Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

KAIO LUIZ BEGOSSO

PROTÓTIPO DE SISTEMA PARA CONTROLE DE PONTO

Assis/SP
2018

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

KAIO LUIZ BEGOSSO

PROTÓTIPO DE SISTEMA PARA CONTROLE DE PONTO

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis – IMESA e a Fundação Educacional do Município de Assis – FEMA, como requisito parcial à obtenção do Certificado de Conclusão.

Orientando(a): Kaio Luiz Begosso
Orientador(a): Prof. Dr. Alex S. R. de Souza Poletto

Assis/SP
2018

FICHA CATALOGRÁFICA

B417p BEGOSSO, Kaio Luiz
Protótipo de sistema para controle de ponto / Kaio Luiz Begosso. – Assis, 2018.

51p.

Trabalho de conclusão do curso (Análise e Desenvolvimento de Sistemas). – Fundação Educacional do Município de Assis-FEMA

Orientador: Dr. Alex Sandro Romeo de Souza Poletto

1.Raspberry PI 2.Internet 3.Sistema

CDD 006.4

RESUMO

O avanço das mais variadas tecnologias culmina em novas possibilidades e entre elas a automação. A automação em empresas está em exponencial crescimento devido ao fato da mesma proporcionar economia tanto em mão de obra quanto em tempo. Com base nessa premissa esse projeto visa atender tais demandas e solucionar os problemas identificados de forma simples e prática, com menor custo possível. Aliando os conceitos de Internet das Coisas (IoT), junto a um *Raspberry Pi* e assim tornando um sistema integrado aos demais sistemas existentes que as empresas possam ter.

Palavras-chave: Automação, Internet das Coisas, *Raspberry Pi*.

ABSTRACT

The advance of the most varied technologies culminates in new possibilities and among them the automation. The automation in companies is in exponential growth due to the fact that it provides economy in both labor and time. Based on this premise this project aims to meet these demands and solve the problems identified in a simple and practical, with the lowest possible cost. Combining the concepts of Internet of Things (IoT) and thus making an integrated system to the other existing systems that companies can have.

Keys word: Automation, Internet of Things, Raspberry Pi.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 – INTRODUÇÃO

Atualmente o controle de ponto, em determinadas entidades ou empresa, é feito de forma manual, por meio de folhas e/ou cadernetas. Com o avanço da tecnologia e da automação, a demanda por tempo e agilidade aumenta. O sistema proposto visa atender justamente esses requisitos com o auxilio da Internet das Coisas (IoT) e placas *Raspberry Pi*, que por sua vez possui um custo relativamente baixo e permite mobilidade e praticidade para utilização em diversas áreas.

1.1–OBJETIVOS

Com base nesse breve panorama, esse sistema tem por objetivo automatizar o controle de ponto de empresas facilitando o processo, promovendo agilidade e confiabilidade. Para tal pretende-se fazer uso de uma placa *Raspberry Pi* junto com o sistema implementado em *Python.*

1.2 - JUSTIFICATIVAS

Demonstrar que o desenvolvimento do sistema se justifica devido ao fato de proporcionar no mercado uma alternativa, com o uso de placas *Raspberry*, automatizando uma função de extrema importância para as empresas. Considerando o aumento da Internet das Coisas (IoT) a demanda por sistemas autônomos que facilitem a gestão das empresas e gerem economia é primordial. Pressupõe-se que, o quanto antes surgirem esses tipos de sistemas, melhor para empresas e para o mercado.

1.3–MOTIVAÇÃO

A motivação para o desenvolvimento da aplicação é a curiosidade em aprender sobre o tema de automação, bem como conhecimento de novas tecnologias e linguagens. Além de como o registro de ponto impacta no desenvolvimento e gestão das empresas.

1.4 – ESTRUTURA DO TRABALHO

Este trabalho está estruturado nas seguintes partes:

- **Capítulo 1 – Introdução**
- **Capítulo 2 – Tecnologia e Ferramentas de Desenvolvimento**
- **Capítulo 3 – Analise e Especificação do Sistema**
- **Capítulo 4 – Estrutura do Projeto**
- **Capítulo 5 – Implementação do Projeto**
- **Capítulo 6 – Considerações Finais**
- **Referências Bibliográficas**

2 – TECNOLOGIAS E FERRAMENTAS DE DESENVOLVIMENTO

O intuito deste capítulo é apresentar tecnologias e ferramentas que serão utilizadas para o desenvolvimento do sistema. Como será feito uso da placa *Raspberry Pi*, a mesma necessita a instalação de um Sistema Operacional. Neste caso, a aplicação será desenvolvida em *Raspbian/Debian*. Também será utilizado a IDE *PyCharm* para o desenvolvimento do software por ser uma plataforma fácil manuseio.

2.1 – PYCHARM

PyCharm é uma IDE (Ambiente de Desenvolvimento Integrado) voltada para o desenvolvimento de códigos/programas na linguagem *Python.* O *PyCharm* possui uma interface limpa e própria para quem está iniciando. Com diversas funções de *Python* internas de rápido acesso, além de diversas bibliotecas.

2.2 – RASPBERRY PI

O *Raspberry Pi* é um computador completo de pequeno porte, capaz de realizar tarefas como qualquer outro computador. Porém com foco na introdução e aprendizagem de programação. O que não o impede de ser utilizado para outros fins, devido ao baixo custo e fácil acesso.

A Figura 1 demonstra um *Raspberry Pi 3 Model B* que será utilizado no projeto:

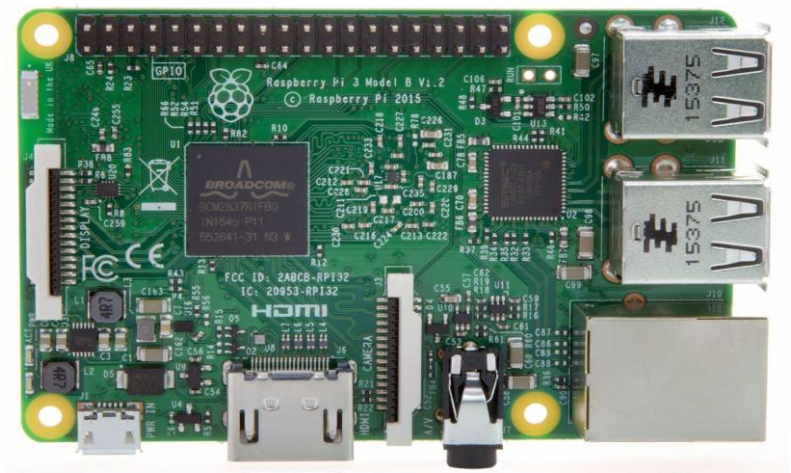


Figura 1 - Raspberry Pi

2.3 SPRING TOOL SUITE

O *Spring Tool Suite* é um ambiente de desenvolvimento baseado em *Eclipse* que é customizado para desenvolver aplicativos *Spring*. Ele fornece um ambiente pronto para uso para implementar, depurar, executar e implantar seus aplicativos Spring, incluindo integrações para *o Pivotal tc Server*, o *Pivotal Cloud Foundry*, o *Git*, o *Maven,* o *AspectJ* e os principais lançamentos do *Eclipse*.

Figura 2 - Spring Tool Suite

2.4 ANGULAR 6

Angular é uma plataforma e framework para construção da interface de aplicações usando HTML, CSS e, principalmente, *TypeScript*, criada pelos desenvolvedores da *Google*.

Figura 3 – Angular

Os blocos de construção básicos de um aplicativo Angular são NgModules, que fornecem um contexto de compilação para os componentes. NgModules coletam código relacionado em conjuntos funcionais; um aplicativo Angular é definido por um conjunto de NgModules. Um aplicativo sempre tem pelo menos um módulo raiz que permite a autoinicialização e normalmente tem muito mais módulos de recursos.

Os componentes definem exibições, que são conjuntos de elementos de tela que o Angular pode escolher e modificar de acordo com a lógica e os dados do programa.

Componentes usam serviços, que fornecem funcionalidade específica não diretamente relacionada as *views*. Os provedores de serviços podem ser injetados em componentes como dependências, tornando seu código modular, reutilizável e eficiente.

Ambos os componentes e serviços são simplesmente classes, com decoradores que marcam seu tipo e fornecem metadados que informam ao Angular como usá-los.

Os metadados de uma classe de componentes associam a um modelo que define uma exibição. Um modelo combina HTML comum com diretivas Angular e marcação de ligação que permitem que o Angular modifique o HTML antes de exibi-lo para exibição.

Os metadados de uma classe de serviço fornecem as informações que o Angular precisa para disponibilizá-los aos componentes por meio da injeção de dependência (DI).

Os componentes de um aplicativo geralmente definem várias exibições, organizadas hierarquicamente. Angular fornece o serviço Roteador para ajudá-lo a definir caminhos de navegação entre visualizações. O roteador oferece recursos sofisticados de navegação no navegador.[ANGULAR]

2.5 VISUAL STUDIO CODE

O *Visual Studio Code* é um editor de código fonte desenvolvido pela *Microsoft* para *Windows*, *Linux* e *macOS*. Ele inclui suporte para depuração, controle *Git* incorporado, realce de sintaxe, complementação inteligente de código, *snippets* e refatoração de código. Ele também é customizável, fazendo com que os usuários possam mudar o tema do editor, teclas de atalho e preferências. Ele é um software livre e de código aberto.

Figura 4 - Visual Studio Code

2.6 OPENCV

O *OpenCV* (*Open Source Computer Vision Library)* é uma biblioteca multiplataforma *open source* para o desenvolvimento de aplicativos na área de Visão Computacional.

Figura 5 - OpenCV

2.7 HAAR CASCADES

Um *Haar Cascade* é um tipo de classificador cascata que é usado para detectar o objeto para o qual foi treinado, a partir da fonte.

O *Haar Cascade* sobrepõe a imagem positiva sobre um conjunto de imagens negativas. O treinamento geralmente é feito em um servidor e em vários estágios. Melhores resultados são obtidos usando imagens de alta qualidade e aumentando a quantidade de estágios para os quais o classificador é treinado. Também é possível usar *Haar Cascade* pré-definidos que estão disponíveis no *Github*. [KAPUR, 2012]

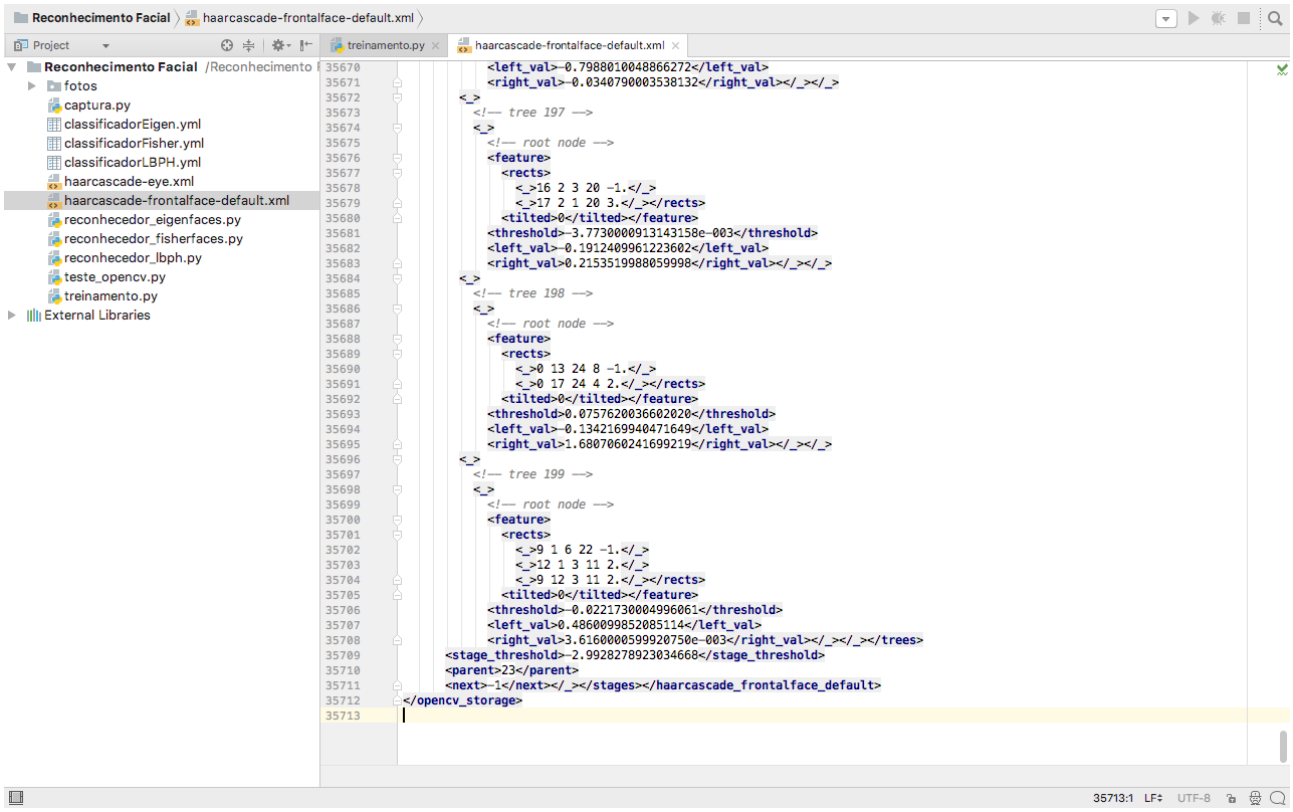


Figura 6 - Exemplo de Haar Cascade

3 – ANÁLISE E ESPECIFICAÇÃO DO SISTEMA

Neste capítulo serão apresentadas as ferramentas utilizadas para análise, especificações de requisitos funcionais e a modelagem de diagramas para melhor entendimento do sistema.

3.1 – MAPA MENTAL

O mapa mental pode ser usado para gerar, visualizar, organização, tomada de notas, resolução de problemas, tomada de decisão, revisão e esclarecimento para que se possa começar com as tarefas de avaliação.

Visando um melhor entendimento do *software* o Mapa Mental, descreve de modo simples uma visão sistêmica das funções do *software*, permitindo ao usuário ter uma noção do que o *software* pode proporcionar.

A Figura 7 é uma representação das funcionalidades da aplicação proposta:

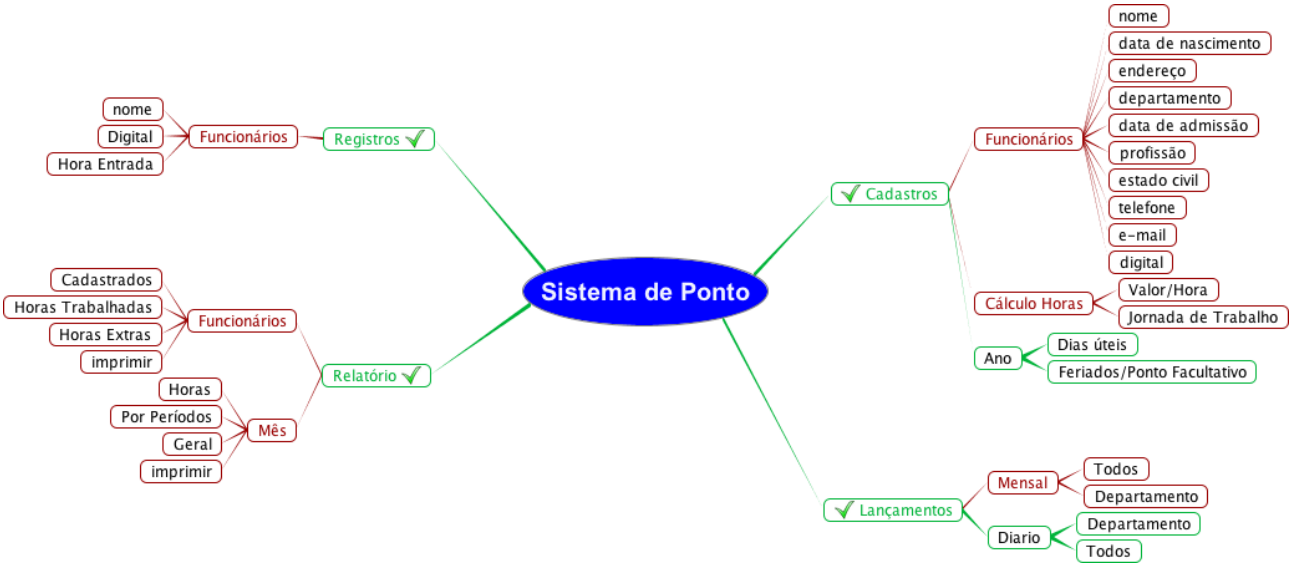


Figura 7 - Mapa Mental - Sistema de Ponto

3.2 – LISTA DE REQUISITOS

O software proposto neste trabalho possuirá diversas funcionalidades, tais como:

- Efetuar Login;
- Cadastrar Pessoa;
- Cadastrar Departamento;
- Consultar Pessoa;
- Registrar Entrada/Saída;
- Consultar Registro;
- Alterar Pessoa;
- Alterar Departamento;

3.3 – DIAGRAMA E ESPECIFICAÇÃO DE CASOS DE USO

O diagrama de caso de uso é o diagrama mais comum e informal da *Unified Modeling Language* (UML), utilizado geralmente nas fases de levantamento e análise de requisitos do sistema, além de que pode servir de base para outros diagramas.

A fim de melhor descrever as funcionalidades da aplicação, foram elaborados alguns diagramas, e assim descrever a interação do usuário com o sistema. Colocando narrativas para especificar os diagramas, descrevendo em forma textual a interação do usuário com o sistema.

