**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS  
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA**

**Pós-graduação *Lato Sensu* em Desenvolvimento Web Full Stack**

**Caio Lúcio Sousa Duarte**

**ANALISADOR DE CADASTRO (REGZER)**

Belo Horizonte

2021

**Caio Lúcio Sousa Duarte**

**ANALISADOR DE CADASTRO (REGZER)**

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Desenvolvimento Web Full Stack como requisito parcial à obtenção do título de especialista.

Belo Horizonte

2021

**RESUMO**

Tendo em vista a necessidade de preenchimento manual de formulários em papel para envio e análise de informações cadastrais/financeiras de pessoas físicas e jurídicas, a aplicação REGZER possibilita o envio digital dos dados e documentos digitalizados a serem enviados pelo Usuário e validados por um Analista de Cadastro. Dessa forma, o tempo de envio das informações é reduzido, o preechimento torna-se mais rápido e confiável, a análise conta com ferramentas integradas com a aplicação e a presença física é dispensada. Esse é o contexto de empresas que necessitam de aprovação cadastral como: imobiliárias, financeiras e seguradoras. A aplicação foi desenvolvida para web na arquitetura Back-end for Front-end e sua usabilidade/responsividade foi testada em dispositivos móveis variados e desktop.

Javascript. TypeScript. Node.js. React. Typeorm. AWS.

**SUMÁRIO**

[**1. Apresentação 5**](#_Toc445198572)

[1.1. Contexto 5](#_Toc445198573)

[1.1. Público alvo 5](#_Toc445198574)

[1.2. Requisitos 5](#_Toc445198575)

[2. Modelagem](#_Toc445198576) 7

[2.1. Atores](#_Toc445198578) 7

[2.2. Interfaces](#_Toc445198580) 9

[2.3. Diagrama de entidades e relacionamento](#_Toc445198581) 13

[3. Projeto](#_Toc445198582) 14

[3.1. Arquitetura de *software*](#_Toc445198583) *e da informação*14

[4. Testes de usabilidade](#_Toc445198585) 16

[5. URL](#_Toc445198586) 21

[5.1. Aplicação web](#_Toc445198587) 21

[5.2. Repositório código-fonte](#_Toc445198588) 21

[REFERÊNCIAS](#_Toc445198589) 22

# 1. Apresentação

## 1.1. Contexto

A aplicação foi desenvolvida para resolver a dependência de meios físicos para apresentar e analisar informações cadastrais.

O objetivo é possibilitar o envio digital de dados cadastrais por pessoas físicas e jurídicas, para serem submetidas a uma equipe de analistas responsáveis pela análise dos dados enviados. É o contexto de imobiliárias, financeiras, bancos, empresas de seguros, enfim, locais que exigem o processo de envio de documentos e preenchimento de formulários em papel visando a análise cadastral.

## 1.1. Público alvo

O perfil de usuários da aplicação são pessoas que utilizam celular e que estão acostumadas a preencher informações financeiras para obtenção de crédito ou serem fiadores ou avalistas. E, ainda, pessoas acostumadas a analisar as informações de cunho cadastral e financeiro de pessoas físicas e jurídicas.

## 1.2. Requisitos

A aplicação prevê o funcionamento na web. São requisitos não funcionais da aplicação por categoria:

Categoria Desempenho

* A infraestrutura deve prover espaço suficiente para a API e o banco de dados. Dessa forma, o espaço do banco de dados tem que ser de no mínimo 5 GB
* Deve ser fornecido um espaço para armazenamento de arquivos separado do banco de dados e da API. Tendo em vista as regras de limitação do tamanho do arquivo, considera-se 10 GB.
* A aplicação deve utilizar uma cache do banco de dados para as tabelas críticas como a de usuário

Categoria Disponibilidade

* Deve ser disponibilizado um ambiente de integração contínua que avalie a qualidade do deploy antes para que a aplicação não pare. Dessa forma, o deploy tem que ser acompanhado por testes.

Categoria Segurança

* A senha dos usuários deve ser armazenada de forma criptografada
* O sistema deve autenticar usuários com JWT (Json Web Token).

Categoria Interoperabilidade

* O sistema deve ser desenvolvido com as camadas cliente e servidor separadas. Sendo o servidor no formato de uma API rest e possibilitando o acesso de vários tipos de clientes.

