CURSO: Análise de Desenvolvimento de sistemas
POLO DE APOIO PRESENCIAL: Campinas, Higienópolis, Brás e Osasco
SEMESTRE: 5/2020
COMPONENTE CURRICULAR / TEMA: PR PROF ADS – Aula 2 – Sintetize
NOME COMPLETO DO ALUNO: Gabriel Diniz Mart Manuel Alejandro Martinez Escalona Fabiano Barros de Brito Wilson Ferreira Dias Junior
TIA: 19000741, 19004869, 19002424 e 19009836
NOME DO PROFESSOR: Fábio Silva

FSL

Documento de arquitetura de Software

Versão <1.0 >

Histórico de Revisões

Data	Versão	Descrição	Autor
21/04/2021	1.0	Release Inicial	Gabriel Diniz

Sumario

1. Introdução

1.1 Finalidade

Este documento oferece uma visão da arquitetura geral do sistema FSL, utilizando diferentes visões para representar os aspectos do mesmo. O objetivo do documento é capturar e comunicar as decisões arquiteturais significativas que foram tomadas no decorrer do desenvolvimento do sistema.

1.2 Escopo

As definições contidas neste documento auxiliam os envolvidos no projeto a captar os aspectos arquiteturais do mesmo, que são necessários para seu desenvolvimento.

Estão descritos neste documento: padrões adotados, frameworks e linguagens.

1.3 Definições, Acrônimos e Abreviações

- *MVC – Model View Controller:* Padrão de arquitetura de software constituído por três camadas.
- *MTV – Model Template View:* Padrão de arquitetura de software utilizado pelo Framework Django.

1.4 Visão Geral

Neste documento serão apresentadas as visões arquiteturais. Estas visões têm por objetivo segmentar cada parte do comportamento em diferentes processos do sistema e assim reduzir a complexidade, aplicar boas práticas e facilitar o caminho a ser percorrido no desenvolvimento.

2. Representação Arquitetural

- **Visão de caso de uso**: Descreve o sistema e suas funcionalidades, permitindo visualizar os atores e suas interações através de ações;

- **Visão lógica:** Apresenta uma visão da estrutura estática do sistema, suas classes e sua organização arquitetural que será utilizada no desenvolvimento;

- **Visão de processos**: Apresenta o padrão de comportamento do sistema diante das diferentes ações, visando o desempenho, confiabilidade e tolerância a falhas;

- **Visão de implantação:** Apresenta comunicação do sistema, uma representação dos aspectos físicos ou da organização. Será representada pelo diagrama de implantação;

- **Visão de implementação:** Descreve o ponto de vista do programador, mostra uma distribuição dos pacotes e componentes do sistema.

3. Metas e Restrições da Arquitetura

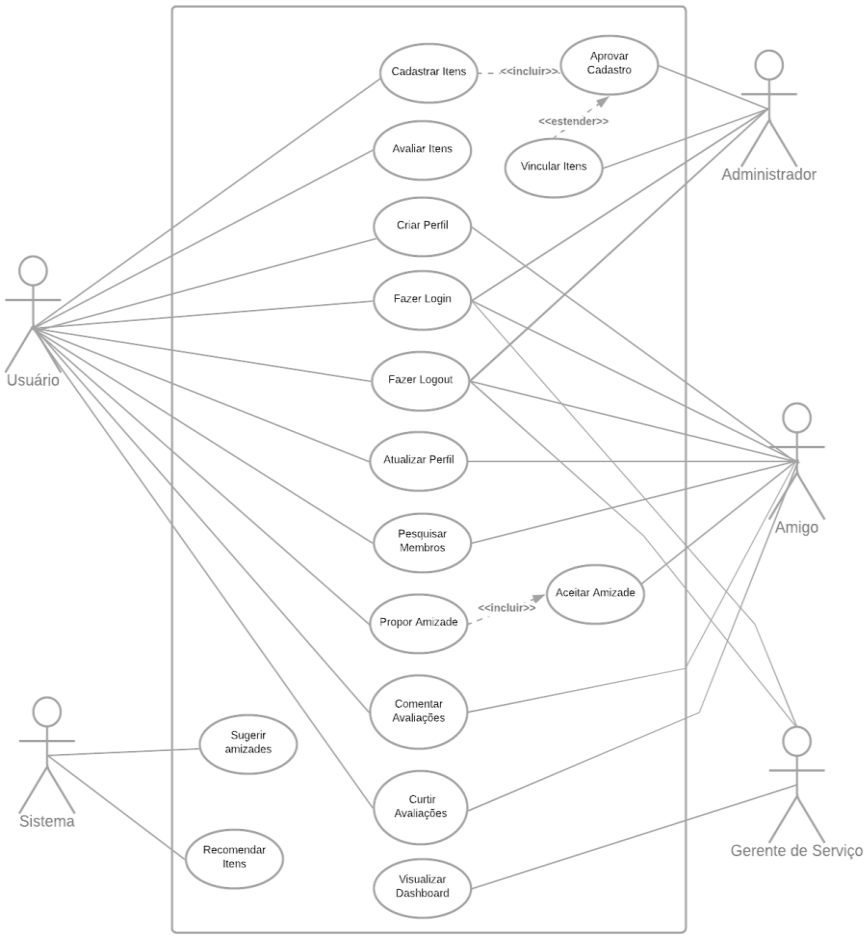
Para o desenvolvimento deste projeto serão utilizadas as seguintes tecnologias:

- Python: Versão 3.5+ como linguagem base do *back-end*
- Django: Framework para o desenvolvimento de aplicações web em *Python*
- Bootstrap: Framework para a construção do *front-end*
- PostgreSQL: Banco de dados relacional
- Heroku: Serviço de Cloud

Requisitos não funcionais e Metas:

- Front-end responsive para mobile.
- Diferenciação de tipo de usuário na autenticação.
- Sistema multiplataforma.
- Disponibilidade de 99.99% em regime de 24X7.