A Figura 8 apresenta o Diagrama de Caso de Uso comum para as entidades “Administrador”, “Funcionário” e “Sistema”:

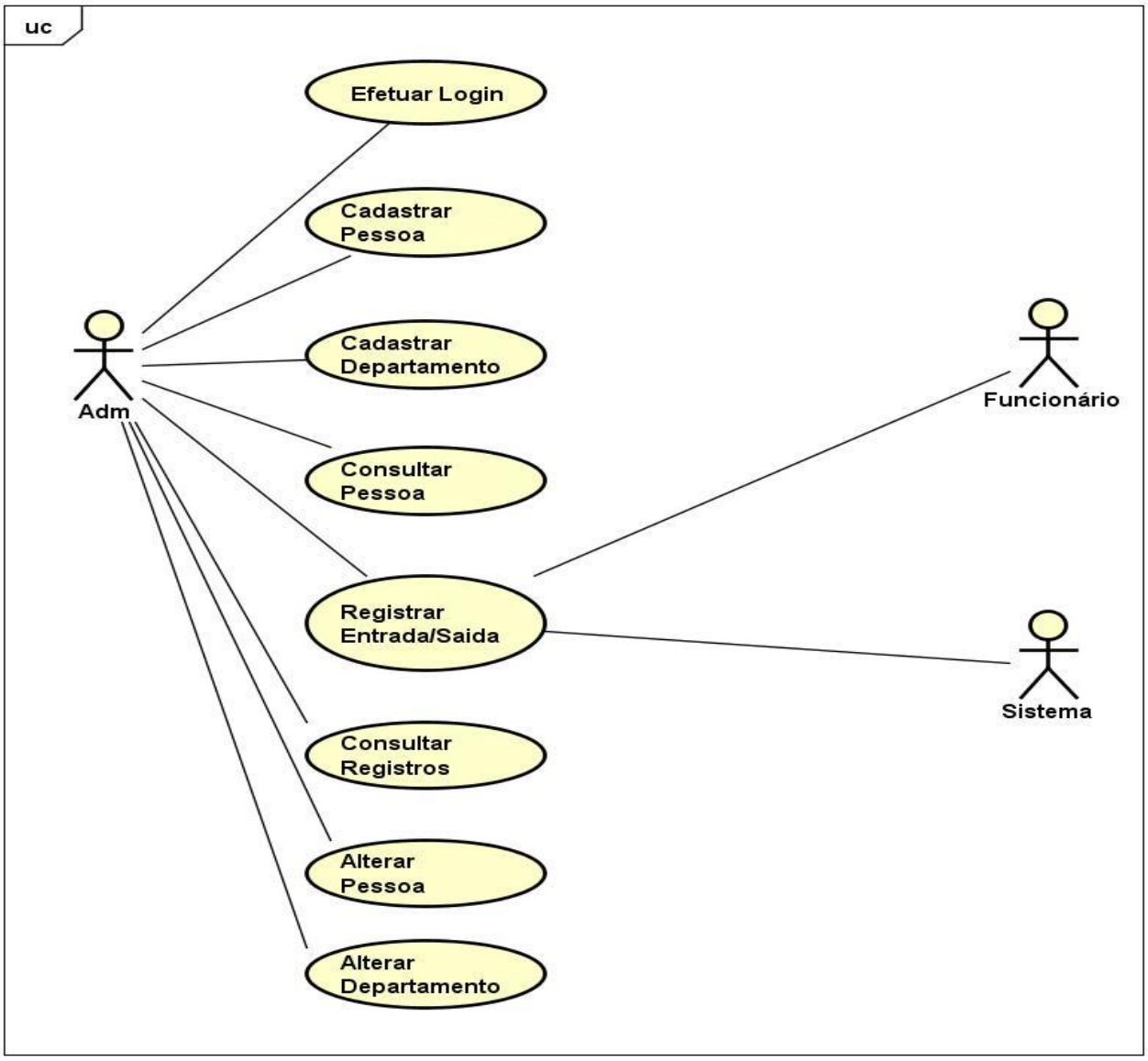


Figura 8 - Diagrama de Caso de Uso comum "Adm", "Usuário" e "Sistema"

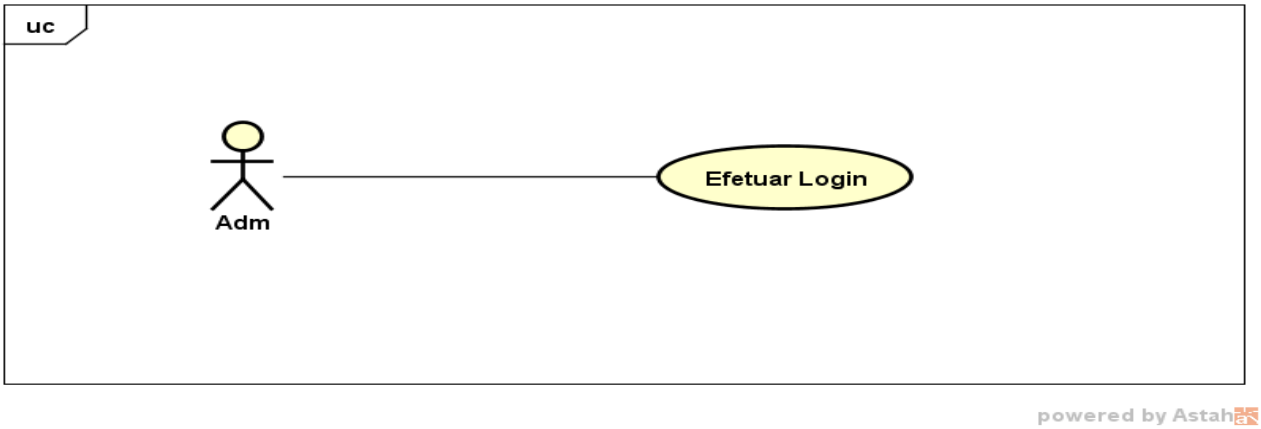


Figura 9 - Diagrama de Caso de Uso Efetuar Login

Nome do Caso de uso	Efetuar Login
Atores	Administrador
Pré-Condições	Não há
Cenário Principal	1. O ator inicia o caso de uso selecionando efetuar login. 2. O sistema oferece a interface para efetuar login. (A1) 3. O ator informa os dados para acesso. (E1) 4. O sistema verifica os dados. 5. O sistema permite acesso ao usuário.
Cenário Alternativo	A1 - Cancela a operação. • O ator cancela a operação de cadastro de funcionário. • O sistema retorna ao passo 1 do cenário principal.
Cenário de Exceção	E1 - Dados inválidos • O sistema verifica se os dados informados pelo usuário são inválidos. • O sistema retorna ao passo 2 do cenário principal.
Casos de Teste	Verificar campo de código do funcionário.

Tabela 1 - Efetuar Login

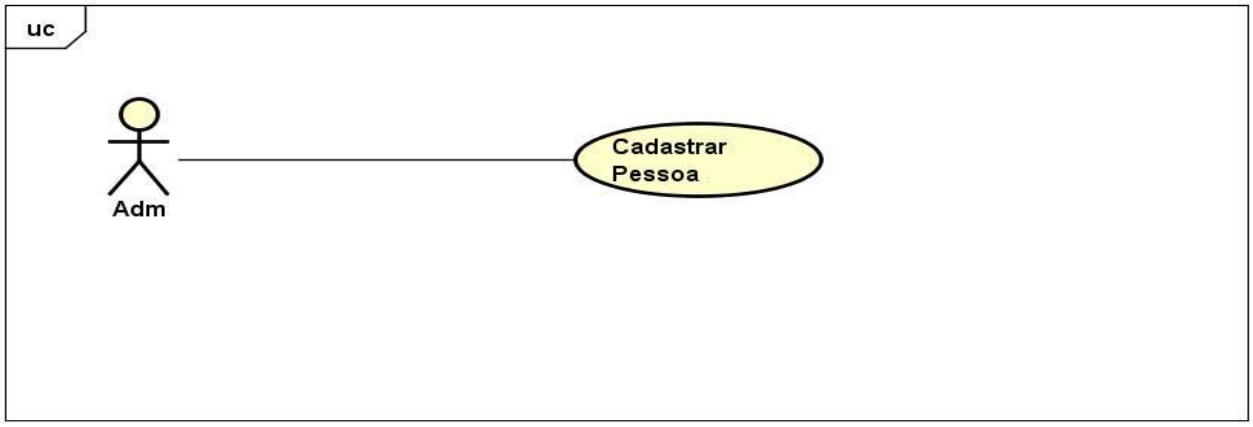


Figura 10 - Diagrama de Caso de Uso Cadastrar Pessoa

Nome do Caso de uso	Cadastrar Pessoa
Atores	Administrador
Pré-Condições	Efetuar Login
Cenário Principal	1. O ator inicia o caso de uso selecionando cadastrar pessoa. 2. O sistema oferece a interface para cadastrar a pessoa. (A1) 3. O ator informa os dados da pessoa para cadastro. (E1) 4. O sistema solicita confirmação para cadastro da pessoa. 5. O ator confirma o cadastro da pessoa. 6. O sistema exibi a mensagem de pessoa cadastrada com sucesso.
Cenário Alternativo	A1 - Cancela a operação. • O ator cancela a operação de cadastro de pessoa. • O sistema retorna ao passo 1 do cenário principal.
Cenário de Exceção	E1 - Dados inválidos • O sistema verifica se os dados informados pelo usuário são inválidos. • O sistema retorna ao passo 2 do cenário principal.
Casos de Teste	Verificar campo de código da pessoa.

Tabela 2 - Cadastrar Pessoa

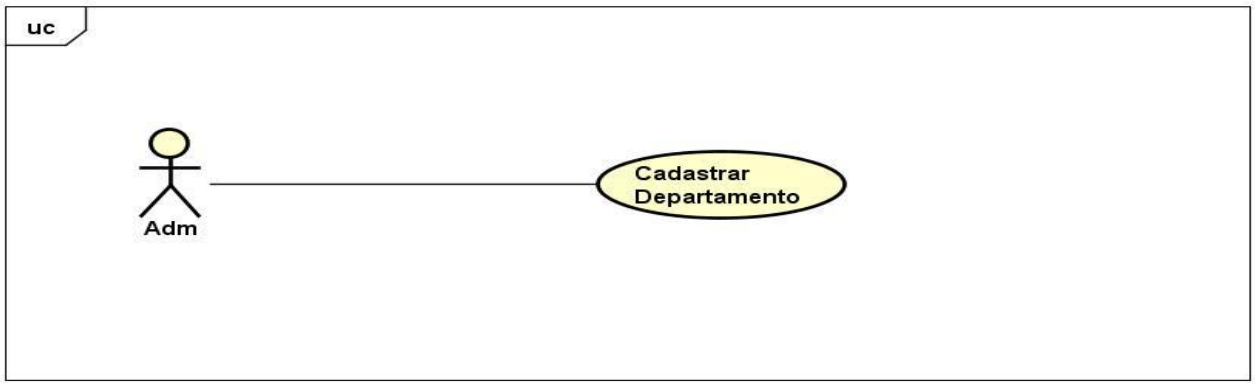


Figura 11 - Diagrama Caso de Uso Cadastrar Departamento

Nome do Caso de uso	Cadastrar Departamento
Atores	Administrador
Pré-Condições	Efetuar Login
Cenário Principal	1. O ator inicia o caso de uso selecionando cadastrar departamento. 2. O sistema oferece a interface para cadastrar o departamento. (A1) 3. O ator informa os dados do departamento para cadastro. (E1) 4. O sistema solicita confirmação para cadastro de departamento. 5. O ator confirma o cadastro do departamento. 6. O sistema exibi a mensagem de departamento cadastrado com sucesso.
Cenário Alternativo	A1 - Cancela a operação. • O ator cancela a operação de cadastro de departamento. • O sistema retorna ao passo 1 do cenário principal.
Cenário de Exceção	E1 - Dados inválidos • O sistema verifica se os dados informados pelo usuário são inválidos. • O sistema retorna ao passo 2 do cenário principal.
Casos de Teste	Verificar campo nome.

Tabela 3 - Cadastrar Departamento

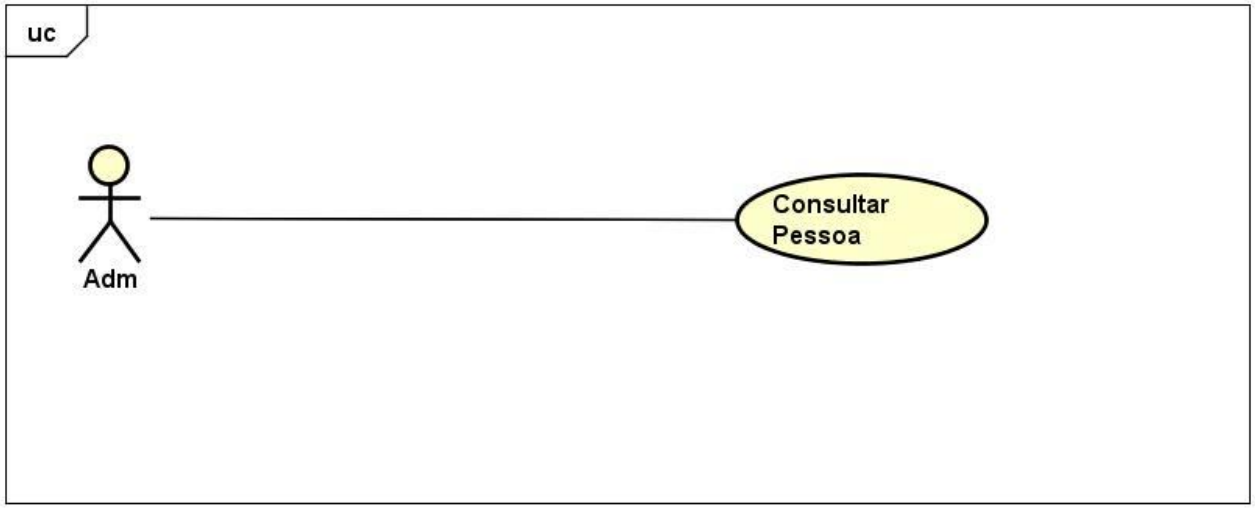


Figura 12 - Diagrama de Caso de Uso Consultar Pessoa

Nome do Caso de uso	Cadastrar Consultar Pessoa
Atores	Administrador
Pré-Condições	Efetuar Login
Cenário Principal	1. O ator inicia o caso de uso selecionando consultar pessoa. 2. O sistema oferece a interface com todas as pessoas cadastradas. (A1)
Cenário Alternativo	A1 - Cancela a operação. • O ator cancela a operação de consultar pessoa. • O sistema retorna ao passo 1 do cenário principal.
Cenário de Exceção	Não há.
Casos de Teste	Caso os dados estejam corretos executa a operação

Tabela 4 – Consultar Pessoa

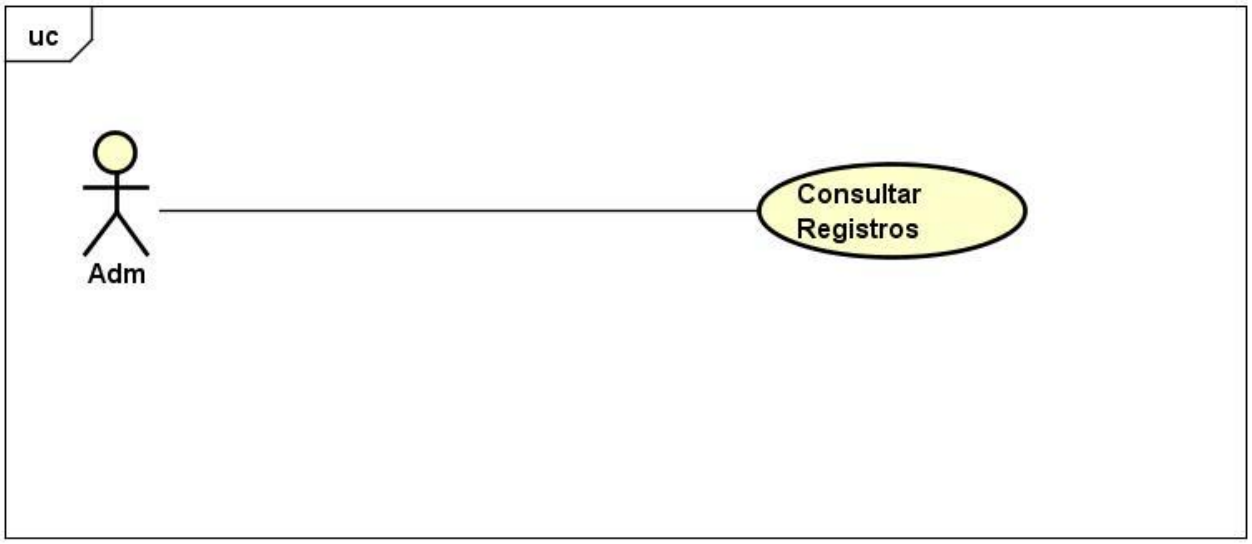


Figura 13 - Diagrama de Caso de Uso Consultar Registros

Nome do Caso de uso	Cadastrar Consultar Registros
Atores	Administrador
Pré-Condições	Efetuar Login
Cenário Principal	1. O ator inicia o caso de uso selecionando consultar registros. 2. O sistema oferece a interface com todas os registros cadastrados. (A1)
Cenário Alternativo	A1 - Cancela a operação. • O ator cancela a operação de consultar registros. • O sistema retorna ao passo 1 do cenário principal.
Cenário de Exceção	Não há.
Casos de Teste	Caso os dados estejam corretos executa a operação

Tabela 5 - Consultar Registros

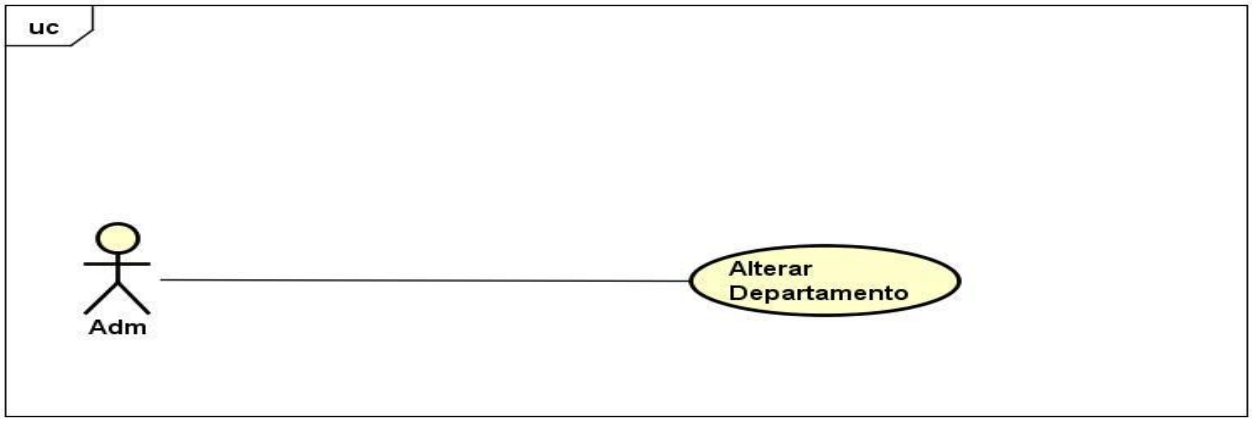


Figura 14 - Diagrama Caso de Uso Alterar Departamento

Nome do Caso de uso	Alterar Departamento
Atores	Administrador
Pré-Condições	Efetuar Login
Cenário Principal	1. O ator inicia o caso de uso selecionando alterar departamento. 2. O sistema oferece a interface para alterar o departamento. (A1) 3. O ator informa os dados do departamento para alteração. (E1) 4. O sistema solicita confirmação para alteração de departamento. 5. O ator confirma a alteração do departamento. 6. O sistema exibi a mensagem de departamento alterado com sucesso.
Cenário Alternativo	A1 - Cancela a operação. • O ator cancela a operação de alterar de departamento. • O sistema retorna ao passo 1 do cenário principal.
Cenário de Exceção	E1 - Dados inválidos • O sistema verifica se os dados informados pelo usuário são inválidos. • O sistema retorna ao passo 2 do cenário principal.
Casos de Teste	Verificar campo nome.

