

**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 preenchimento 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

<b>1. Apresentação.....</b>	<b>5</b>
1.1. Contexto.....	5
1.1. Público alvo.....	5
1.2. Requisitos.....	5
<b>2. Modelagem.....</b>	<b>7</b>
2.1. Atores.....	7
2.2. Interfaces .....	9
2.3. Diagrama de entidades e relacionamento .....	13
<b>3. Projeto.....</b>	<b>14</b>
3.1. Arquitetura de <i>software e da informação</i> .....	14
<b>4. Testes de usabilidade .....</b>	<b>16</b>
<b>5. URL.....</b>	<b>21</b>
5.1. Aplicação web.....	21
5.2. Repositório código-fonte.....	21
<b>REFERÊNCIAS.....</b>	<b>22</b>

## **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 dispositivos 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:

#### PERSONA O ZEPELIM



© Can Stock Photo - csp39369350

Nome: Fernando Diniz  
Idade: 28 anos  
Ocupação: Engenheiro civil

O que a motiva?



Estar conectado, atualizado no mercado de trabalho e trabalhar de forma produtiva.

O que a frustra?



Atividades repetitivas, filas, trânsito, perdas de tempo em geral.

Hobbies, História e outros detalhes:

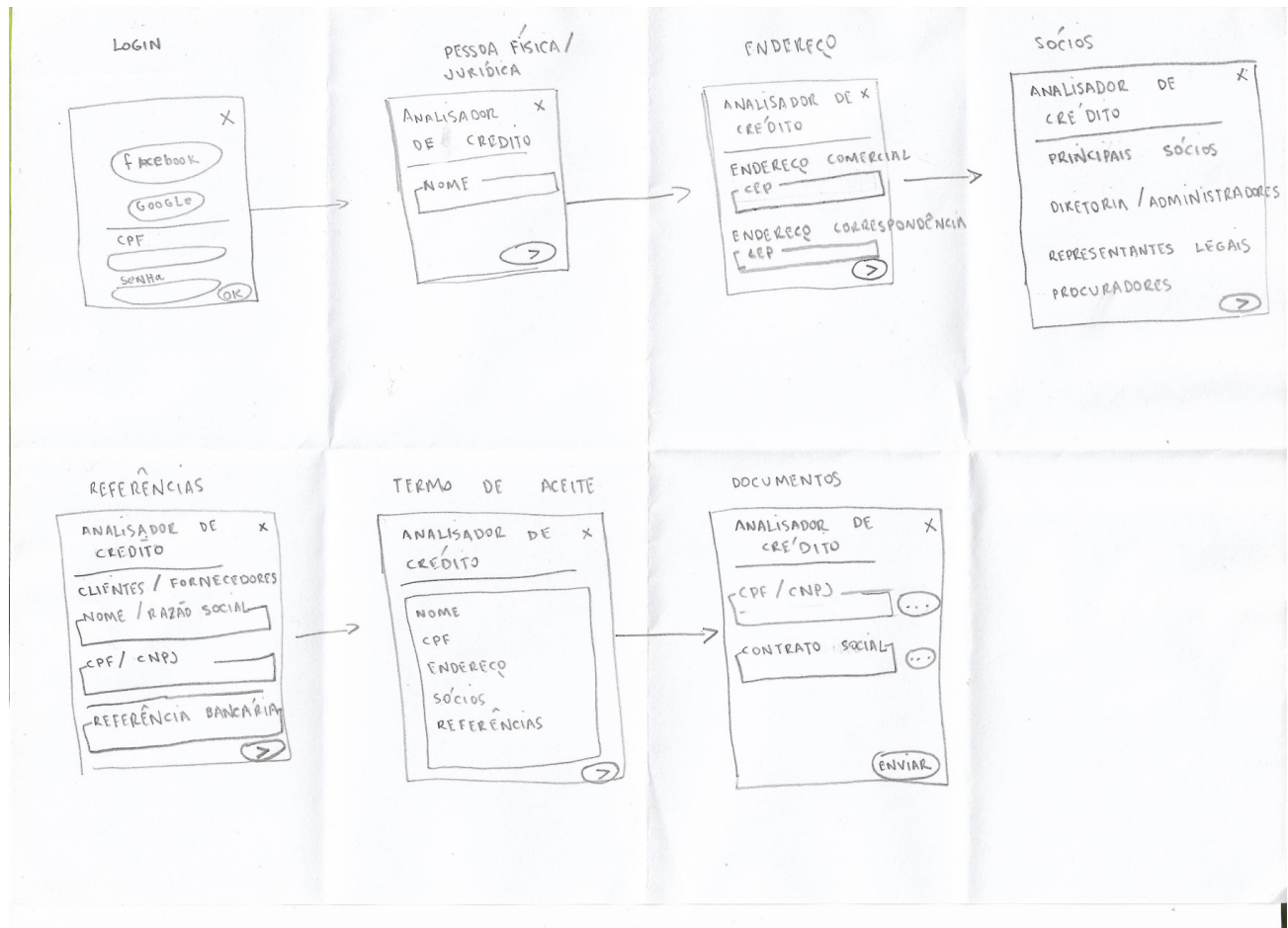


Trabalho como freelancer, aplico o que aprendi na faculdade, pratico tenis, gosto de redes sociais, viajo muito a trabalho e sonho em morar na praia.



## 2.2. Interfaces

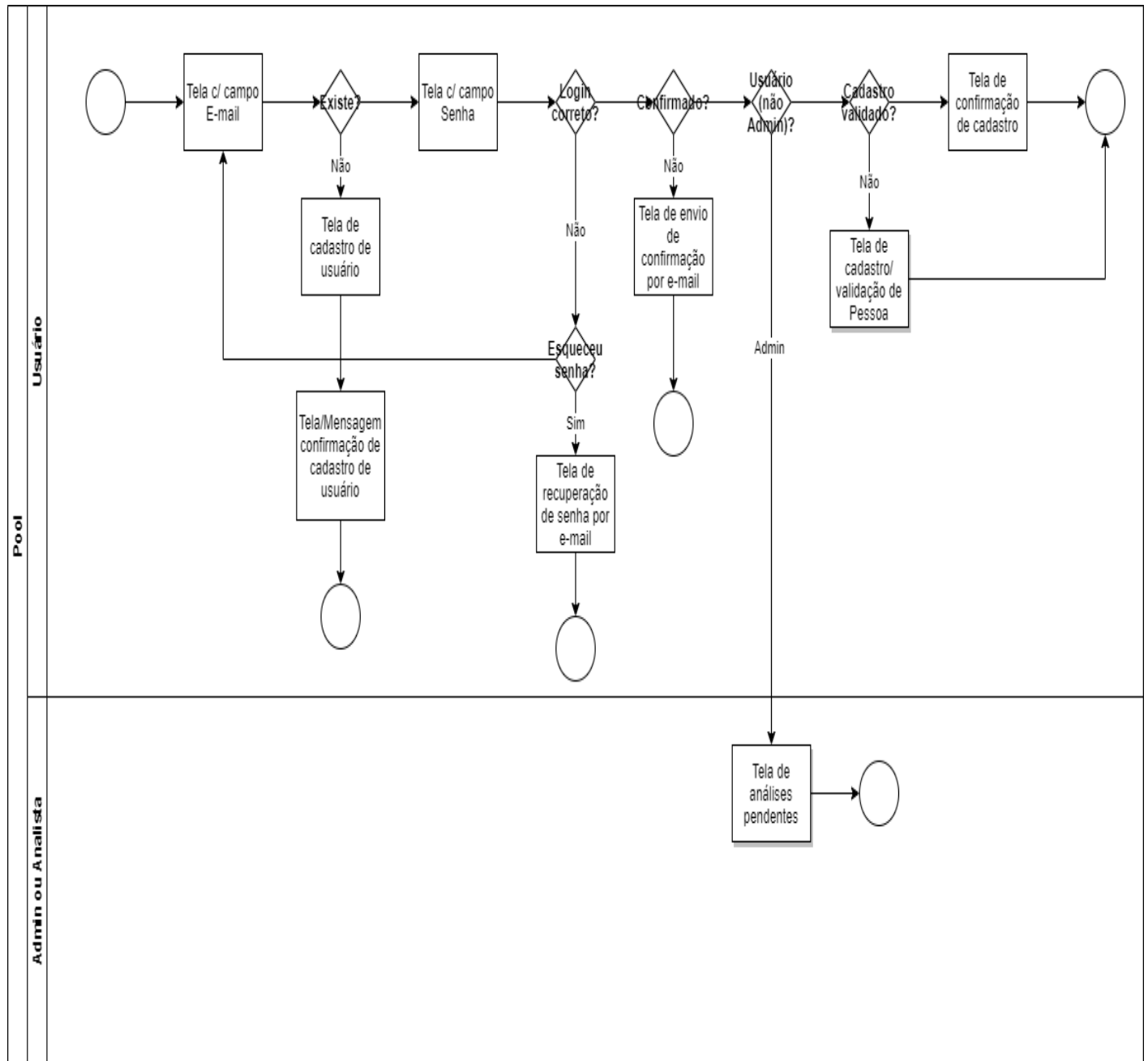
### 2.2.1. Mock-ups com o Crazy 8:



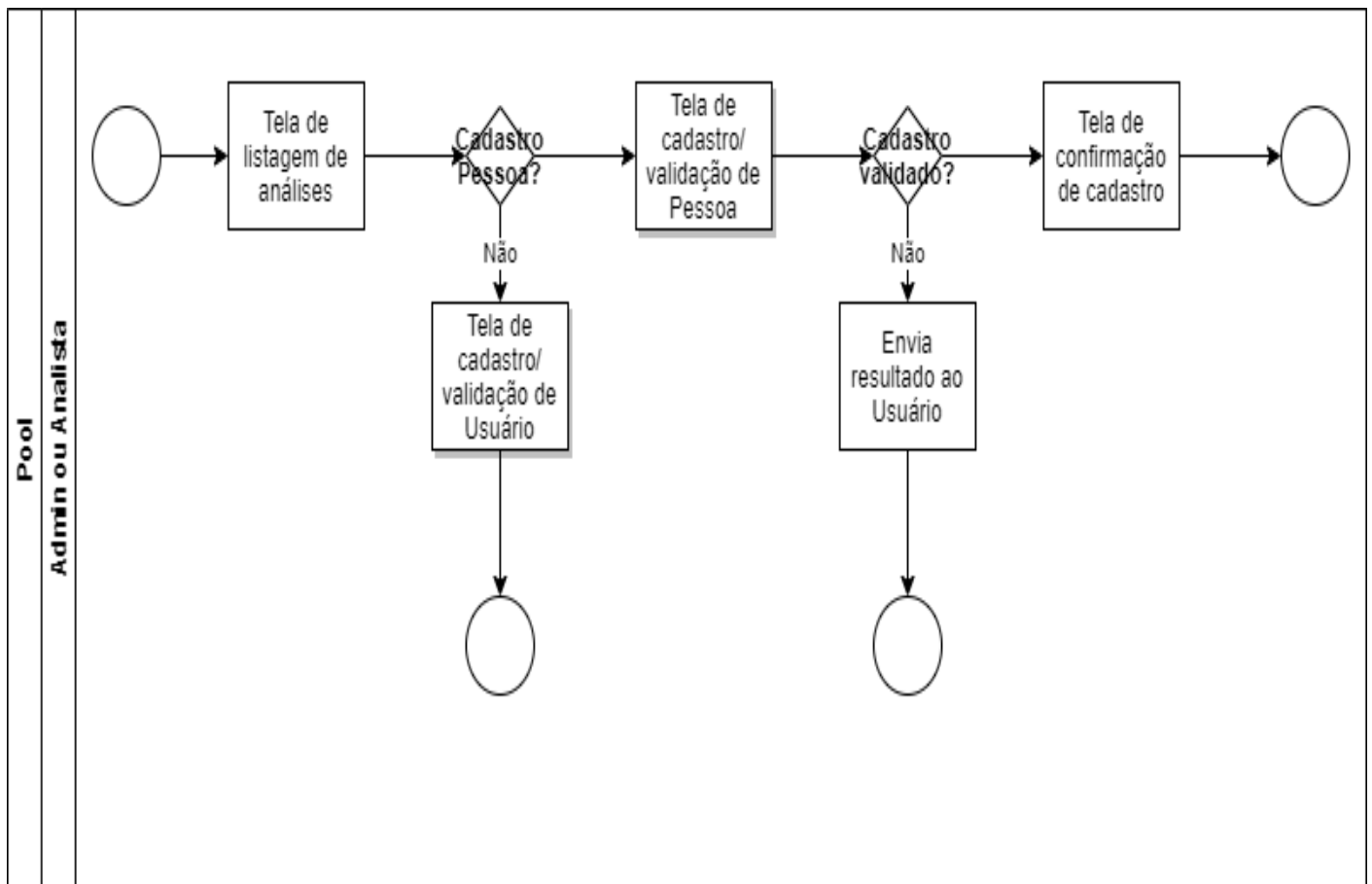
## 2.2.2. Layout da primeira versão:

The screenshot displays the 'Analizador de Crédito' application interface. At the top, there is a purple header bar with a hamburger menu icon on the left, the title 'Analizador de Crédito' in the center, and a vertical ellipsis icon on the right. Below the header is a blue navigation bar with three tabs: 'Pessoais' (selected, with a person icon), 'Documentos' (with a heart icon), and 'Resumo' (with a heart icon). The main content area is light gray and contains several input fields. The first field is labeled 'Nome' and contains the text 'Fernando Diniz'. Below it is a label 'Assistive text'. The next field is labeled 'CPF' and is empty, with a hint 'Formato 000.000.000-00' below it. The following field is labeled 'Nascimento' and contains a date separator '\_\_\_/\_\_\_/\_\_\_'. Below that is a field labeled 'RG \*' which is empty and has a red border; a red eye icon is visible on the right side of the field, and the text 'RG em branco' is shown below it. The final field is labeled 'Endereço' and is empty. At the bottom right of the form is a blue button with the text 'Próximo'.

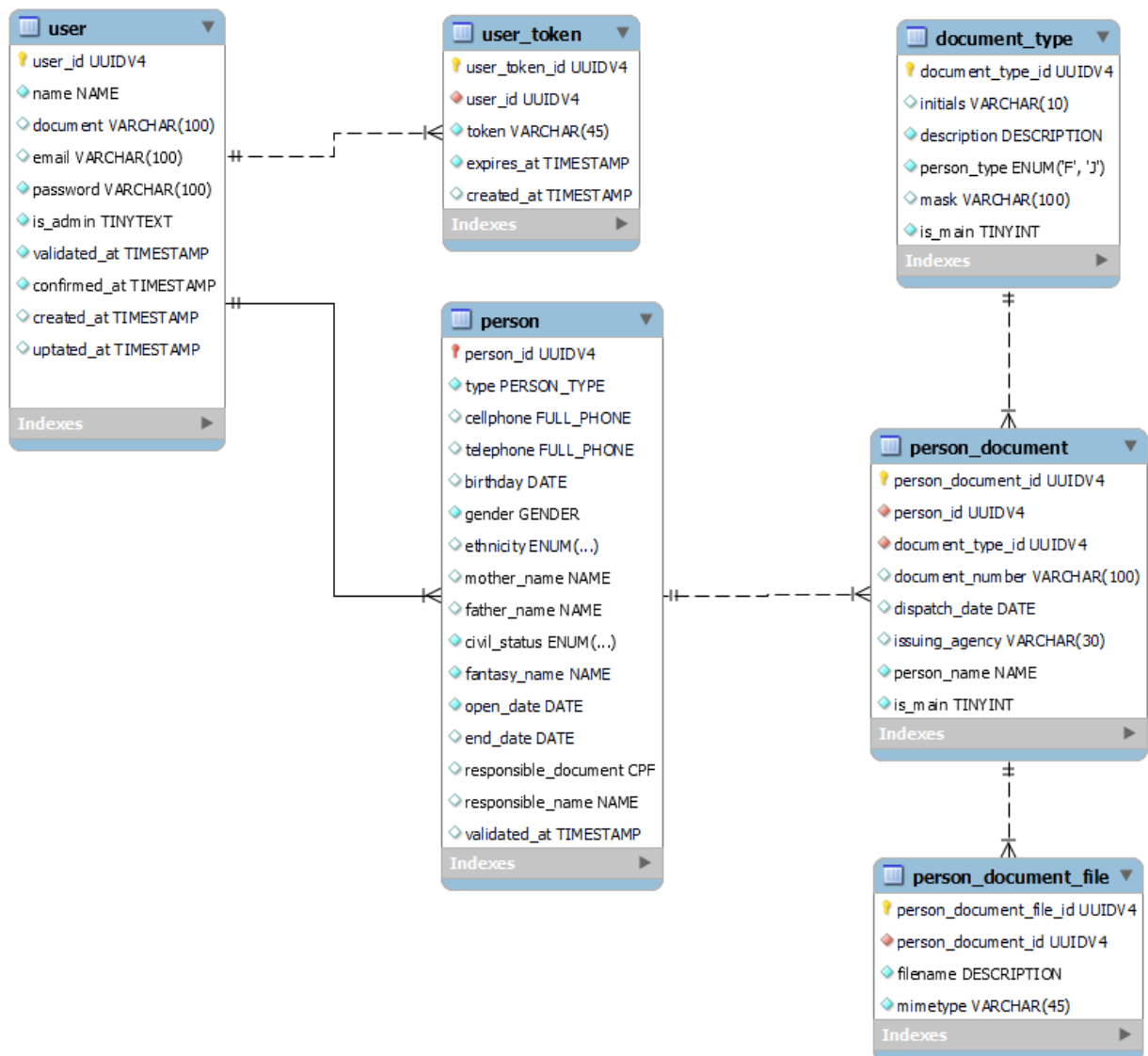
## 2.2.3. Diagrama com o comportamento da interface para o Usuário:



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



### 2.3. Diagrama de entidades e relacionamentos



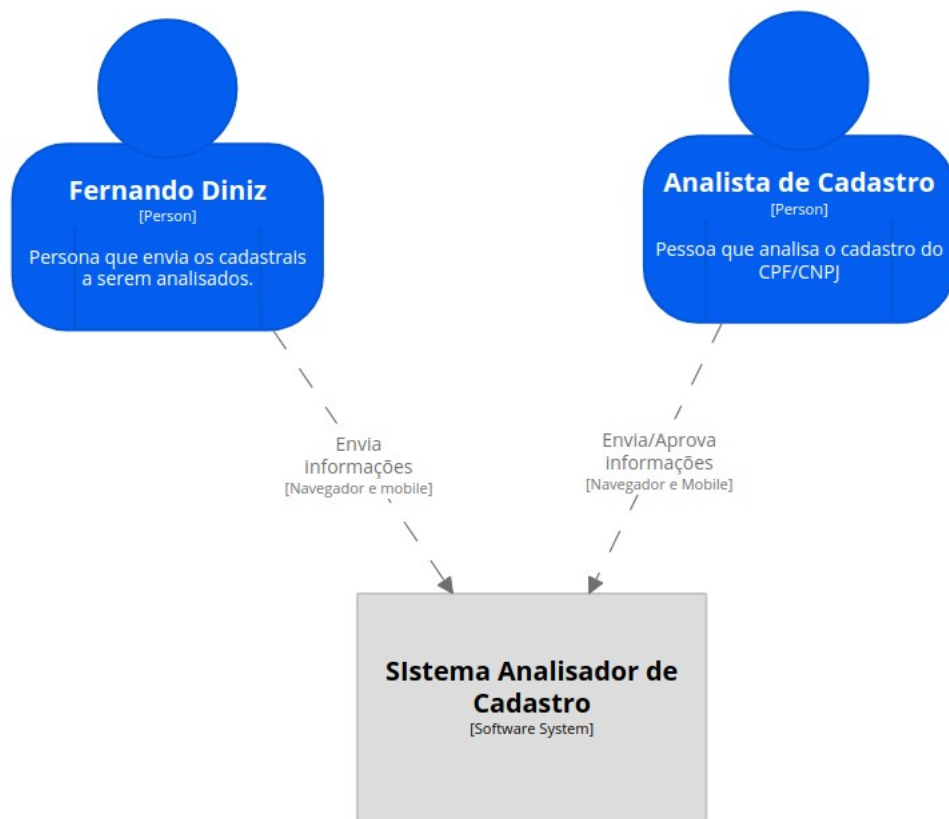
As entidades são o resultado do mapeamento objeto relacional (ORM) feito nas classes: User, Token, Person, Individual, Company, PersonDocument, PersonDocumentFile e DocumentType.