Categoria Usabilidade

* A aplicação de ser responsiva de forma a possibilitar o acesso por disponsitivos móveis e computadores.

Categoria Compatibilidade

* A aplicação deve ser compatível com navegadores que leem javascript versão ECMAScript 2015 (https://262.ecma-international.org/6.0/).

Categoria Confiabilidade

* As operações de banco de dados devem ser sempre transacionais de forma a garantir a consistência dos dados
* O banco de dados deve ser modelado de forma assegurar as chaves primárias e estrangeiras das entidades

São requisitos funcionais da aplicação:

* Deve permitir cadastrar um novo Usuário, por padrão com perfil comum.
* Deve permitir enviar e-mail de recuperação de senha.
* Deve ser possível “logar” com e-mail e senha.
* Deve ser possível ao “logar”, receber um token JWT e cadastrar um refresh token para autenticação.
* Deve ser possível solicitar novamente a confirmação dos dados de cadastro por email.
* O Usuário só pode logar após a validação do cadastro pelo Administrador.
* Deve ser possível o Usuário enviar seus dados cadastrais para sem aprovados
* O Administrador pode listar e alterar os dados dos Usuários não administradores, bem como enviar o e-mail de confirmação de cadastro

# 2. Modelagem

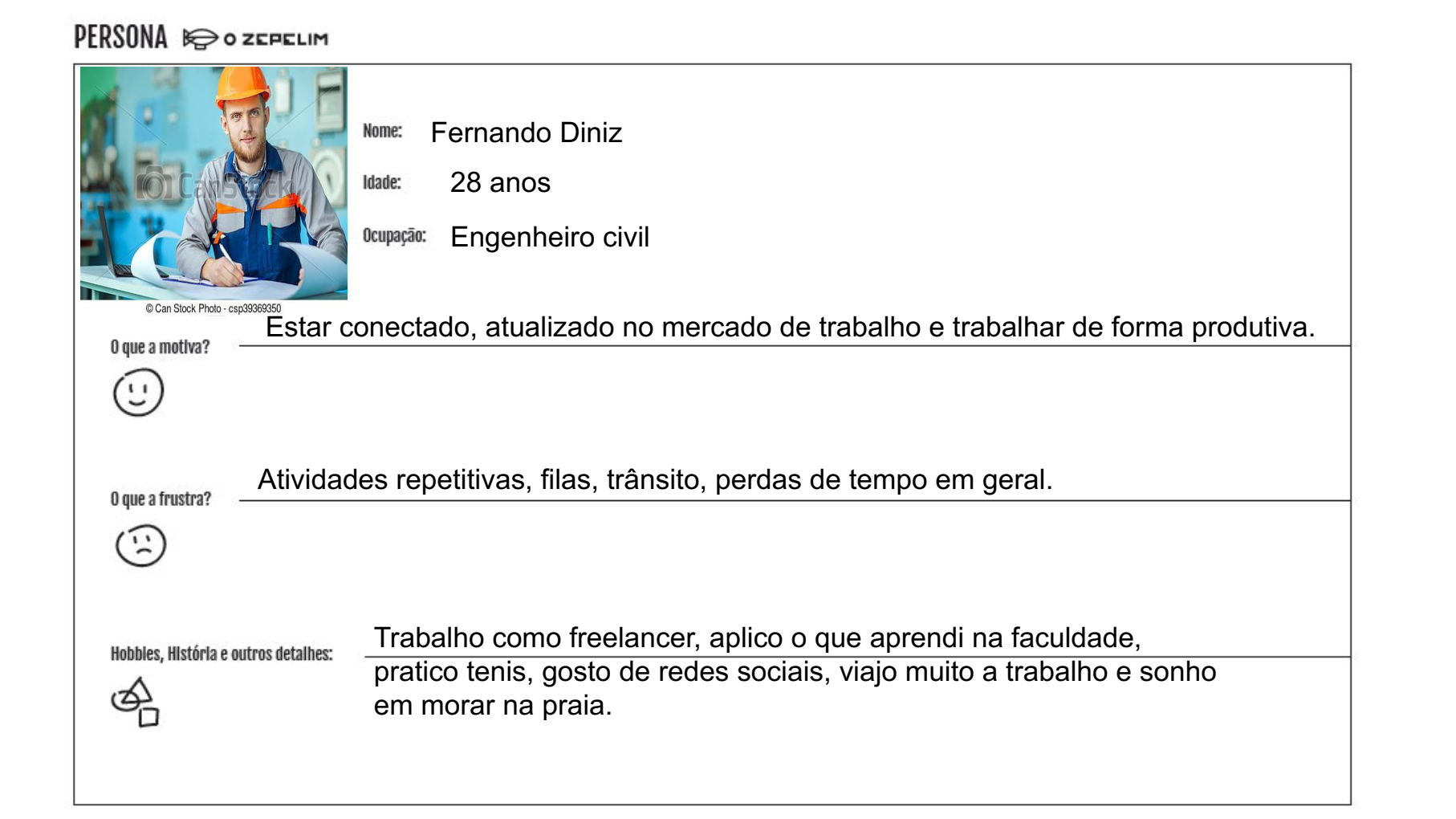
## 2.1. Atores

São atores que interagem com o sistema:

Usuário: pessoas físicas e jurídicas que desejam se cadastrar e enviar seus dados cadastrais de forma digital.

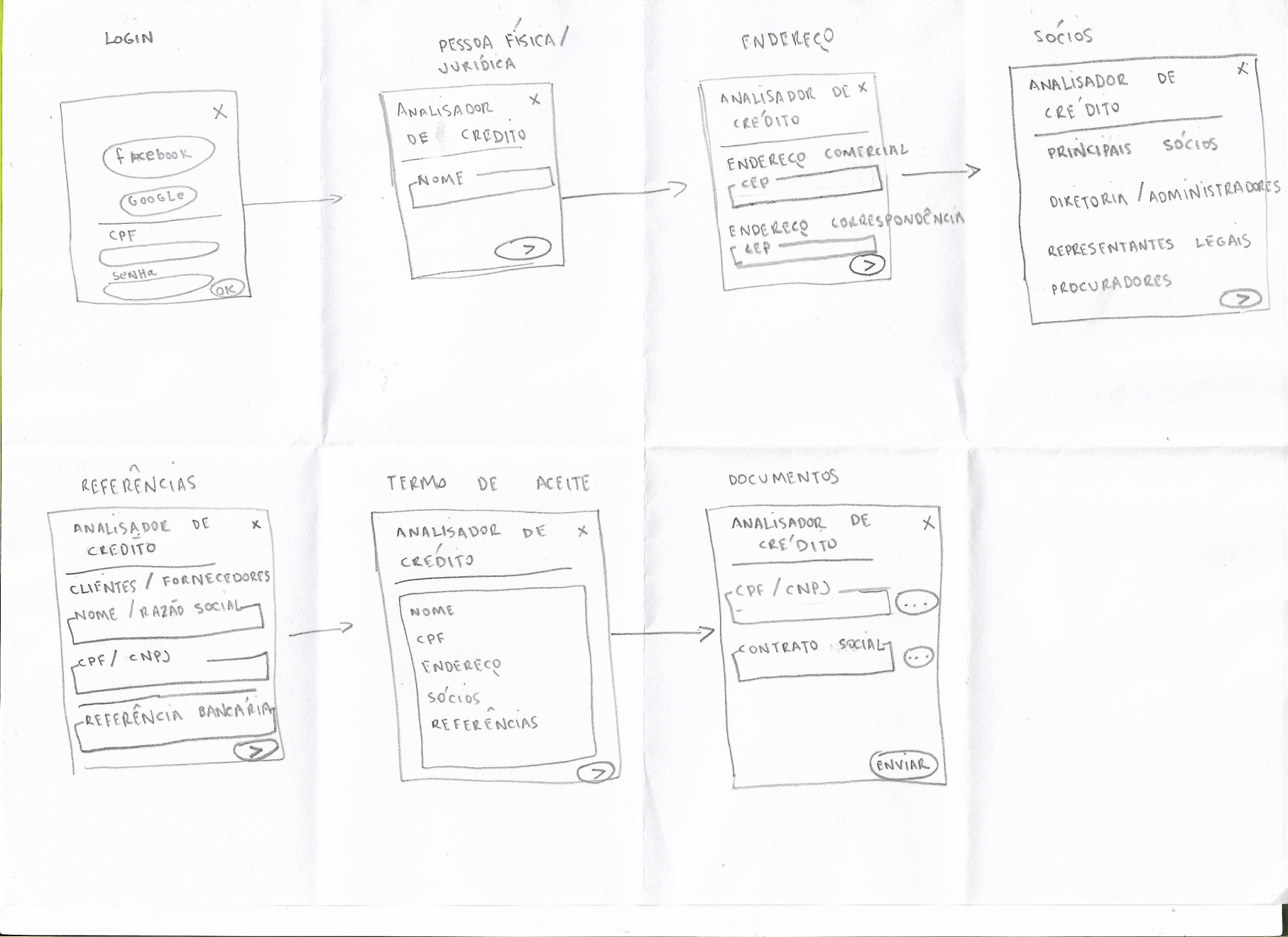
Analista: são os administradores do sistema que fazem a validação de acesso do Usuário recém cadastrado e validam os dados enviados por estes.