Tabela 6 – Alterar Departamento

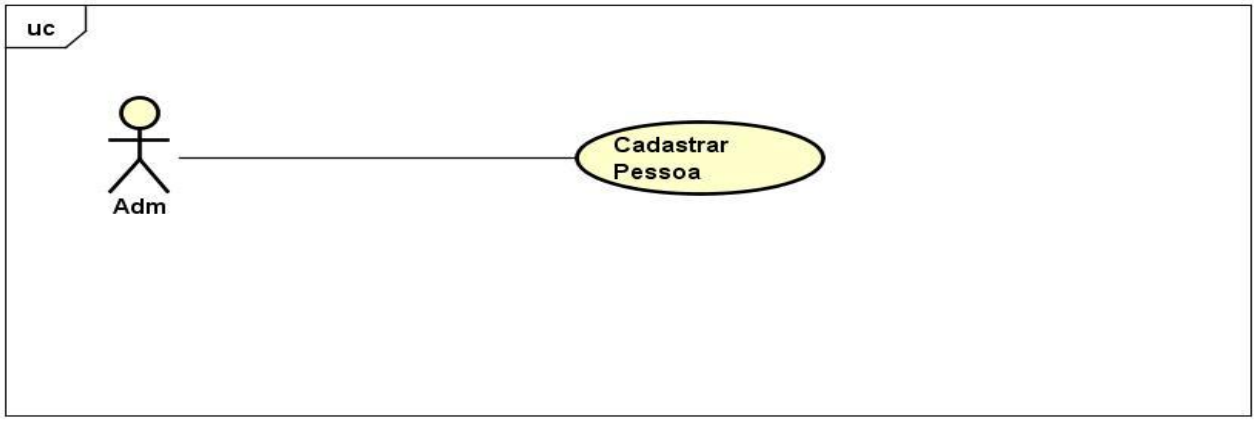


Figura 15 - Diagrama de Caso de Uso Alterar Pessoa

Nome do Caso de uso	Cadastrar Pessoa
Atores	Administrador
Pré-Condições	Efetuar Login
Cenário Principal	1. O ator inicia o caso de uso selecionando alterar pessoa. 2. O sistema oferece a interface para alterar a pessoa. (A1) 3. O ator informa os dados da pessoa para alteração. (E1) 4. O sistema solicita confirmação para alterar pessoa. 5. O ator confirma a alteração da pessoa. 6. O sistema exibi a mensagem de pessoa alterada com sucesso.
Cenário Alternativo	A1 - Cancela a operação. • O ator cancela a operação de cadastro de pessoa. • O sistema retorna ao passo 1 do cenário principal.
Cenário de Exceção	E1 - Dados Inválidos • O sistema verifica se os dados informados pelo usuário são inválidos. • O sistema retorna ao passo 2 do cenário principal.
Casos de Teste	Verificar campo de código da pessoa.

Tabela 7 - Alterar Pessoa

Figura 16 - Diagrama de Caso de Uso Registrar Entrada Funcionário

Nome do Caso de uso	Registrar Entrada/Saída
Atores	Funcionário
Pré-Condições	Não há
Cenário Principal	1. O ator inicia o caso colocando seu rosto em posição frontal a câmera. 2. O sistema verifica o rosto do funcionário. (A1) 3. O sistema informa os dados do funcionário. (E1) 4. O ator confirma o registro. 5. O sistema exibi a mensagem de registro efetuado com sucesso.
Cenário Alternativo	A1 - Cancela a operação. • O ator cancela a operação de cadastro de funcionário. • O sistema retorna ao passo 1 do cenário principal.
Cenário de Exceção	E1 - Dados inválidos • O sistema verifica se os dados informados pelo usuário são inválidos. • O sistema retorna ao passo 2 do cenário principal.
Casos de Teste	Verificar campo de precisão do reconhecimento.

Tabela 8 - Registrar Entrada/Saída Funcionário

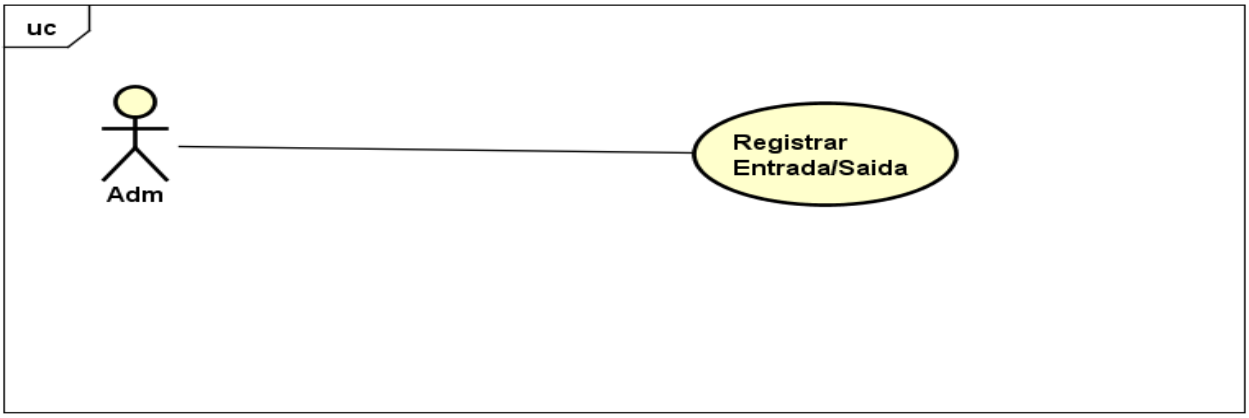


Figura 17 - Diagrama de Caso de Uso Registrar Entrada/Saída Adm

Nome do Caso de uso	Registrar Entrada/Saída Adm
Atores	Adm
Pré-Condições	Efetuar Login
Cenário Principal	1. O ator inicia o caso de uso selecionando registrar Entrada/Saída. 2. O sistema oferece a interface para registrar entrada/saída. (A1) 3. O usuário informa os dados para registro. (E1) 4. O sistema solicita confirmação para registro de entrada/saída. 5. O ator confirma o registro. 6. O sistema exibi a mensagem de registro efetuado com sucesso.
Cenário Alternativo	A1 - Cancela a operação. • O ator cancela a operação de cadastro de funcionário. • O sistema retorna ao passo 1 do cenário principal.
Cenário de Exceção	E1 - Dados inválidos • O sistema verifica se os dados informados pelo usuário são inválidos. • O sistema retorna ao passo 2 do cenário principal.
Casos de Teste	Verificar campo precisão do reconhecimeto.

Tabela 9 - Registrar Entrada/Saída Adm

3.4 – DIAGRAMA DE CLASSE

Os diagramas de classes são os diagramas encontrados com maior frequência na modelagem de sistema orientado a objetos. Um diagrama de classes mostra um conjunto de classes, interfaces e colaborações e seus relacionamentos. Os diagramas de classe são importantes não só para a visualização, a especificação e a documentação de modelos estruturais, mas também para construção de sistemas executáveis por intermédio de engenharia de produção e reserva (BOOCH, RUMBAUGH, JACOBSON, 2005).

A Figura 18 Diagrama de Classe demonstrar algumas atividades que o sistema pode prover ao usuário:

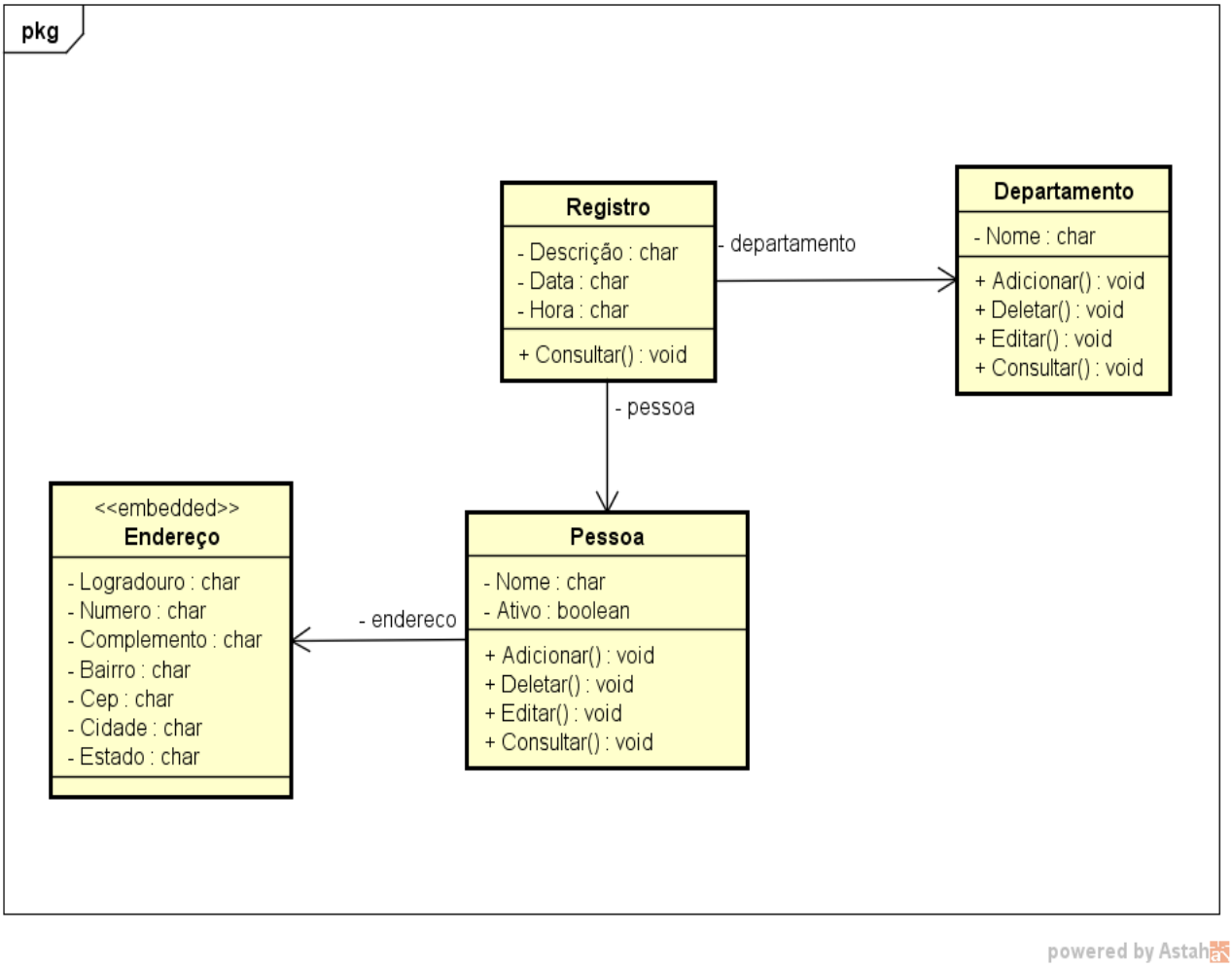


Figura 18 - Diagrama de Classes

3.5 – DIAGRAMA DE ENTIDADE E RELACIONAMENTO

O Diagrama de Entidade e Relacionamento (DER) foi originalmente proposto por Peter Chen para projeto de sistema de banco de dados relacionais e foi estendido por outros. Um conjunto de componentes primordiais é identificado para o DER: objetos de dados, atributos, relacionamentos e indicadores de vários tipos. O principal objetivo do DER é representar objetivos de dados e seus relacionamentos (PRESSMAN, 2011).

A Figura 19 demonstra o Diagrama de Entidade e Relacionamento do sistema:

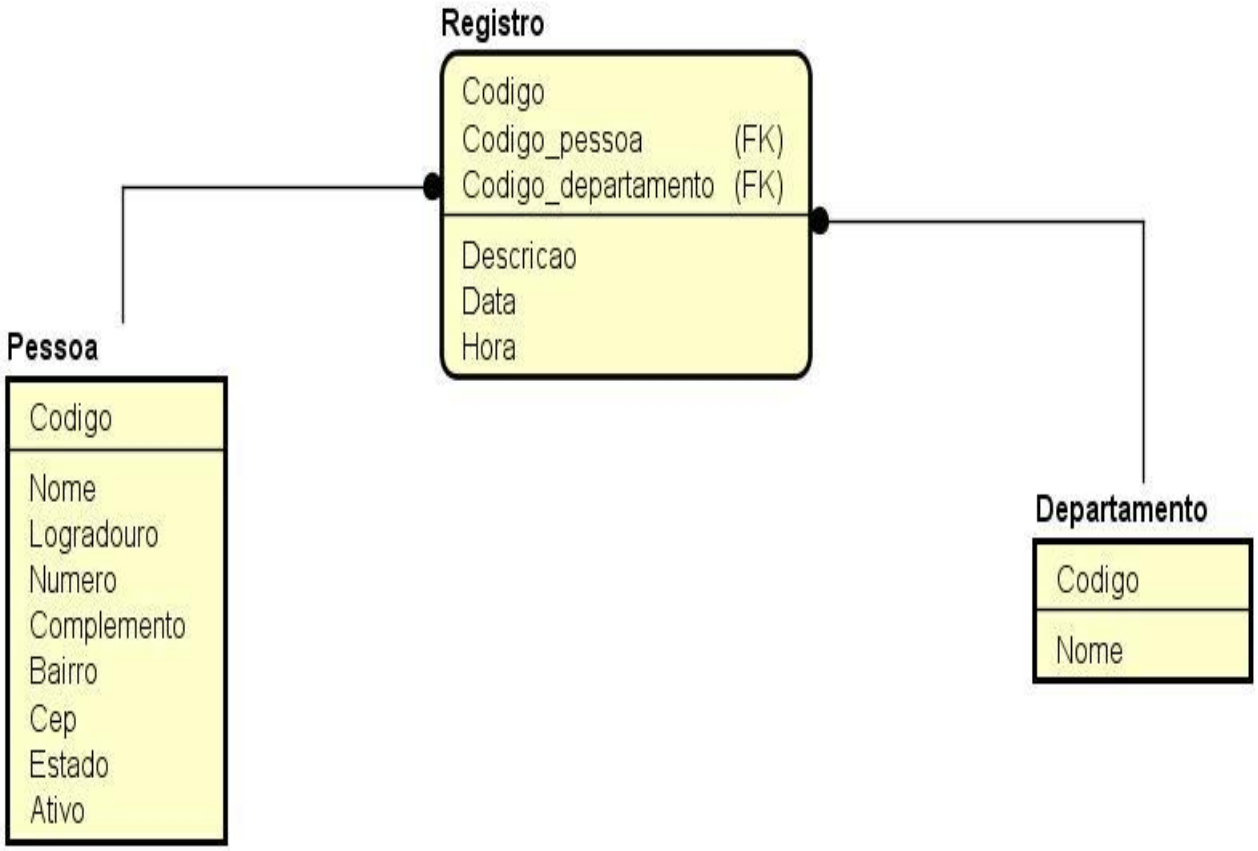


Figura 19 - Diagrama de Entidade Relacionamento

4 – ESTRUTURA DO PROJETO

Neste capitulo será apresentada a estrutura do projeto adotada para o desenvolvimento do trabalho de conclusão de curso, a qual consiste em fases e etapas.

4.1 – ESTRUTURA ANALÍTICA DO PROJETO

A Estrutura Analítica do Projeto *(*EAP*) é* o processo de subdivisão das entregas e do trabalho do projeto em componentes menores e de melhor gerenciamento. A EAP é uma decomposição hierárquica orientada às entregas do trabalho a ser executado, para atingir os objetivos do projeto e criar as entregas requisitadas, sendo que cada nível descendente da EAP representa uma definição gradualmente mais detalhada da definição do trabalho do projeto.

