" para visualizar o status dos formulários relacionados.

### **Endpoints Principais**

* GET /: Página inicial.
* GET /status: Página de consulta de status.

## **5. Estrutura do Projeto**

A estrutura de diretórios do projeto é organizada da seguinte maneira:

project\_name/  
│  
├── app/  
│ ├── \_\_init\_\_.py # Inicialização do aplicativo Flask  
│ ├── routes.py # Definição das rotas da aplicação  
│ ├── services.py # Lógica de negócio e interações API  
│ └── config.py # Configurações da aplicação  
│  
├── static/ # Arquivos estáticos (CSS, JS, imagens)  
│ ├── css/  
│ │ └── styles.css # Arquivo de estilos CSS  
│ ├── js/  
│ │ └── scripts.js # Arquivo JavaScript  
│ └── img/  
│ └── logo.png # Logotipo do projeto  
│  
├── templates/ # Templates HTML  
│ ├── layout.html # Template base para páginas HTML  
│ ├── index.html # Página inicial  
│ └── status.html # Página de status  
│  
├── tests/ # Testes unitários  
│ ├── test\_routes.py # Testes para as rotas da aplicação  
│ ├── test\_services.py # Testes para os serviços da aplicação  
│ └── \_\_init\_\_.py # Inicialização dos testes  
│  
├── .env # Arquivo de configuração de ambiente  
├── .gitignore # Arquivo gitignore  
├── README.md # Documentação do projeto  
├── requirements.txt # Arquivo de dependências  
└── run.py # Script para iniciar a aplicação

## **6. Testes**

O projeto inclui testes unitários para verificar a funcionalidade das rotas e dos serviços principais. Os testes estão localizados na pasta tests/.

### **Executando os Testes**

Para executar os testes, use o seguinte comando:

bash

python -m unittest discover -s tests

Se todos os testes forem executados com sucesso, você verá uma saída indicando que todos os testes passaram.

## **7. Tecnologias Utilizadas**

* **Flask**: Framework web para Python.
* **Jinja2**: Motor de templates para gerar HTML a partir de variáveis dinâmicas.
* **HTML/CSS/JavaScript**: Tecnologias de front-end para construir a interface do usuário.
* **Unittest**: Biblioteca de testes integrada ao Python.
* **API do Pipefy**: Utilizada para interagir com o sistema de gestão de processos Pipefy.

## **8. Considerações Finais**

Este projeto foi desenvolvido para proporcionar uma solução para o Onboarding de clientes na Galapagos Capital. Ele pode ser expandido para incluir novas funcionalidades ou adaptado para diferentes cenários de negócio.