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*

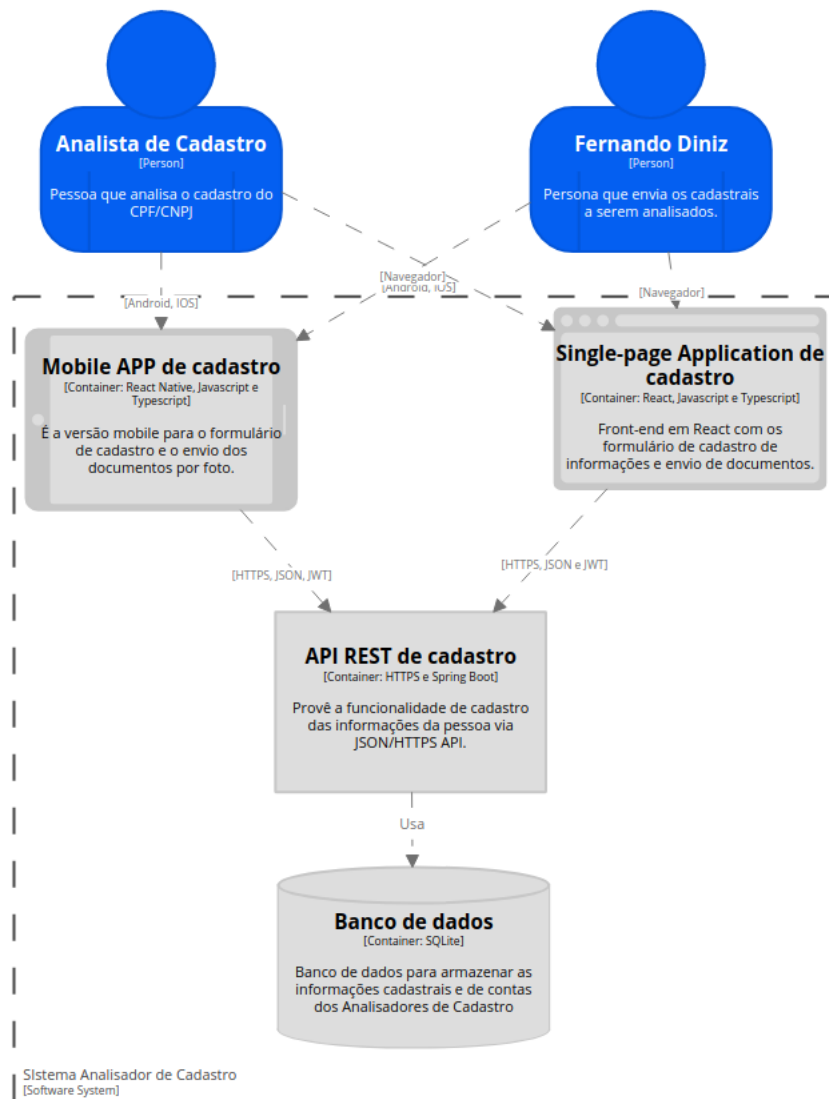
##### 3.1.1. C4 Level 1:



System Context diagram for Sistema Analisador de Cadastro

Thursday, June 25, 2020, 10:29 PM Brasilia Standard Time

##### 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:**





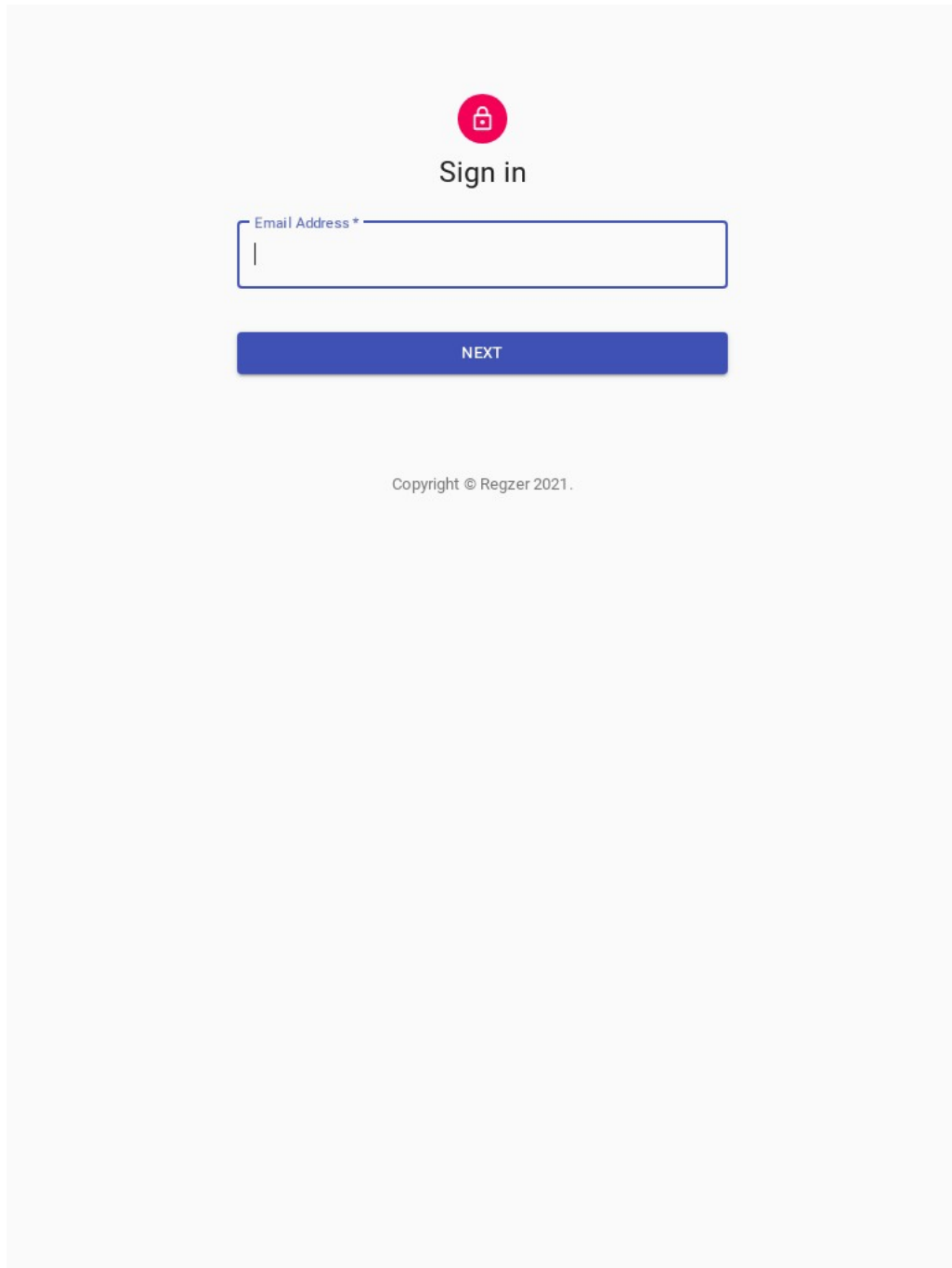
Sign in

Email Address \*

NEXT

Copyright © Regzer 2021.

Google Pixel



A screenshot of a sign-in screen on an iPad Air 2. The screen has a light gray background. At the top center is a red circular icon with a white padlock. Below it is the text "Sign in" in a dark gray font. Underneath is a white rectangular input field with a thin blue border. The text "Email Address\*" is in the top left corner of the field, and a vertical cursor is on the left side. Below the input field is a solid blue rectangular button with the word "NEXT" in white capital letters. At the bottom center of the screen is the text "Copyright © Regzer 2021." in a small, dark gray font.