As etapas do desenvolvimento da aplicação são ilustradas na Figura 12 que apresenta a EAP que será utilizado para organizar e orientar o trabalho a ser desenvolvido:

Figura 20 - Estrutura Analítica do Projeto

4.2 – CRONOGRAMA FÍSICO FINANCEIRO

Branch Name	Duration	Start	End	Priority	Completion	Resources	Work	Cost
EAP - Sistema Ponto	**102 days**	**13/11/2017**	**03/04/2018**	**500**	**0%**		**1008 hrs**	**R$24.796,80**
⊟ **Planejamento**	**4 days**	**13/11/2017**	**16/11/2017**	**500**	**0%**		**48 hrs**	**R$1.800,00**
Análise da Oportunidade	1 day	13/11/2017	13/11/2017	500	0%	Gerente;Analista	16 hrs	R$540,00
Análise dos Requisitos d...	1 day	14/11/2017	14/11/2017	500	0%	Gerente;Analista	16 hrs	R$540,00
Proposta Técnico-Comer...	2 days	15/11/2017	16/11/2017	500	0%	Gerente	16 hrs	R$720,00
⊟ **Análise dos Requisitos**	**8 days**	**17/11/2017**	**28/11/2017**	**500**	**0%**		**128 hrs**	**R$4.320,00**
Requisitos Funcionais e ...	2 days	17/11/2017	20/11/2017	500	0%	Gerente;Analista	32 hrs	R$1.080,00
Requisitos do Design	2 days	21/11/2017	22/11/2017	500	0%	Gerente;Analista	32 hrs	R$1.080,00
Requisitos de Performance	2 days	23/11/2017	24/11/2017	500	0%	Gerente;Analista	32 hrs	R$1.080,00
Requisitos de Infraestrut...	2 days	27/11/2017	28/11/2017	500	0%	Gerente;Analista	32 hrs	R$1.080,00
⊟ **Desenvolvimento**	**70 days**	**29/11/2017**	**06/03/2018**	**500**	**0%**		**512 hrs**	**R$9.932,80**
Definição da Metodologia	1 day	29/11/2017	29/11/2017	500	0%	Analista;Programador	16 hrs	R$334,40
Definição das Ferramentas	1 day	30/11/2017	30/11/2017	500	0%	Analista;Programador	16 hrs	R$334,40
⊟ Codificação/Programação	43 days	01/12/2017	30/01/2018	500	0%		280 hrs	R$5.404,00
Implementação do Mód...	5 days	01/12/2017	07/12/2017	500	0%	Programador	40 hrs	R$772,00
Implementação da Co...	5 days	08/12/2017	14/12/2017	500	0%	Programador	40 hrs	R$772,00
Implementação da Inter...	5 days	15/12/2017	21/12/2017	500	0%	Programador	40 hrs	R$772,00
Implementação dos Rel...	5 days	03/01/2018	09/01/2018	500	0%	Programador	40 hrs	R$772,00
Implementação da Mov...	5 days	10/01/2018	16/01/2018	500	0%	Programador	40 hrs	R$772,00
Implementação dos Re...	5 days	17/01/2018	23/01/2018	500	0%	Programador	40 hrs	R$772,00
Implementação do Rec...	5 days	24/01/2018	30/01/2018	500	0%	Programador	40 hrs	R$772,00
Documentação	20 days	31/01/2018	27/02/2018	500	0%	Programador	160 hrs	R$3.088,00
Testes Iniciais	5 days	28/02/2018	06/03/2018	500	0%	Programador	40 hrs	R$772,00
⊟ **Organizar**	**10 days**	**07/03/2018**	**20/03/2018**	**500**	**0%**		**160 hrs**	**R$3.344,00**
Piloto em Ambiente de Pr...	5 days	07/03/2018	13/03/2018	500	0%	Analista;Programador	80 hrs	R$1.672,00
Treinamento do Usuário	5 days	14/03/2018	20/03/2018	500	0%	Analista;Programador	80 hrs	R$1.672,00
⊟ **Operação Assistida**	**10 days**	**21/03/2018**	**03/04/2018**	**500**	**0%**		**160 hrs**	**R$5.400,00**
Suporte ao Cliente	10 days	21/03/2018	03/04/2018	500	0%	Analista	80 hrs	R$1.800,00
Gestão das Mudanças	10 days	21/03/2018	03/04/2018	500	0%	Gerente	80 hrs	R$3.600,00

Figura 21 - Cronograma Físico Financeiro

5 – IMPLEMENTAÇÃO

Nesse capítulo é descrito como foi feita a implementação do projeto. A implementação foi subdividida basicamente em três partes, sendo elas: Implementação API REST, Front-End e Aplicação Desktop/*Raspberry.*

5.1 API REST

Para a implementação da API REST foi utilizado o STS (Spring Tool Suit), um framework que facilita o desenvolvimento da mesma.

Primeiramente foram definidas as classes e seus respectivos mapeamentos junto com a implementação dos serviços , conforme as imagens a seguir.