2.1.1. Definição das personas:

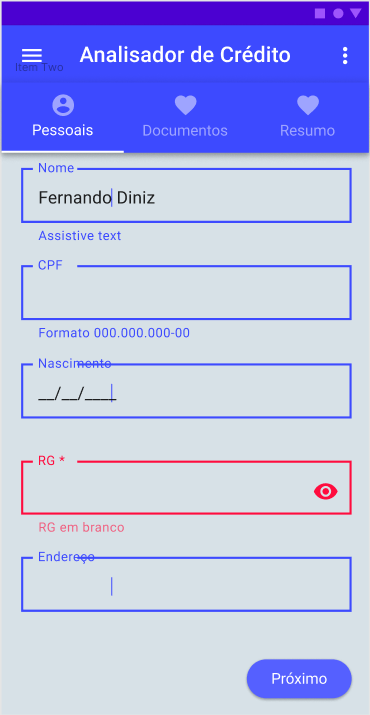


## 2.2. Interfaces

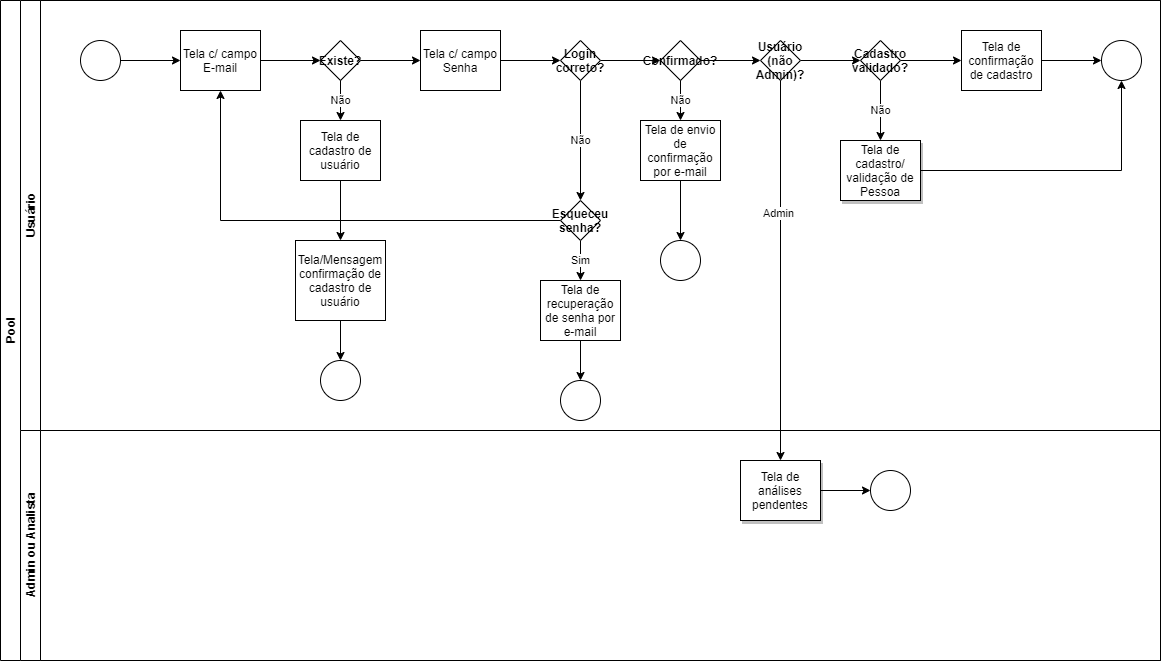
2.2.1. Mock-ups com o Crazy 8:



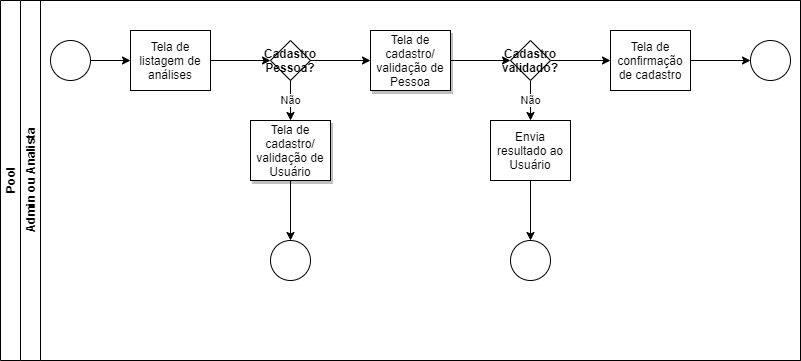
2.2.2. Layout da primeira versão:



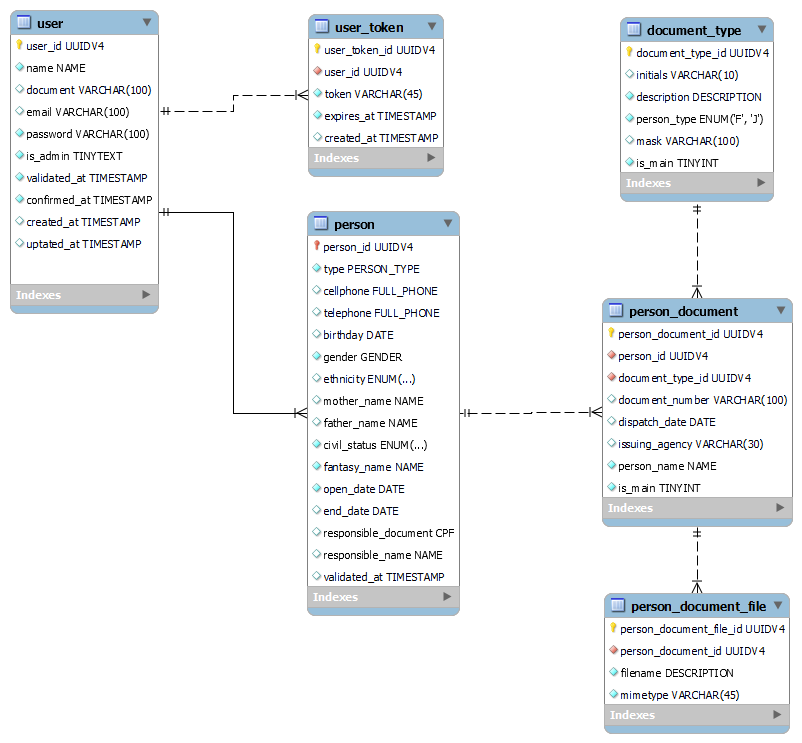
2.2.3. Diagrama com o comportamento da interface para o Usuário:



2.2.4. Diagrama com o fluxo da interface para o Analista:



## 2.3. Diagrama de entidades e relacionamentos



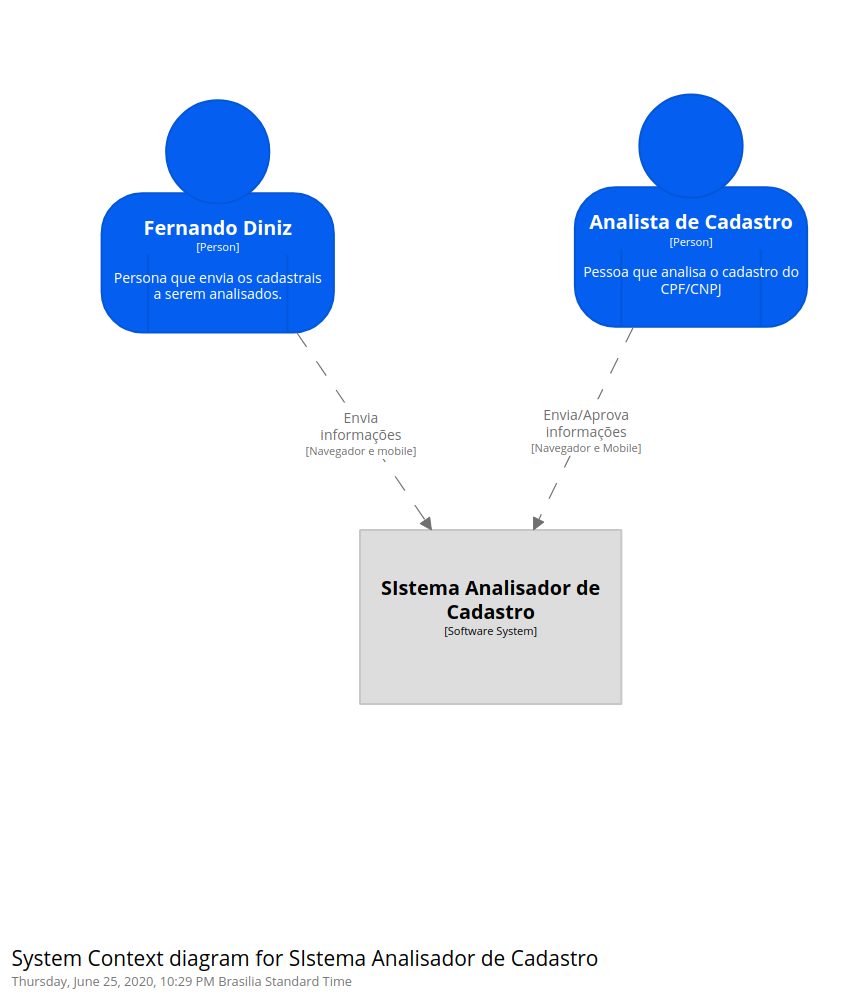
As entidades são o resultado do mapeamento objeto relacional (ORM) feito nas classes: User, Token, Person, Individual, Company, PersonDocument, PersonDocumentFile e DocumentType.

A entidade usuários (user) contém a chave primária da entidade pessoa (person), bem como os campos padrões como nome e e-mail. A entidade pessoa (person) é mapeada no padrão STI (Single Table Inheritance), onde classes Individual (Pessoa Física) e Company (Pessoa Jurídica) compõe a mesma tabela e são diferenciados pelo campo tipo (type).

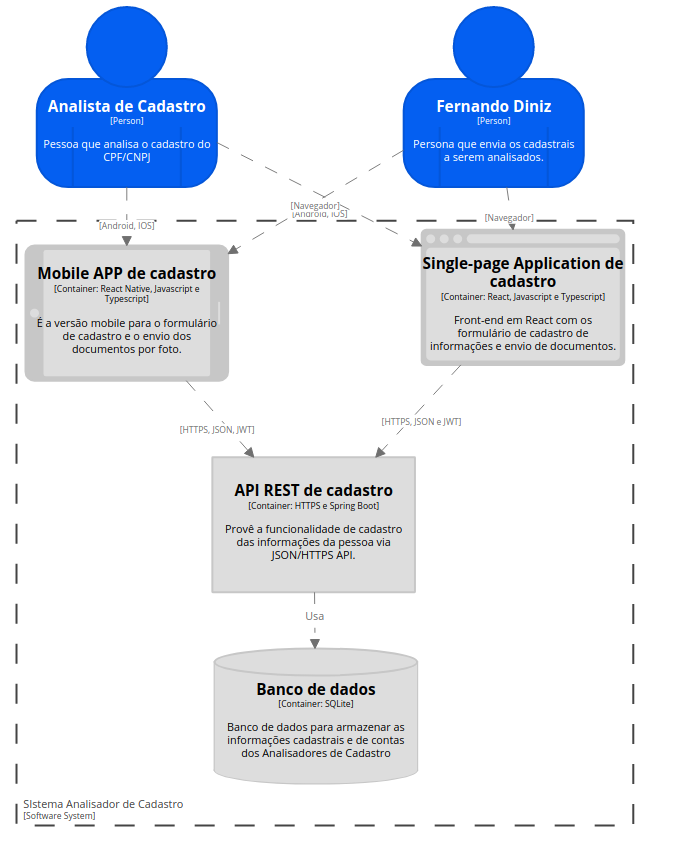
# 3. Projeto

## 3.1. Arquitetura de *software*

3.1.1. C4 Level 1:



3.1.2. C4 Level 2:



O padrão arquitetural adotado é Back-end for Front-end (BFF) de forma a dar suporte a vários tipos e múltiplos clientes. O back-end é uma API REST feita em Node.js com typescript e o front-end é um SPA (Single Page Application) em React.