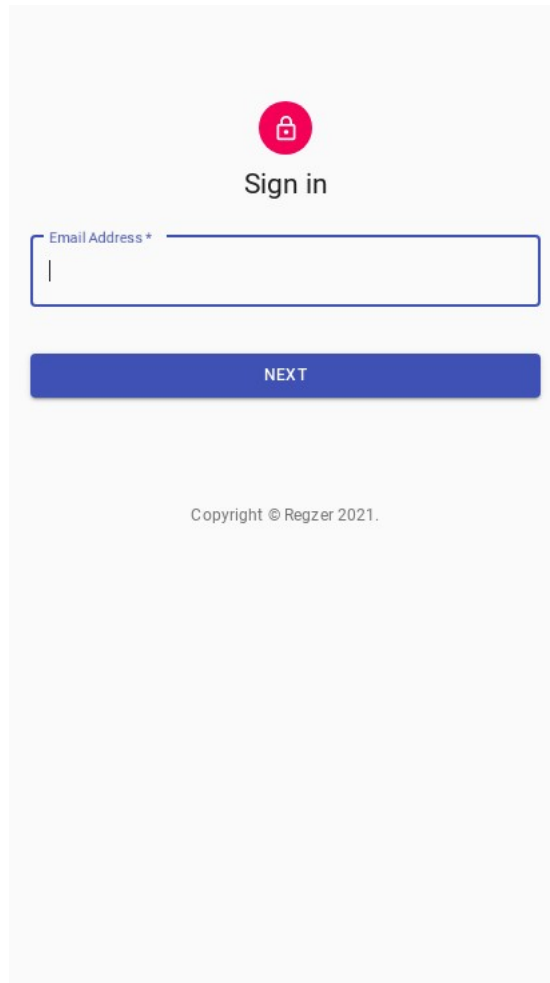
Sign in


Email Address\*

NEXT

Copyright © Regzer 2021.

iPad Air 2





Sign in

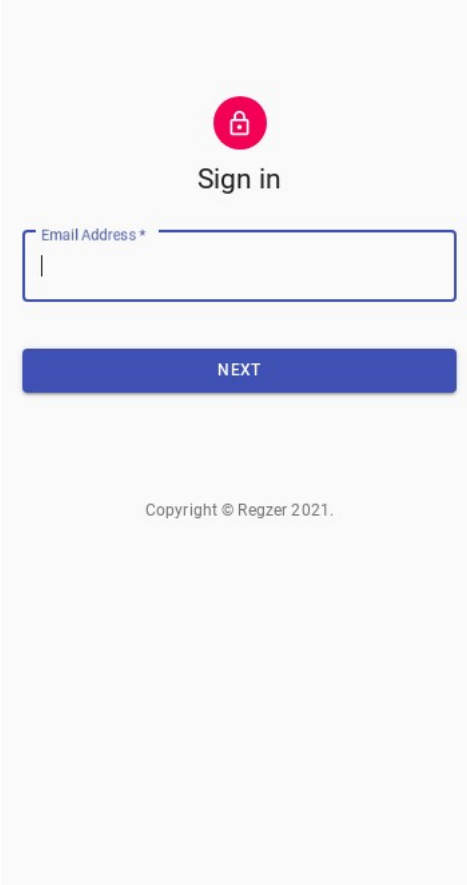
Email Address \*


|

NEXT

Copyright © Regzer 2021.

iPhone 7 Plus





Sign in

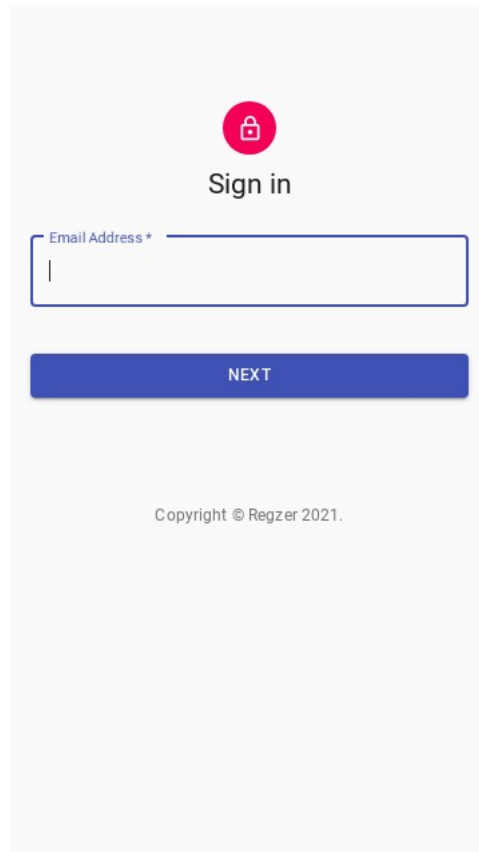
Email Address \*

|

NEXT

Copyright © Regzer 2021.

iPhone 8

A screenshot of a mobile application's sign-in screen. At the top center is a red circular icon with a white padlock. Below it, the text "Sign in" is displayed. Underneath is a text input field with a blue border and a blue outline, containing the placeholder text "Email Address \*". Below the input field is a solid blue button with the word "NEXT" in white capital letters. At the bottom center, the text "Copyright © Regzer 2021." is visible.

Samsung S7

## 5. URLs

### 5.1. Aplicação web

A aplicação web está hospedada em <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/>