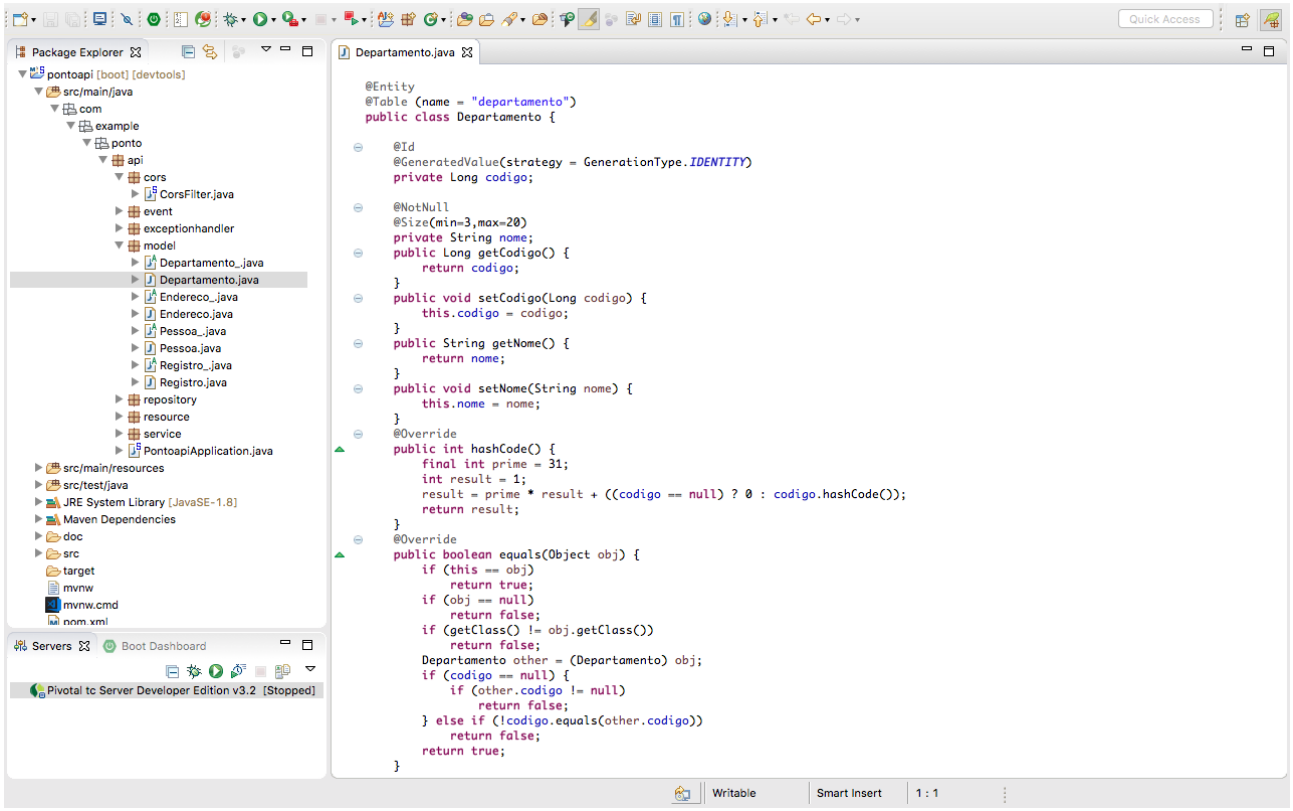


Figura 22 - Implementação da Classe Departamento

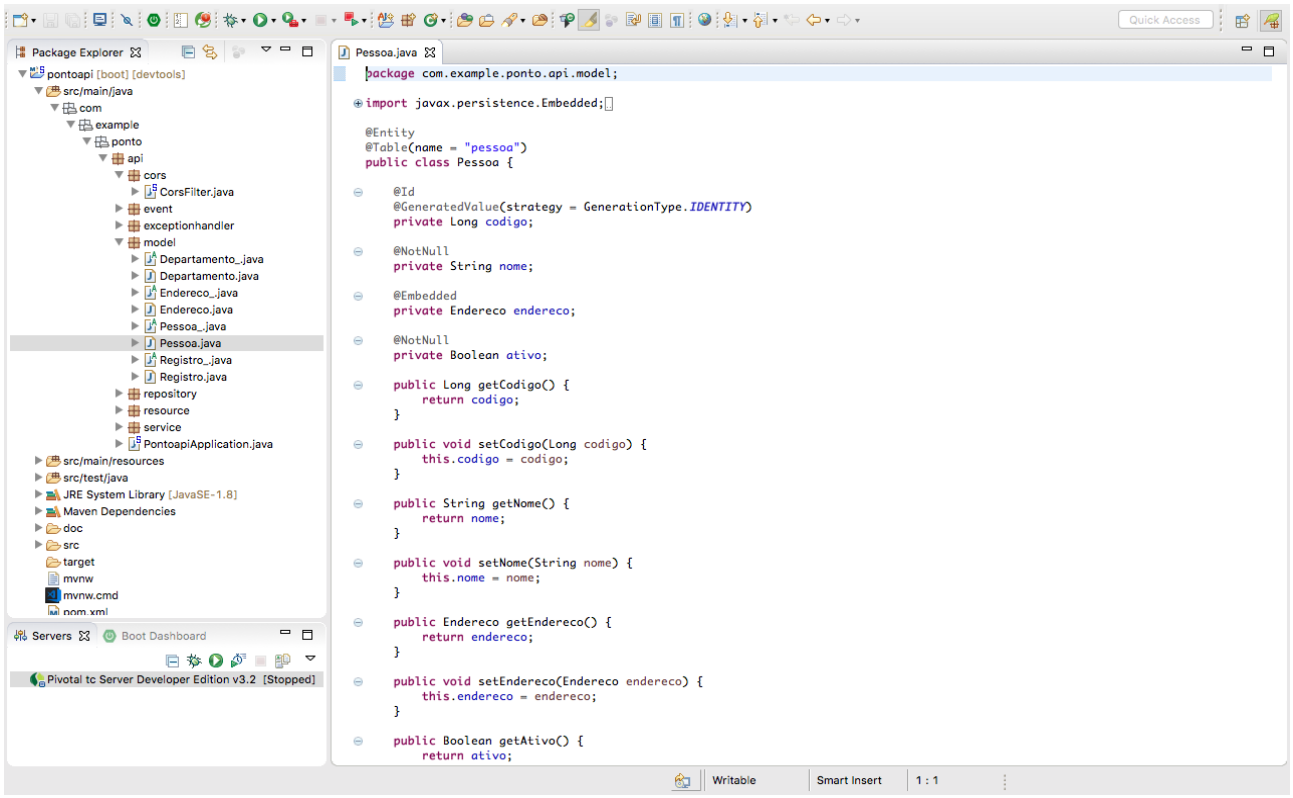


Figura 23 - Implementação da Classe Pessoa

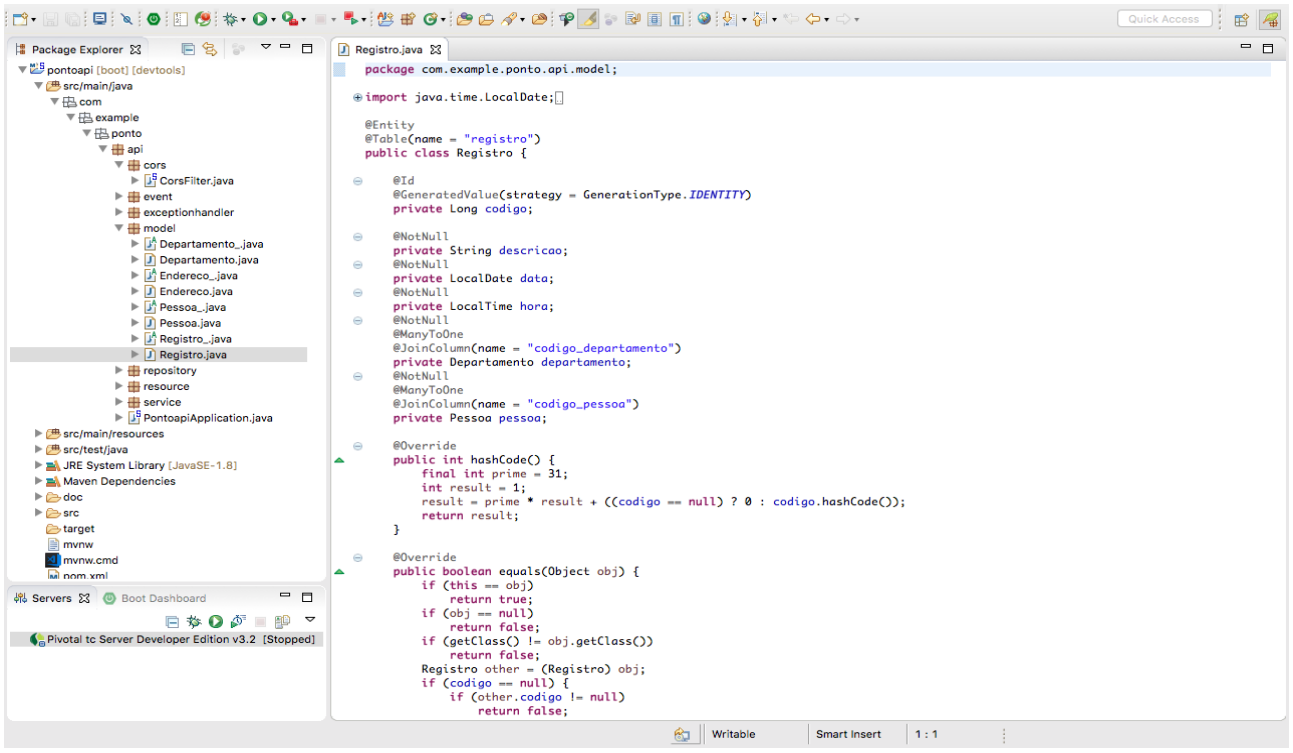


Figura 24 - Implementação da Classe Registro

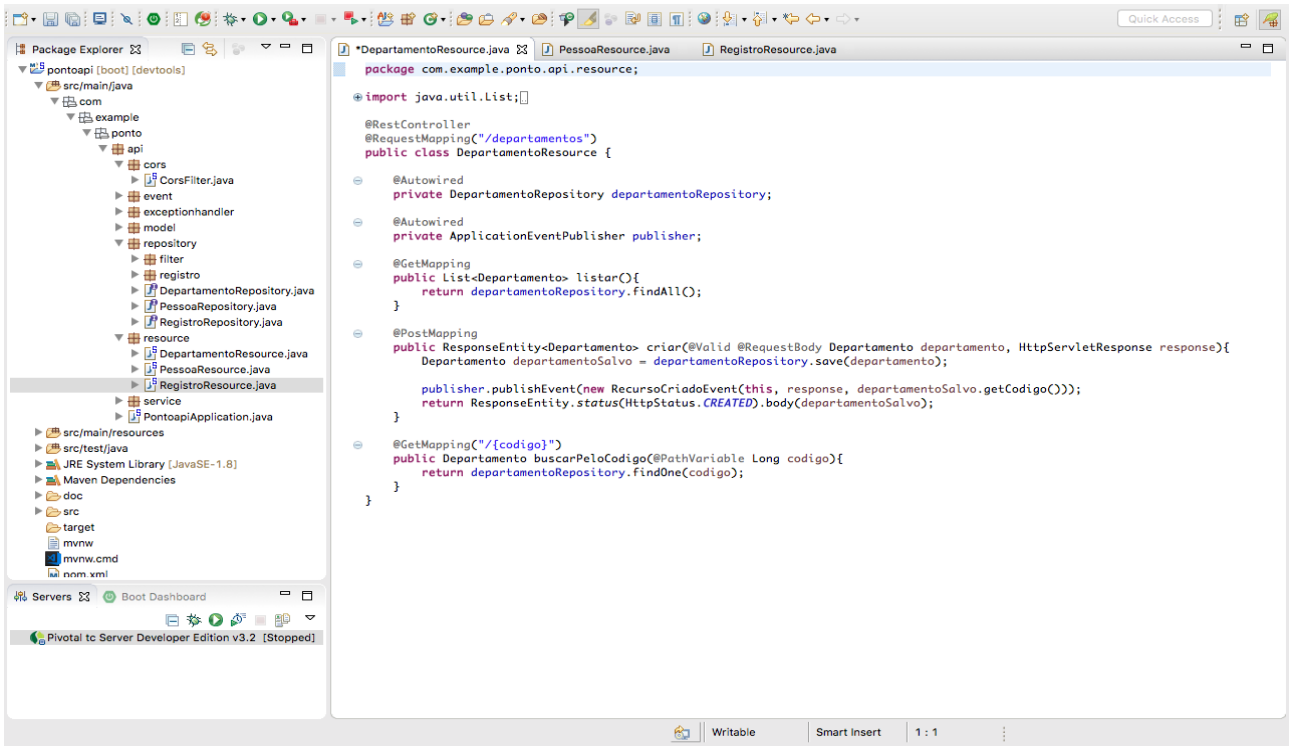


Figura 25 - Mapeamento de Departamento

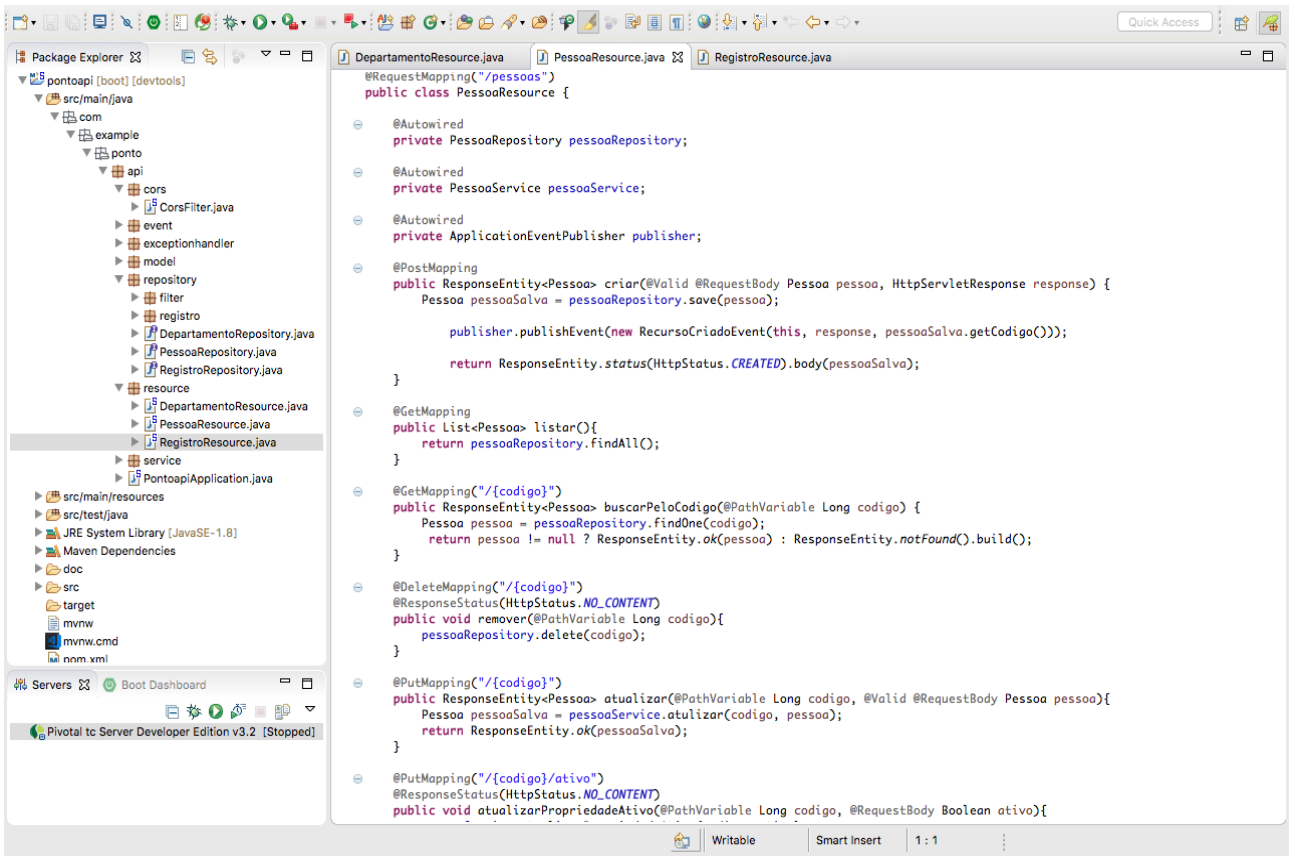


Figura 26 - Mapeamento de Pessoa

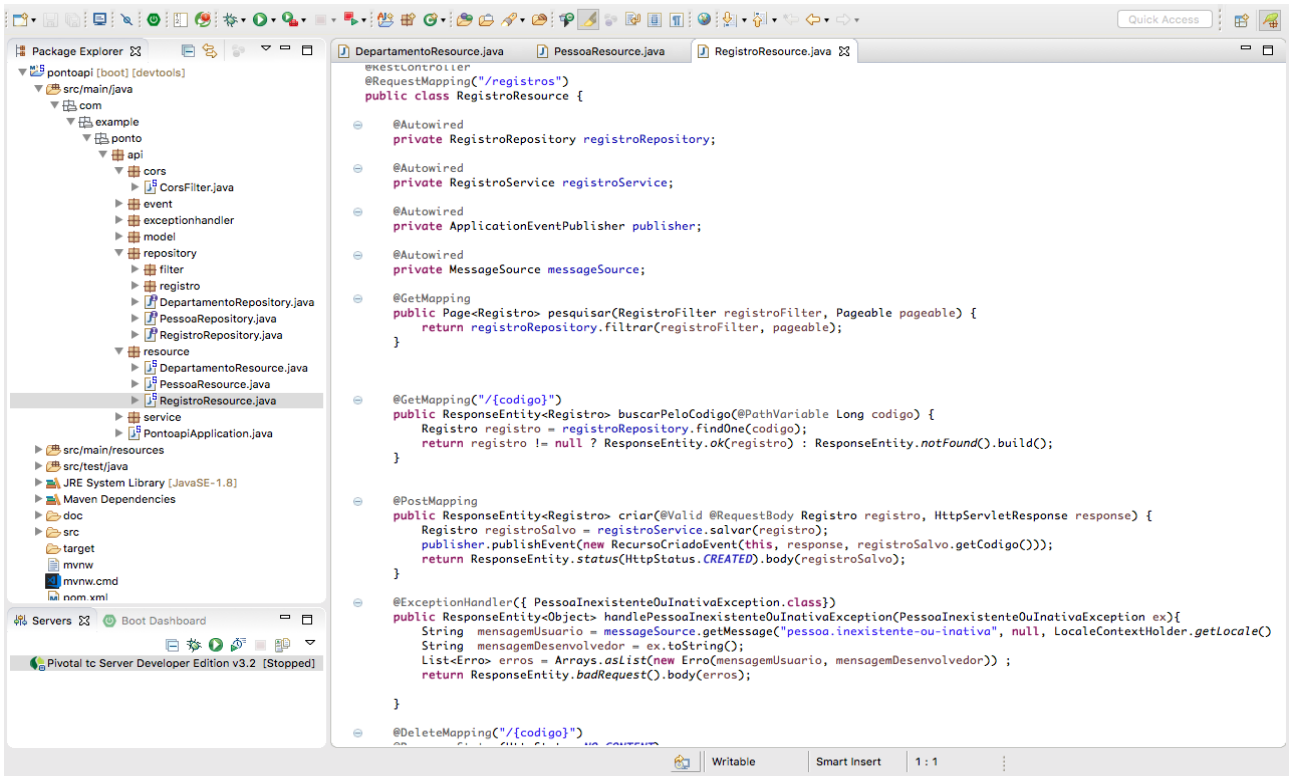


Figura 27 - Mapeamento de Registro

Depois deu-se inicio aos testes da API, para tal, foi utilizado o POSTMAN.

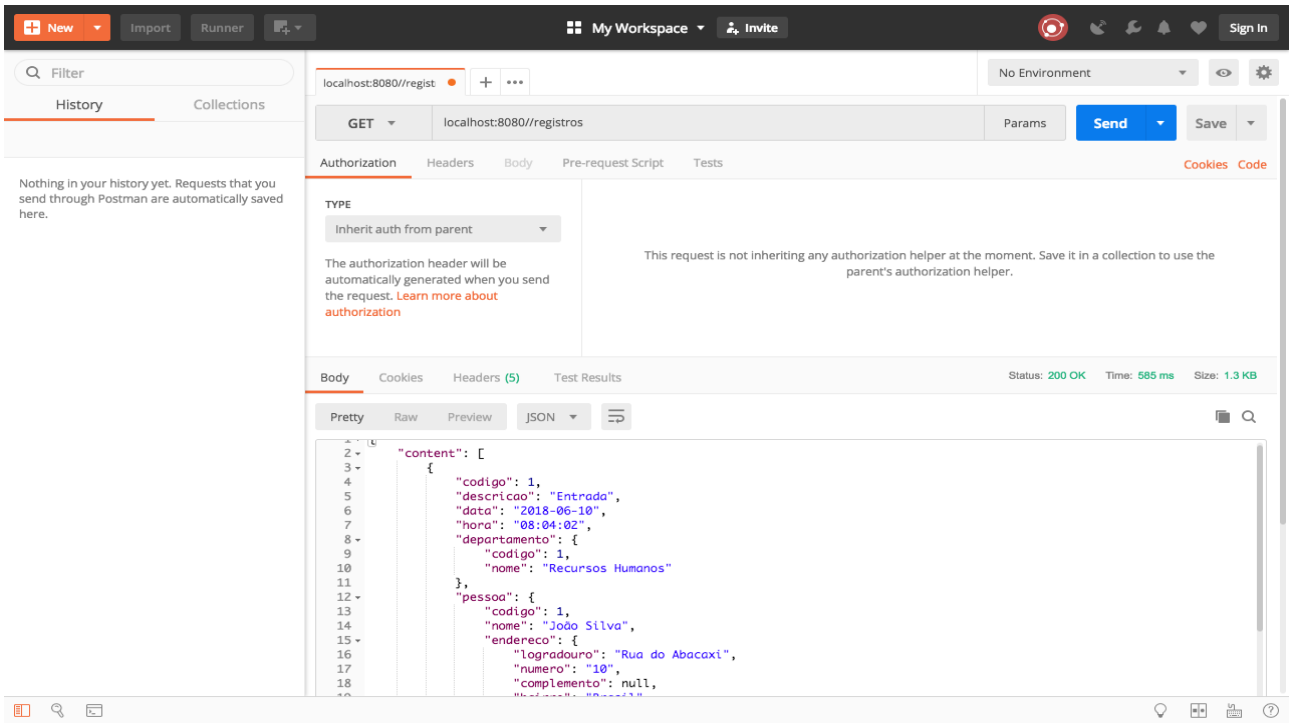


Figura 28 - Teste da API 1

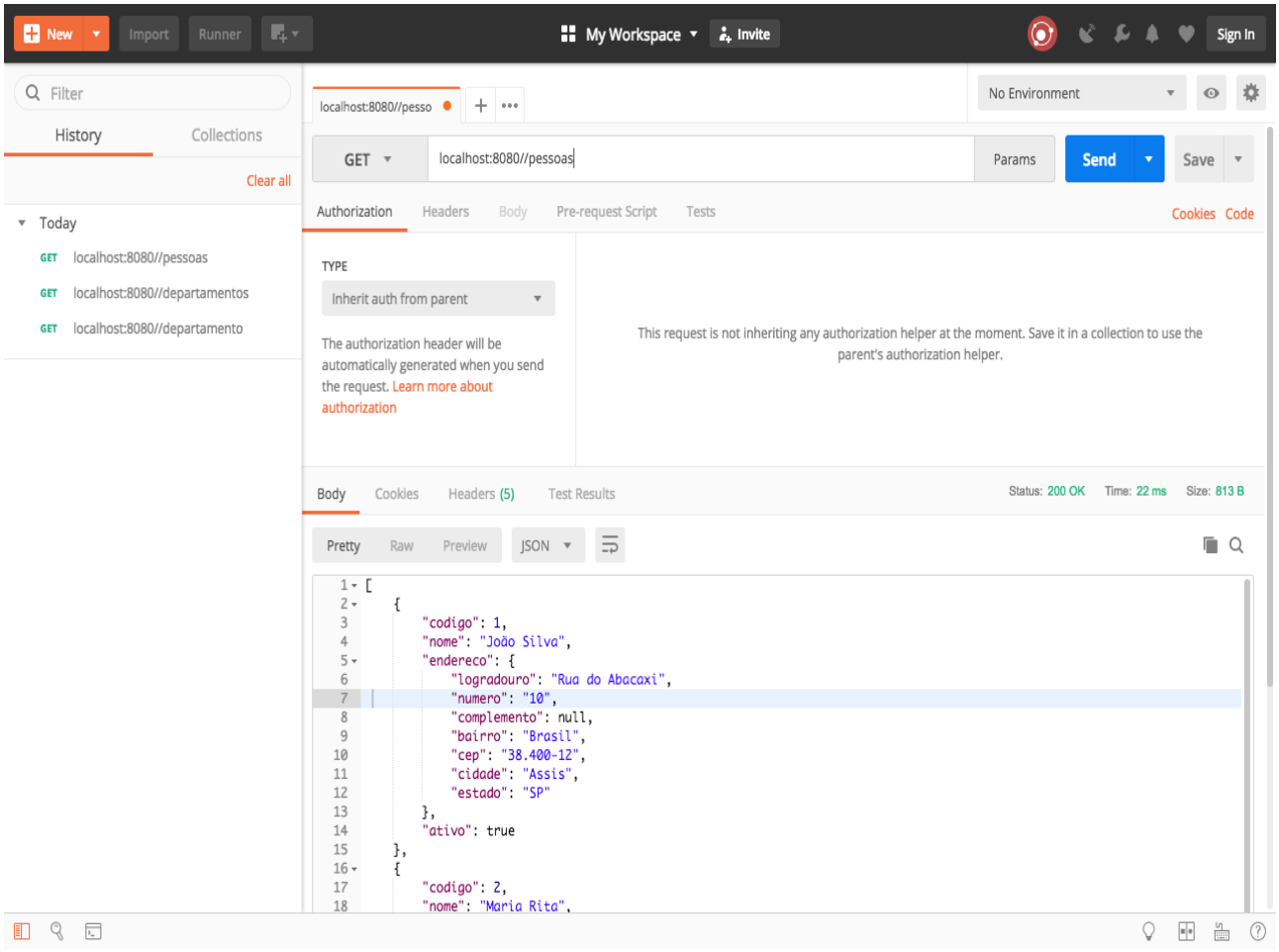


Figura 29 - Teste da API 2

5.2 Front-End

Para a implementação da interface do usuário foi utilizado o Angular 6 e Visual Studio Code. De início foram criados os componentes e os modelos das classes definidas na API e no Diagrama de Classes, conforme as imagens demonstram. Após foram feitos os formulários de inclusão, pesquisa e consultas das respectivas Classes.