O back-end utiliza o banco de dados SQLite para gravação dos dados acessado com a ferramenta de mapeamento objeto relacional (ORM) Typeorm (<https://typeorm.io/>) no padrão de Data Mapper (https://typeorm.io/#/active-record-data-mapper/what-is-the-data-mapper-pattern).

Para facilitar o desenvolvimento, o back-end foi modularizado em: users, people e analysis.

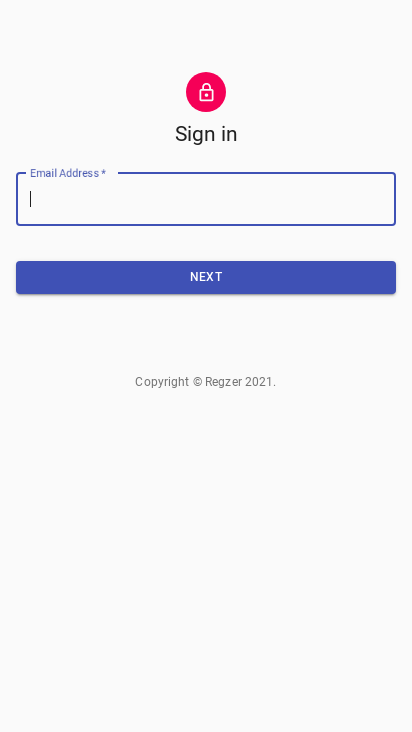
O front-end em React foi feito em typescript e componentizado de forma a reutilizar os recursos de tela, providers e hooks, respectivamente nas pastas components/, pages/ ; providers/ e hooks.

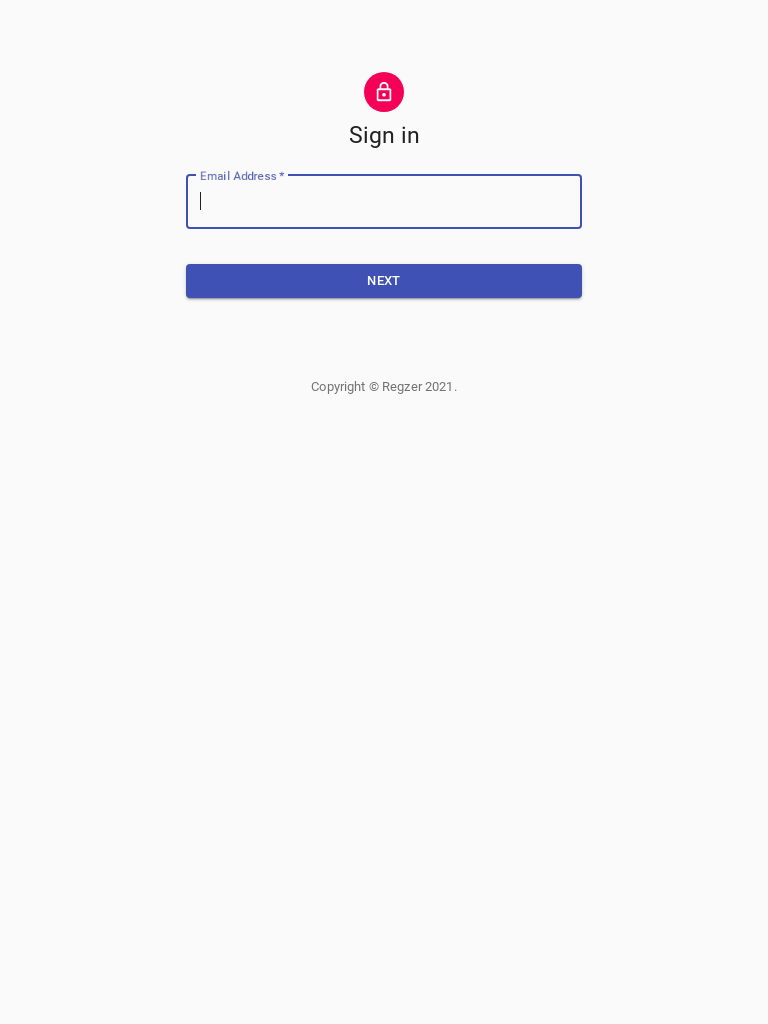
O back-end está hospedado na Amazon (AWS) no serviço ECS com a imagem do Ubuntu 20 e utiliza os serviços AWS S3 para armazenamento de arquivos estáticos, AWS SES com SMTP para envio de e-mails, AWS Route 53 para direcionamento de DNS, NGINX para proxy reverso, PM2 para gerenciamento de processos com Node.js. O deploy do back-end é feito com integração contínua via github actions e sua configuração está no arquivo /.github/workflows/main.yml.