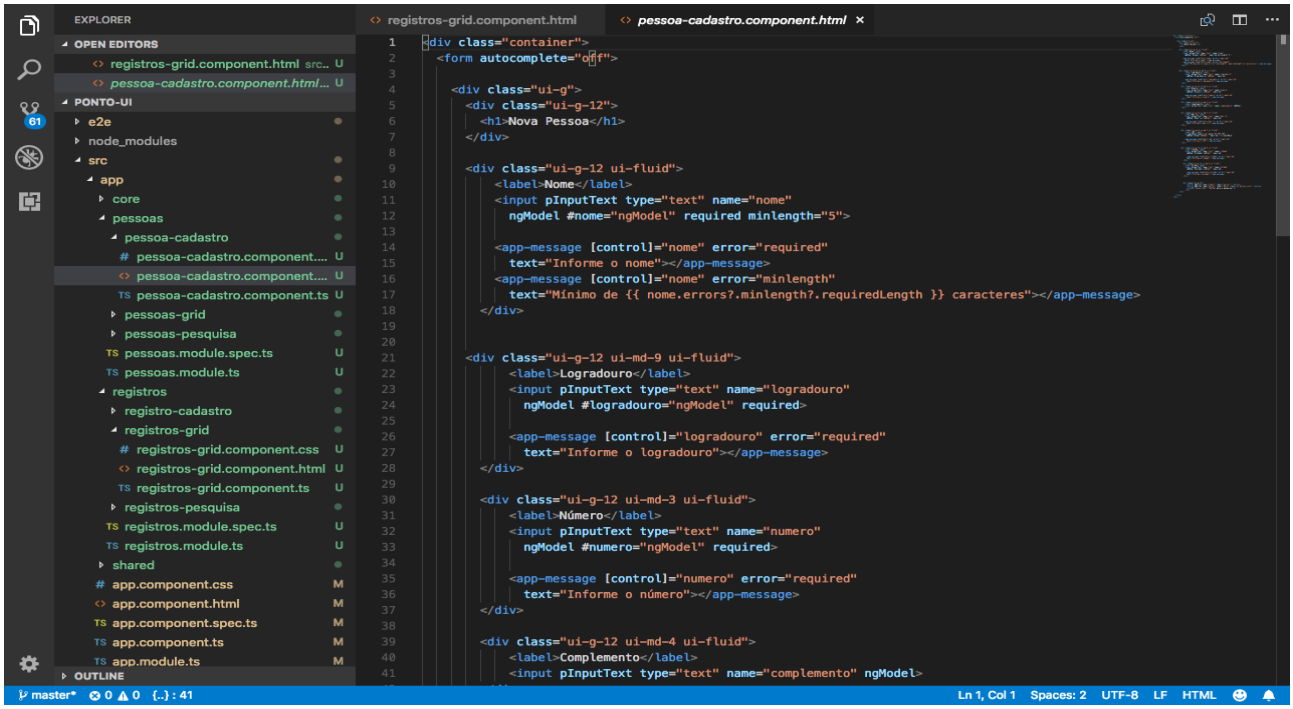


Figura 30 - Componente Pessoa-Cadastro

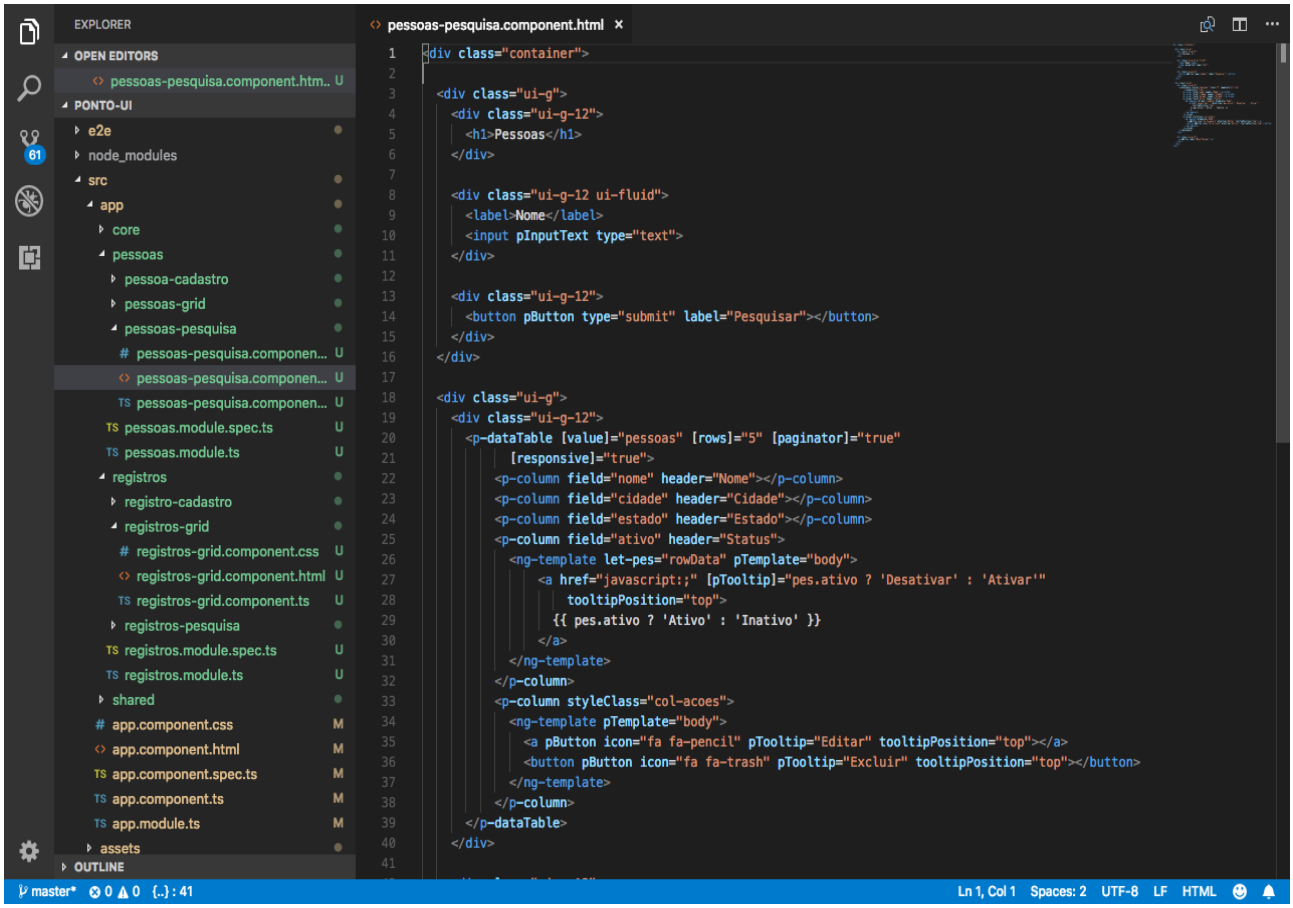


Figura 31 - Componente Pessoa-Pesquisa

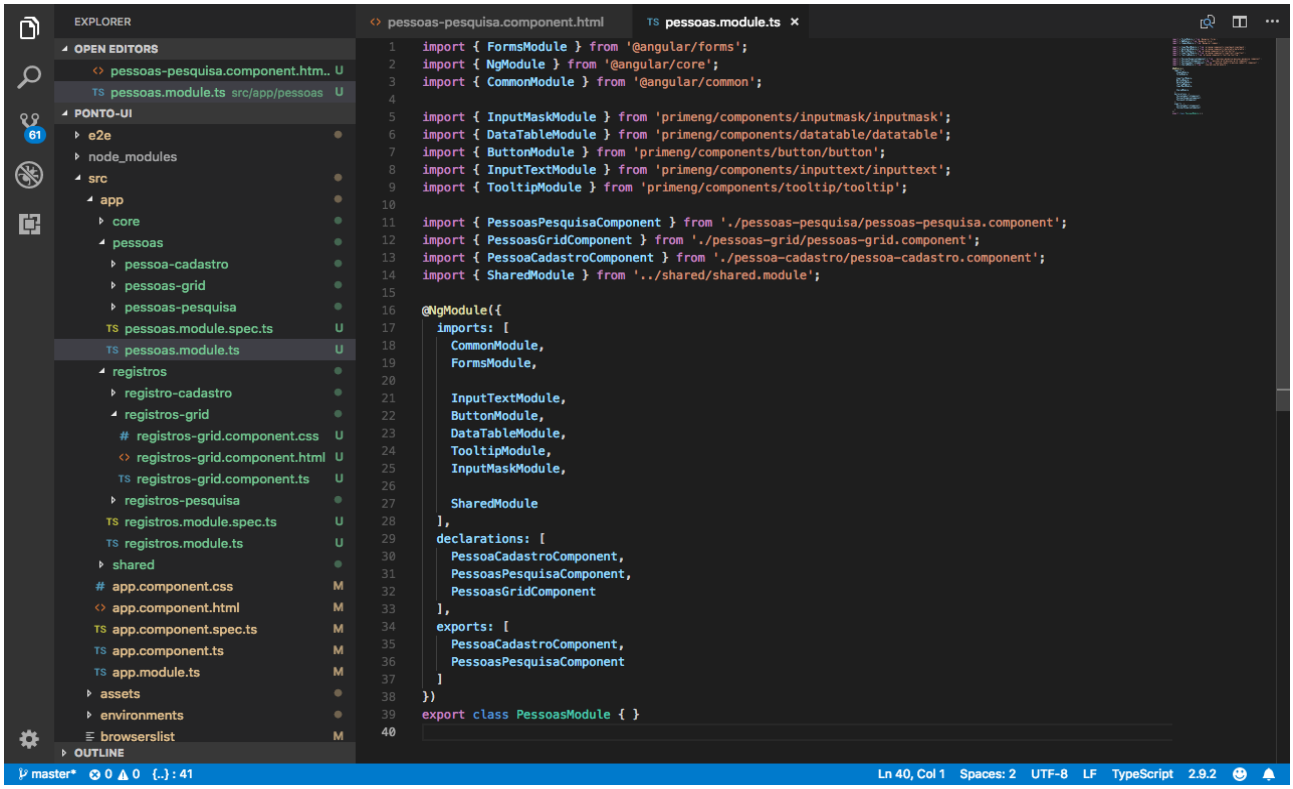


Figura 32 - Módulo Pessoa

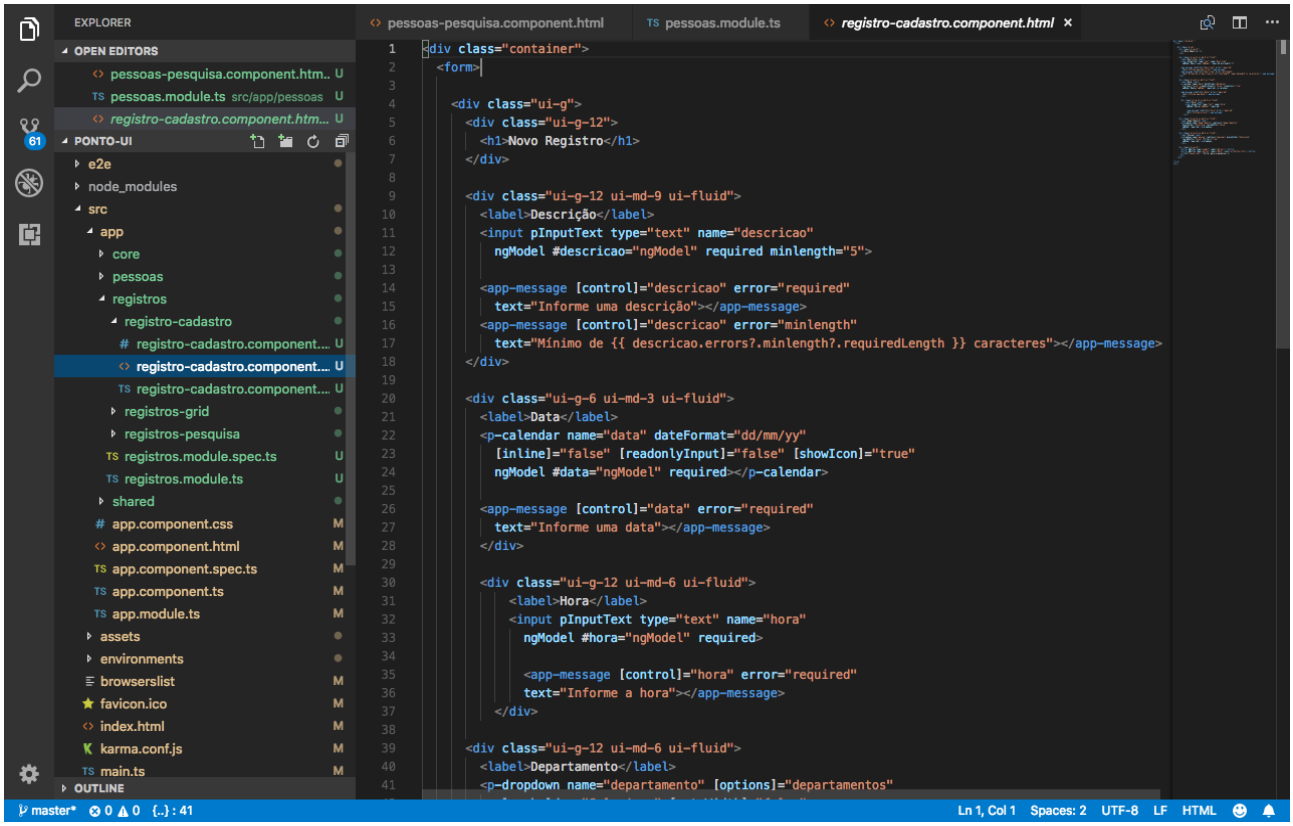


Figura 33 - Componente Registro-Cadastro

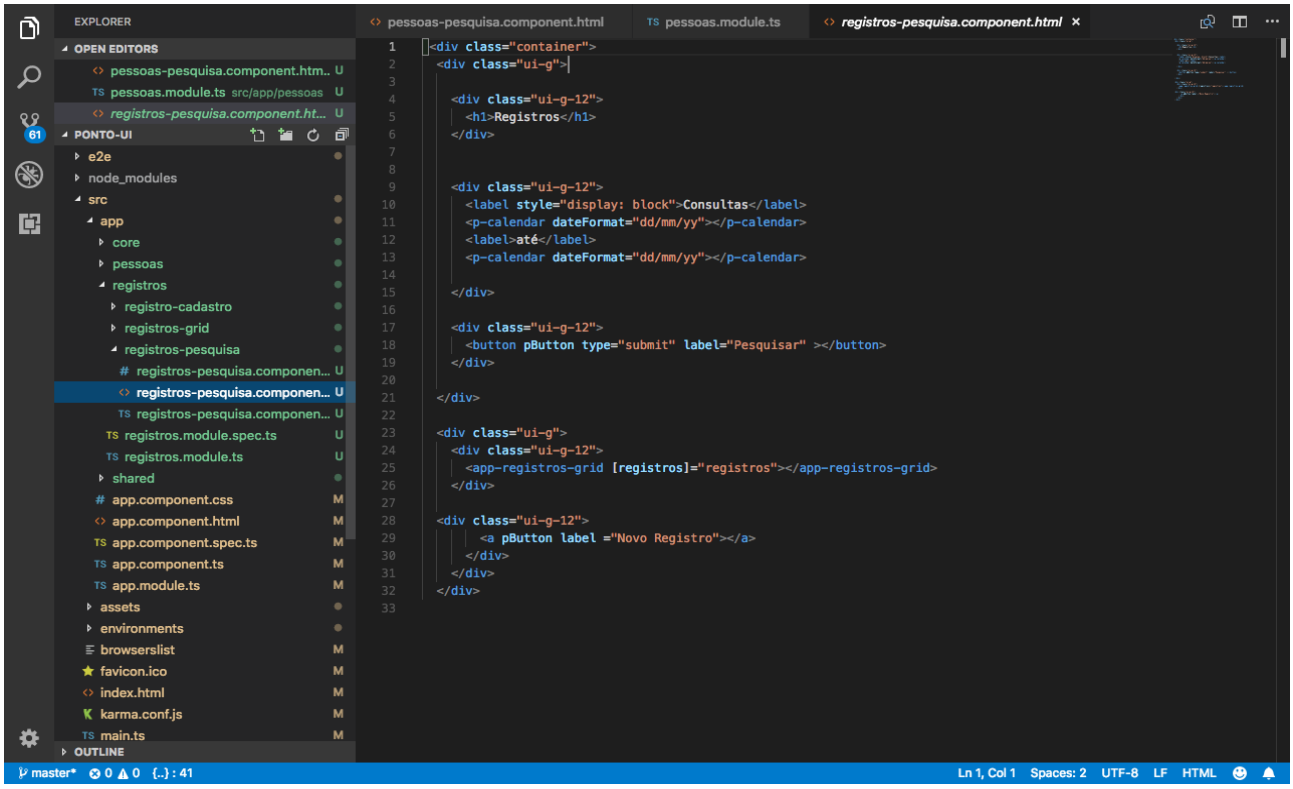


Figura 34 - Componente Registro-Pesquisa

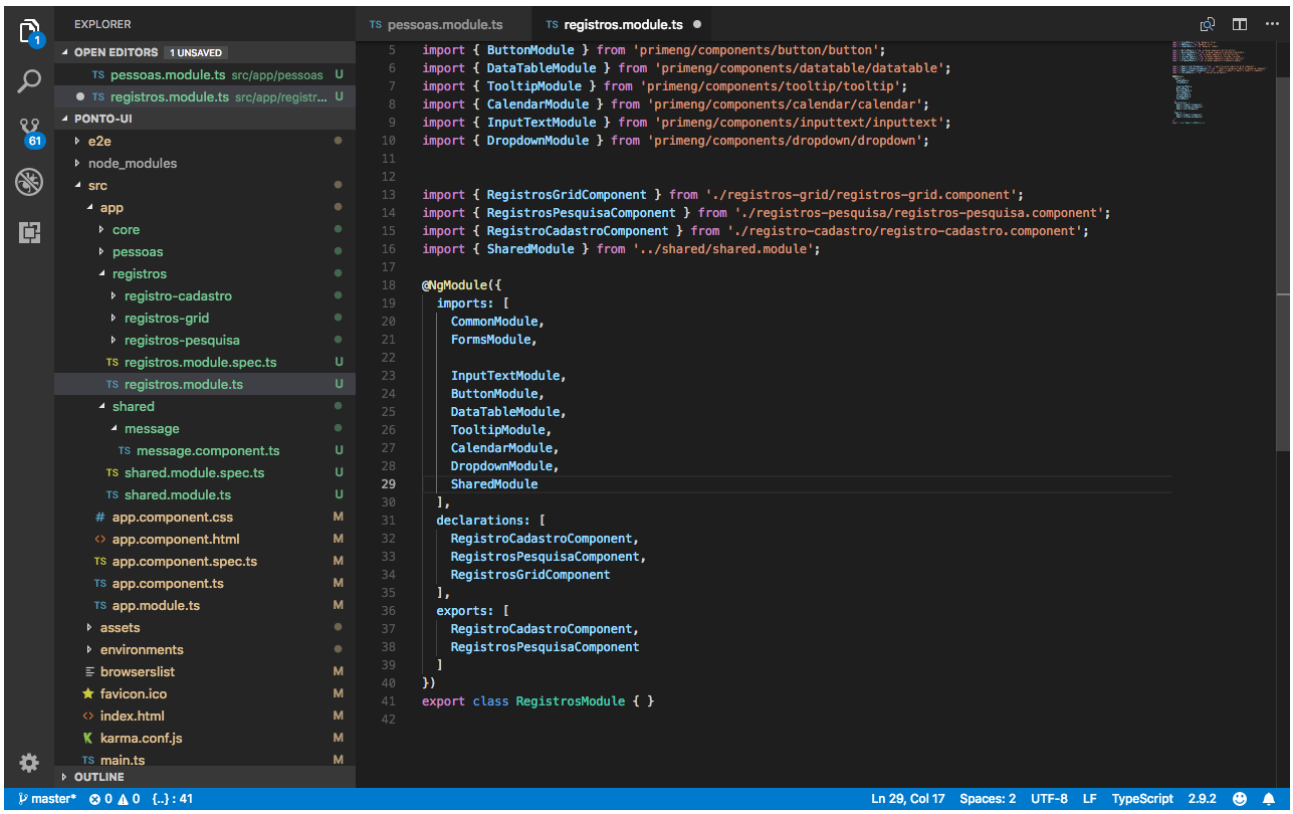


Figura 35 - Módulo Registro

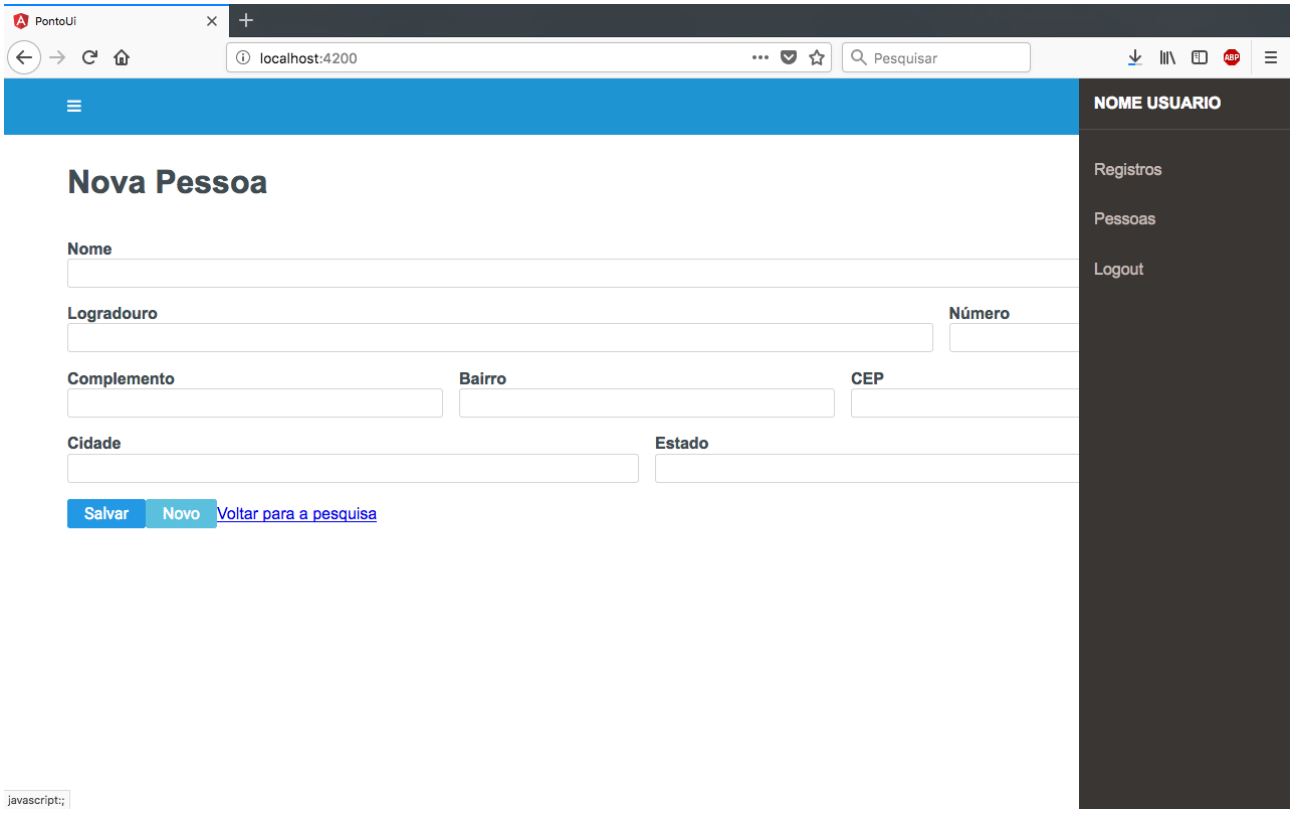


Figura 36 - Formulário Cadastro Pessoa

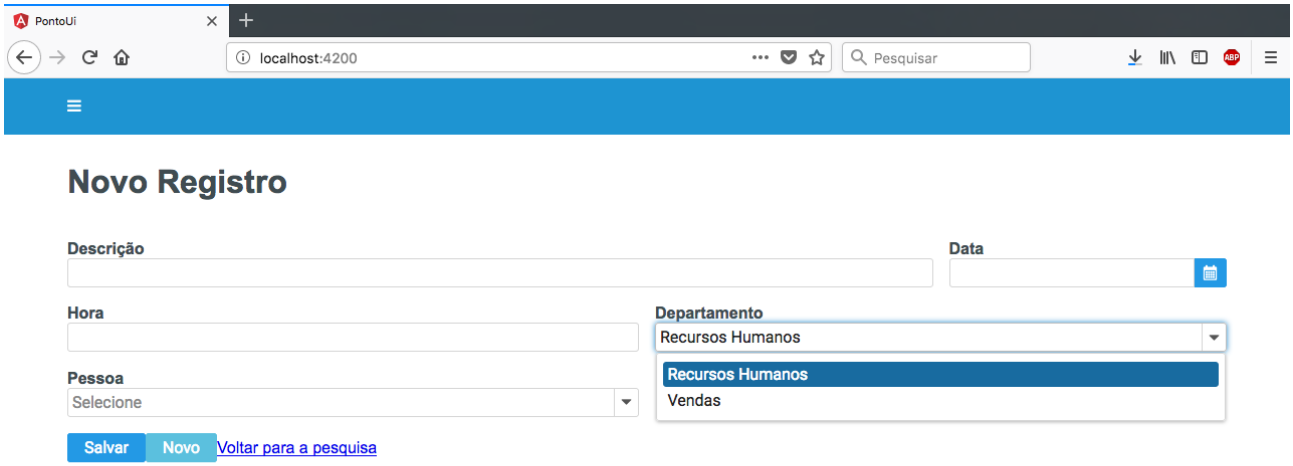
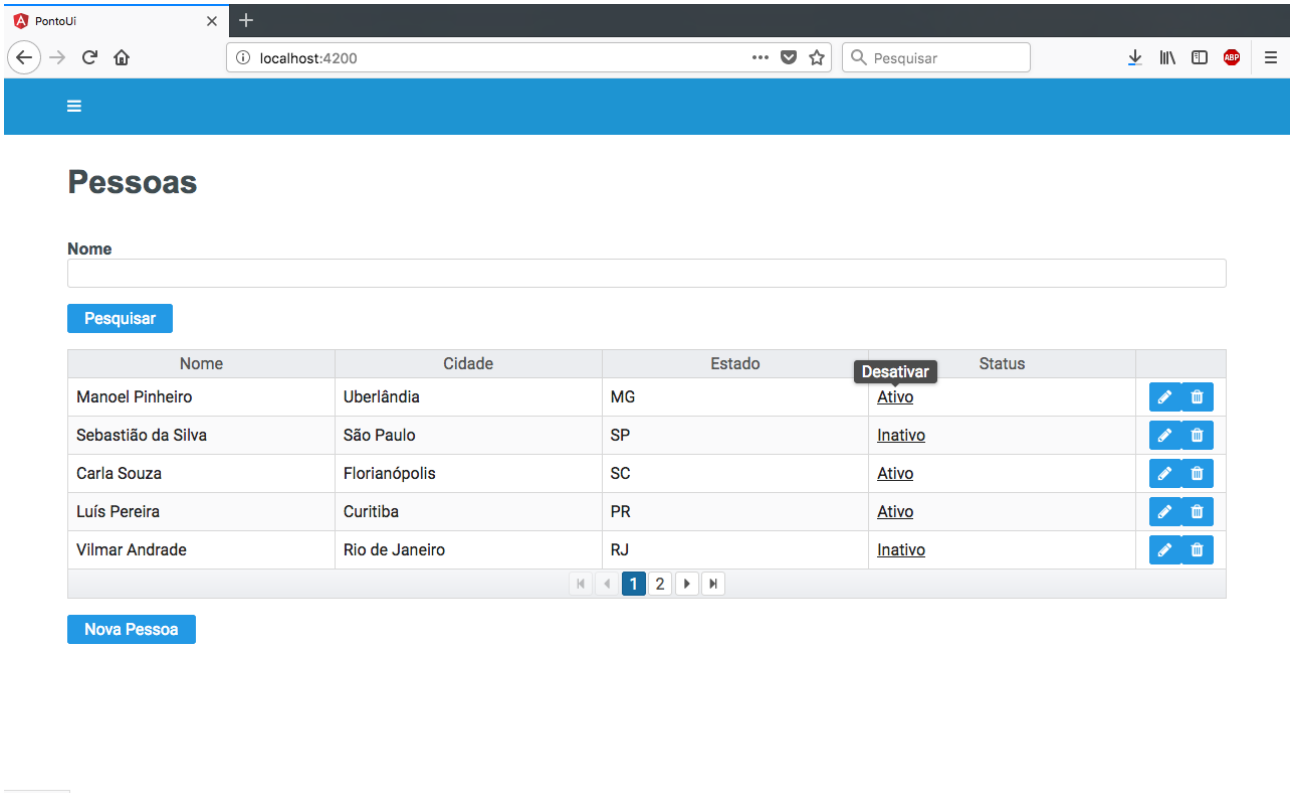


Figura 37 - Formulário Cadastro Registro



javascript:;

Figura 38 - Formulário Consulta Pessoa

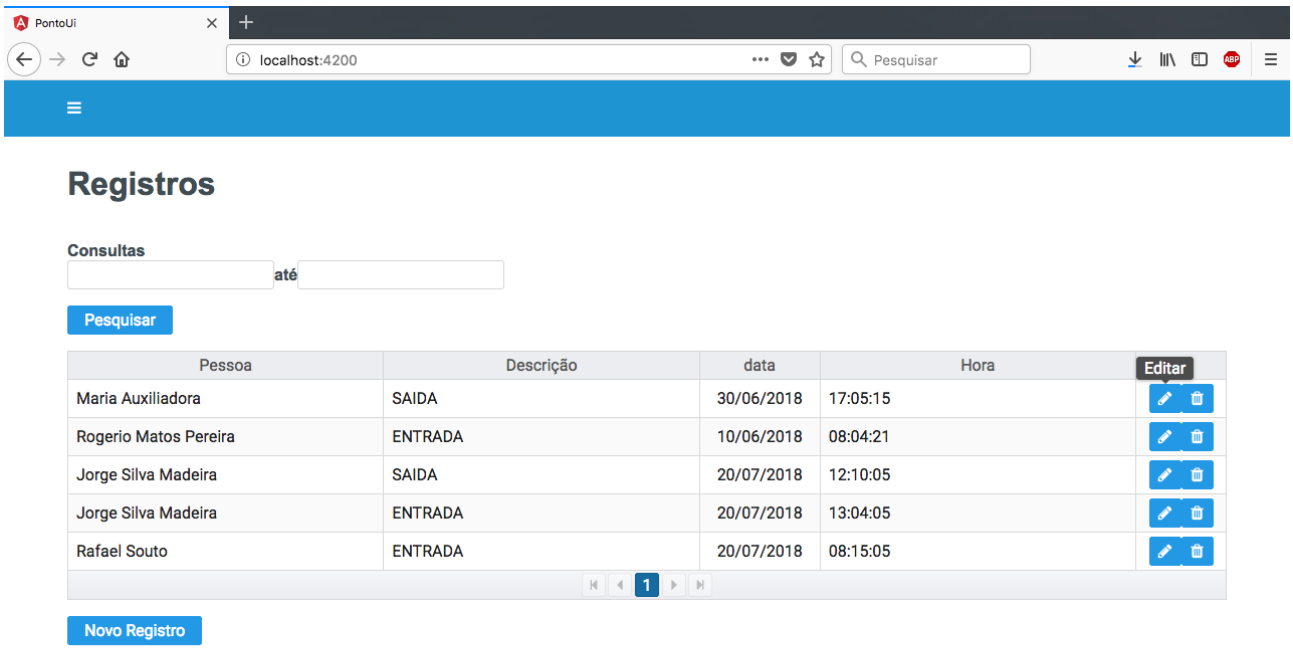


Figura 39 - Formulário Consulta Registro

5.3 Aplicação Reconhecimento Facial

Para a aplicação de reconhecimento facial foi utilizado a IDE PyCharm junto a biblioteca OpenCV.
A aplicação de Reconhecimento Facial foi dividida em três partes. Na primeira parte, foi desenvolvido o algoritmo chamado de captura.py. Conforme a imagem abaixo.

Figura 40 - Algoritmo Captura de Fotos

Para implementação desse algoritmo é necessário importar as bibliotecas cv2 e numpy.
A biblioteca cv2 permite a inclusão de diversas funções e recursos de suma importância para o desenvolvimento de toda a aplicação. Primeiro foram definidos dois variáveis de nome classificador e classificadorOlho.

Tais variáveis recebem o método CascadeClassifier da biblioteca importada cv2. Tal metódo utiliza o haarcascade já definido e treinado, permitindo assim a detecção de faces e olhos. Em seguida foi atribuído a variavél camera o método VideoCapture, que ativa a webCam do computador. O parâmetro (0) significa que está sendo chamada a primeira webCam do dispositivo.

Também foi definido a variavél amostra, que recebe o número de fotos que será capturado para cada pessoa. O tamanho da amostra é proporcional a confiança do

algoritmo, ou seja, quanto mais fotos, melhor a confiança do algoritmo. Em seguida foram definidas a variaveis id, que recebe o valor informado pelo usuário e largura que define o tamanho do quadro aberto pela webCam.

A bibiloteca cv2 disponibiliza três reconhecedores faciais. Porém para criá-los as imagens capturadas precisam estar em preto e branco, ou, em escala de cinza. Então para facilicitar a implementação futura dos algoritmos reconhecedores faciais, no momento da captura das imagens para armazenamento e posterior treinamento, transformamos as imagens em escala de cinza.

Para isso, foi feito um loop, onde foram definidas as variáveis conectado e imagem, que recebem o método camera.read. Tal método também é disponibilizado pela biblioteca cv2 e permite que as variáveis recebam a leitura da webCam. Também foi definida a variável imagemCinza que converte a imagem lida pela webCam em cinza, através do cvtColor.

Após, é declarado a variável facesDetectadas que recebe o classificador declarado previamente, junto do método DetectMultiScale com os parâmetros imagemCinza, scaleFactor=1.5 e minSize (80,80). Esse é o método que detecta a face da imagem. Por isso passamos a imagem capturada em escala de cinza, junto ao scaleFactor que faz o redimensionamento de um rosto maior para um rosto menor e por fim o parâmetro minSize que especifica o tamanho mínimo do menor objeto a ser reconhecido.
Depois é necessário um laço para que possa ser desenhado em volta de cada rosto detectado um retangulo para identificação. Portanto o laço é implementado em cima das faces detectadas. Para implementar esse laço é necessário passar os pontos iniciais da face (x,y) junto de sua largura e altura (l,a) junto da cor do retângulo (0,0,255) e largura da borda (2). A fim de melhorar a detecção foi implementado dentro desse laço um segundo laço para detectar olhos seguindo a mesma lógica para detectar faces, com a diferença de que o classificador haarcascade é um próprio já treinado com características de olhos e importados previamente no início do algoritmo.

Por fim são implementadas duas condições, a primeira define a iluminosidade mínima da imagem e a segunda, caso a primeira seja atendida, estabelece o caminho para salvar as imagens capturadas.

Em seguida foi implementado o algoritmo de codificação das imagens em vetores de 128 medidas únicas para cada face que foi armazenada na pasta ‘fotos’.

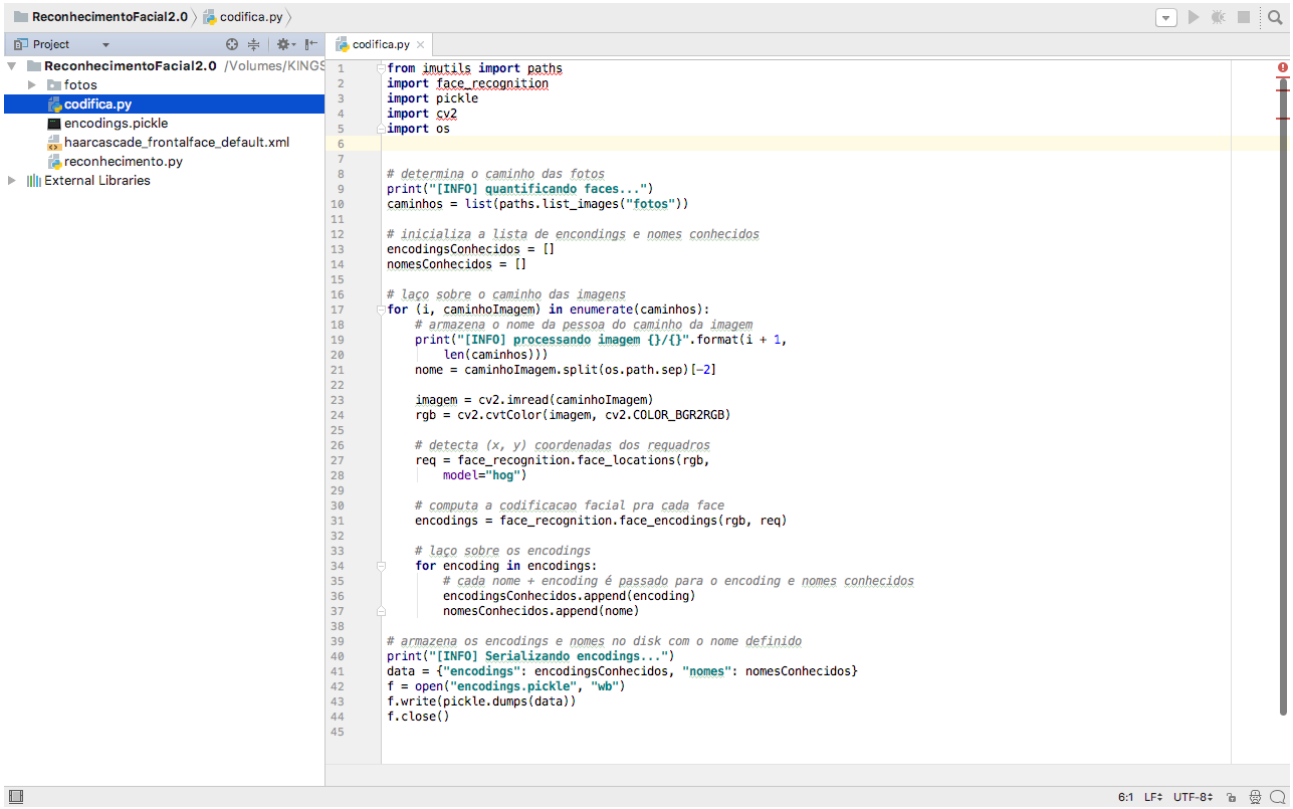


Figura 41 - Algoritmo Codifica.py

Na segunda parte, para a codificação do vetor e posterior serialização do mesmo, é necessário a implementação do algoritmo de codificação, esse processo de vetorização e codificação é necessário pois faremos uso de deep learning, portanto note que, além da biblioteca cv2 utilizamos ainda as bibliotecas face_recognition e imutils. Como o foco aqui está no sistema em si, não nos aprofundaremos na explicação das mesmas. Para gerar os vetores das faces, primeiro, precisamos passar o caminho de onde estão as nossas fotos tiradas com o algoritmo anterior. Portanto cria-se uma variável (caminhos) onde ela recebe o caminho das fotos armazenadas. Após, inicializa-se dois vetores(arrays) onde receberão os vetores codificados e os nomes das pessoas, respectivamente. Com os vetores inicializados, cria-se um laço sobre o caminho das imagens, para percorrê-las, armazenar o nome e convertê-las de BGR para RGB por definição das bibliotecas utilizadas, após, nas imagens percorridas é detectada a face e requadrada, pelo método de detecção “hog” (histograms of oriented gradients), note que para o requadro foi passado um método já existente na biblioteca face_recognition.

Após detectadas as faces, vamos codificá-las para os vetores de 128 medidas com o auxílio do método .face_encodings passando os parâmetros rgb e req, definidos previamente. Por fim percorremos um novo laço para armazenar os nomes e os encodings naqueles vetores definidos no início. Finalizado os laços, que vão ter o tamanho de acordo com o número de fotos tiradas, cria-se um arquivo que armazenará esses dados com a extensão *.pickle*. Note que para atribuir o caminho das imagens salvas, foi necessário a importação da bibioteca os. que permitiu tal atribuição.

Por último, foi implementado o algoritmo de reconhecimento com base nos vetores gerados pelo algoritmo codifica.py.