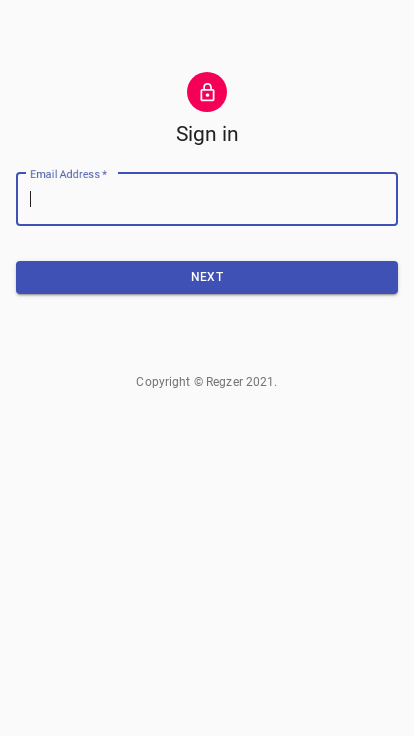
O front-end está hospedado na Amazon (AWS) no serviço AWS Amplify e o deploy é feito com integração contínua. Esta configuração está no arquivo /amplify.yml.

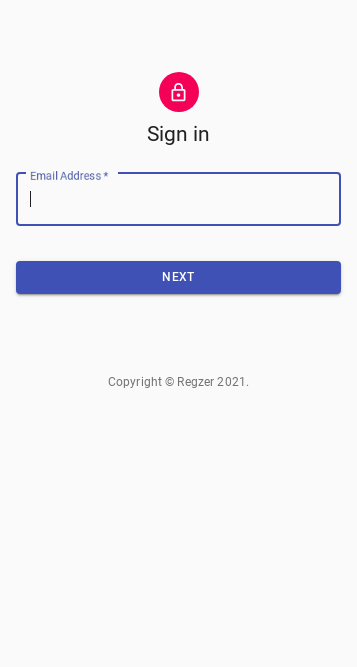
# 4. Testes

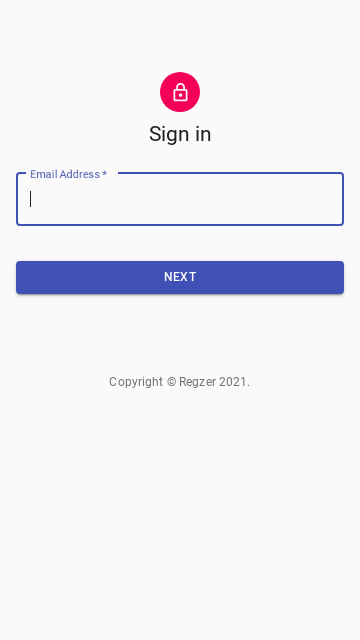
4.1. Testes de usabilidade da interface e seus dispositivos:

Google Pixel

iPad Air 2

iPhone 7 Plus

iPhone 8

Samsung S7

# 5. URLs

## 5.1. Aplicação web

A aplicação web está hospedada em [https://www.regzer.com.br](https://www.regzer.com.br/). O usuário padrão pode ser cadastrado com qualquer e-mail, ao cadastrá-lo deve-se entrar como Analista e validá-lo. O usuário admin ou Analista padrão é o [admin@regzer.com.br](mailto:admin@regzer.com.br) senha root. Ao validar o usuário, um e-mail é encaminhado ao Usuário para confirmação e acesso ao sistema.

## 5.2. Repositório código-fonte

O repositório código fonte está no github no endereço: https://github.com/caiosduarte/dwfs-pia-regzer.

# REFERÊNCIAS

TYPEORM https://typeorm.io/

Amazon Web Services https://aws.amazon.com/pt/

React https://pt-br.reactjs.org/

Node.js https://nodejs.org/