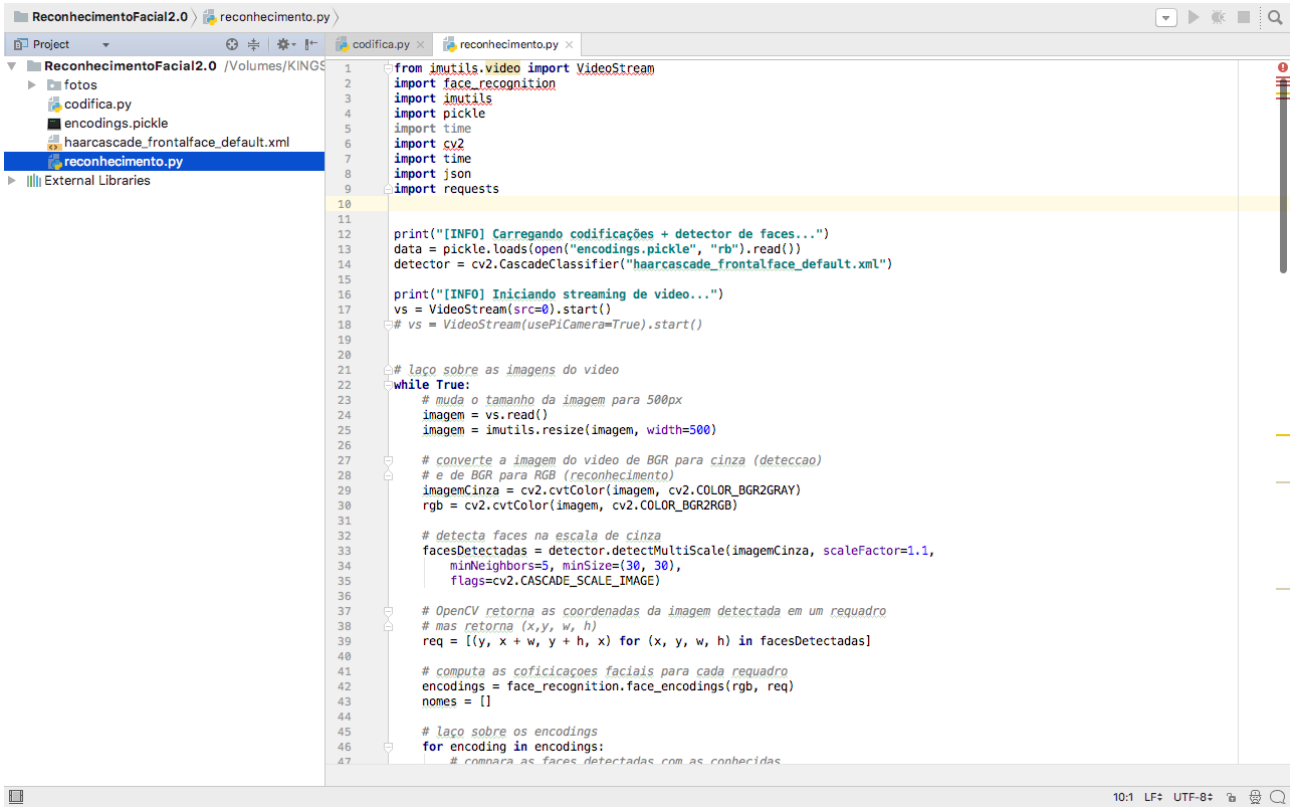


Figura 42 - Algoritmo Reconhecimento Facial

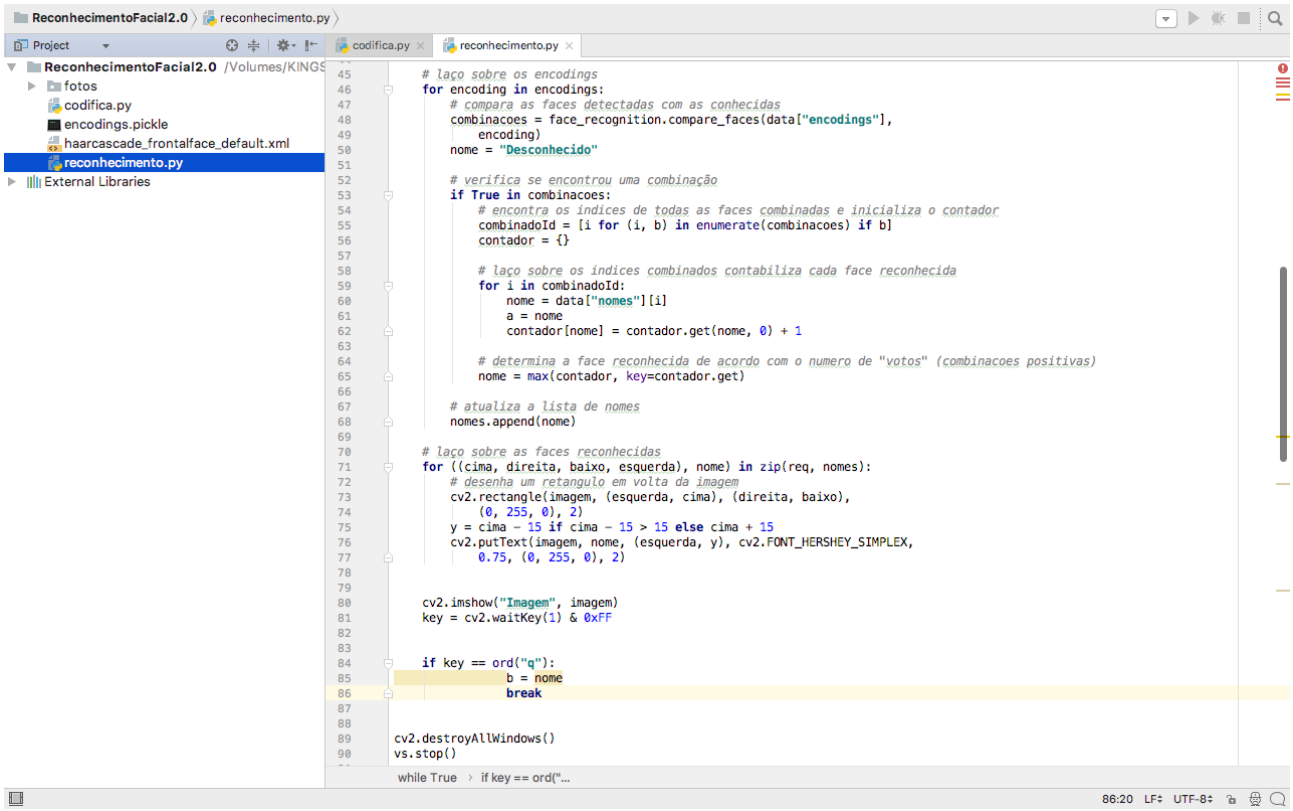


Figura 43 – Reconhecimento Facial 2

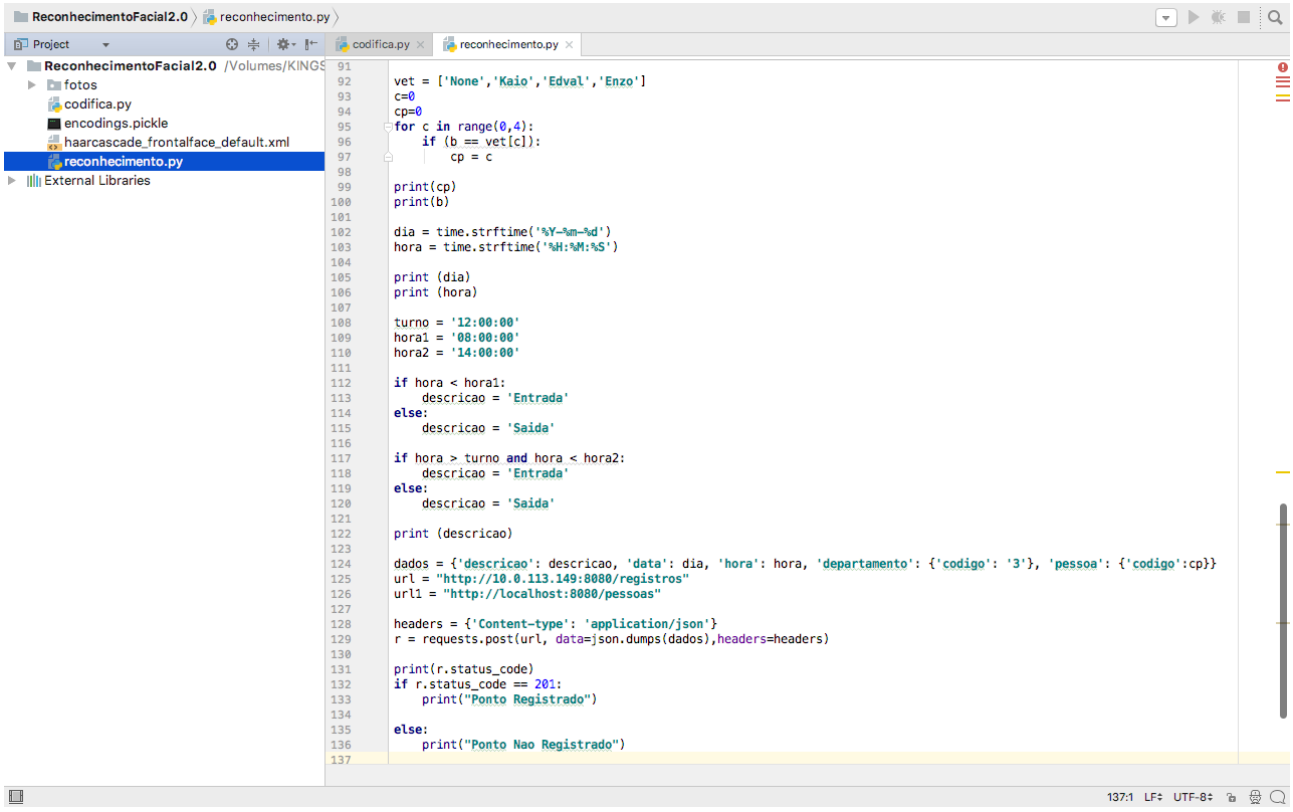


Figura 44 - Reconhecimento Facial 3

Na terceira parte da aplicação, foram definidos as variáveis que recebem tanto o haarcascade treinado para detecção de faces como o reconhecedor treinado no algoritmo anterior. Com isso definido, podemos fazer um loop como no primeiro algoritmo que chama a webCam e converter a imagem que mostrada pela webCam em escala de cinza.

Para depois fazermos o laço de requadro e compararmos a imagem atual com as imagens salvas. E se elas tiverem as mesmas características, é definido o nome da pessoa pelo id. Nesse algoritmo foram definidas apenas três pessoas para testes. O que não impossibilita de cadastrar novas imagens de pessoas diferentes.

Após as implementações apresentadas resta fazer o deploy da API e da Interface de Usuário em alguma máquina local, ou, subí-las na nuvem por meio de um servidor. Como a proposta é um protótipo as aplicações foram executadas em rede local.

6 - CONSIDERAÇÕES FINAIS

Após a implementação, é chegada a hora de testar as aplicações no Raspberry, porém como o mesmo trata-se de uma placa com processadores ARM, as aplicações STS(Spring Tool Suite) e Angular não compilam, sendo necessário o deploy das mesmas em algum serviço cloud, como por exemplo o heroku. Como nosso objetivo é simplesmente a prototipação as aplicações não foram hospedadas em serviços cloud.

Porém ainda é necessário testar o reconhecimento facial no Raspberry, para isso, é necessário a instalação das bibliotecas OpenCV, face_recognition, requests e imutils no mesmo, foi preciso o download e a compilação dessas bibliotecas no Raspberry. Além disso, pelo fato de o Raspberry utilizar processador ARM, a biblioteca necessita ser instalada no *source* do dispositivo, para evitar conflitos com outras bibliotecas. Para esse processo, foi criado um ambiente virtual no próprio Raspberry.

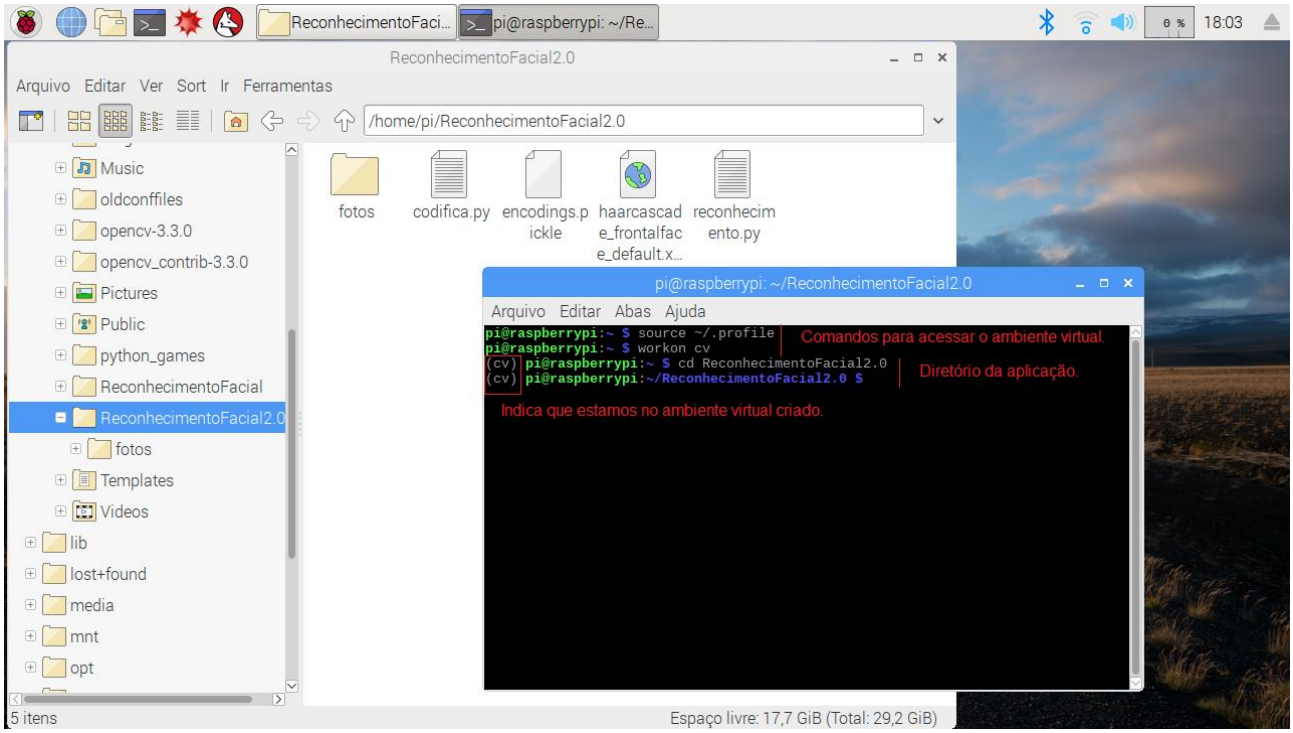


Figura 45 - Acesso ao ambiente virtual via terminal

Por fim, para testar a aplicação, abre-se o terminal e executa-se o comando “python3 nomedoalgoritmo.py”, no caso, foram testados os algoritmos captura.py, codifica.py e reconhecimento.py.

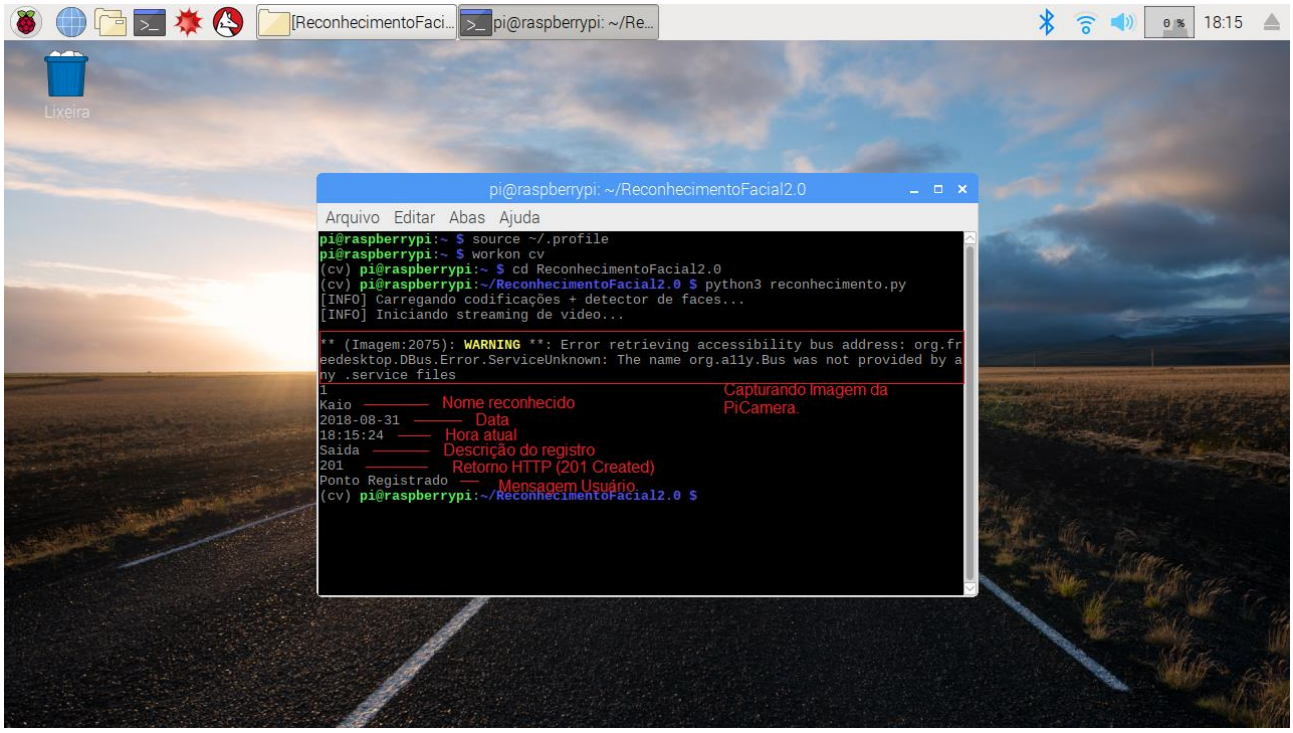


Figura 46 - Execução do Reconhecimento Facial

Conforme visto, a Inteligência Artificial e o Reconhecimento Facial já são uma realidade, e que podem contribuir para o desenvolvimento e melhoria das necessidades do nosso dia a dia. O protótipo proposto possui utilização nas mais variadas empresas e variadas ramificações com o auxílio da Inteligência Artificial e demanda do mercado. Claro que, no momento, é uma tecnologia em ascensão e com certeza sofrerá diversas atualizações, o que poderá acarretar em diferentes intervenções nos projetos futuros. Considerando que meu conhecimento não é muito aprofundado acerca dessa área, no momento, provavelmente a abordagem utilizada, não tenha sido a mais eficaz. O que pode acarretar em alterações futuras no projeto, conforme o amadurecimento na tecnologia.

REFERÊNCIAS

ANGULAR: **Angular**. Disponível em: < https://www.angular.io/docs >. Acessado em 03 de Junho de 2018.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivan. **UML Guia do Usuário**. New York. Editora Apress. 2006.

BRADSKI,KAEHLER, Grary, Adrian. **Learning OpenCV**. Sebastopol: Editora O'Reilly Media. 2008.

KAPUR, Saurabh. **Computer Vision with Python 3**. 2017.

LOPES, Eduardo**. Detecção de Faces e Característica Faciais**. Disponível em <http://www.pucrs.br/facin-prov/wp-content/uploads/sites/19/2016/03/tr045.pdf>. Acessado em 01 de Junho de 2018.

MATTHES, Eric. Curso **intensivo de Python**. 1ª Edição, Novatec, 2016.

MENEZES, Nilo Ney Coutinho. **Introdução a programação com Python.** 2ª Edição, Novatec, 2014.

OPENCV: **Opencv**. Disponível em: <https://opencv.org/>. Acessado em 01 de Junho de 2018.

POLETTO, Alex S. R. S. **Banco de Dados I**. 2013.

PRESSMAN, Roger. **Engenharia de Software – Uma Abordagem Profissional**. Porto Alegre. AMGH Editora Ltda. 2011.

RASPBERRY: **Raspberry**. Disponível em: < https://www.raspberrypi.org />. Acessado em 18 de Abril de 2018.

SOMMERVILLE, Ian. **Engenharia de Software**. São Paulo: Pearson, 2011.

SPRING: **Spring Tool Suite**. Disponível em: < https://spring.io/tools>. Acessado em 01 de Junho de 2018.

TURK, Matthew, PETLAND, Alex. **Face Recognition Using Eigenfaces**. Disponivel em <https://pdfs.semanticscholar.org/8a71/ba0bbba3ef9c542040ec48d53889c70fb243.pdf>. Acessado em 31 de Maio de 2018.