4. Visão de Caso de Uso



5. Visão Lógica

5.1 Visão Geral

A visão lógica define a estrutura da arquitetura. Abaixo especificamos o padrão utilizado para o desenvolvimento do sistema, no caso, o MVC, que Django implementa com uma variante e diferenciando os nome das camadas (MTV):

Padrão MVC	Padrão MTV (Django)

Model	Model
View	Template
Controller	View

- **Model:** é responsável pelo mapeamento do banco de dados, sendo formado por modelos, onde cada modelo representa uma tabela e seus atributos representam os campos da tabela;

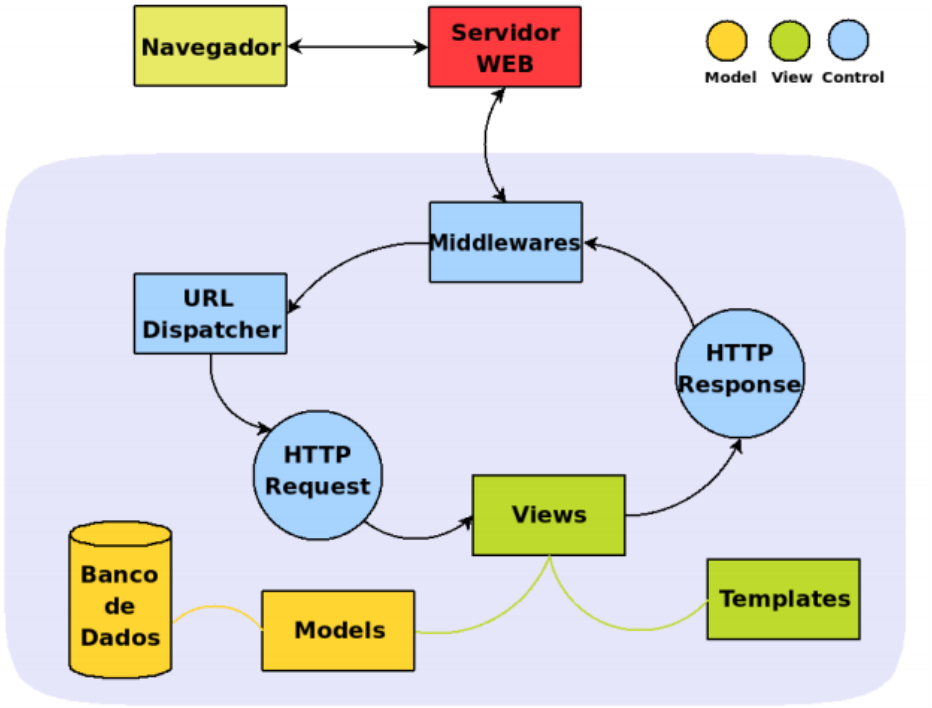
- **Template**: é responsável de exibir as informações para o usuário da aplicação, normalmente utilizando o HTML, CSS e JavaScript;

- **View:** é responsável por recepcionar, processar e responder requisições, tratando do acesso ao banco de dados. Esta camada é responsável pelas regras de negócios.

A camada de *View* (MTV) e *Controller* (MVC) de ambos os padrões, ainda que possuam responsabilidades semelhantes na parte conceitual, apresentam algumas diferenças que são resumidas pelo próprio framework na seção FAQs:

> *"Em nosso entendimento de MVC, a "view" representa os dados que são apresentados ao usuário. Não é necessariamente* como *a informação* é apresentada*, mas* qual *informação é mostrada. A view representa* qual informação você vê*, não* como você vê*. Há uma sutil diferença.*
>
> *Então, nesse caso, uma "view" é uma função de retorno para uma URL específica, por que esta função de retorno descreve qual informação é apresentada.*
>
> *[...] No Django, uma "view' descreve qual informação é apresentada, mas uma view normalmente delega para um template, que descreve* como *a informação é apresentada.*
>
> *Onde o "controller" se encaixa, então? No caso do Django, é provavelmente o próprio framework"*

Dessa forma, no modelo MVC no *Controller* é preciso escrever todo o código específico referente ao controle, já no MTV uma parte do controlador é cuidada pelo próprio framework, através dos componentes como *o URL dispatcher, middlewares e handlers*.



De acordo como o Django Book, o Django segue o padrão MVC suficientemente para ser considerado um framework MVC.

6. Visão de implantação

A visão de implantação será realizada ao final do projeto e será representada pelo *diagrama de implantação.*

7. Visão de implementação

O sistema será implementado utilizando os conceitos de Programação Orientada a Objetos através do framework Django, estrutura MTV (MVC), Linguagem de programação Python, Bootstrap, e banco de dados PostreSQL.

No projeto Django, serão criados dois APPS, um para os usuários e outro para os itens que compõem o projeto. Cada app é composto pelos seguintes arquivos:

- **models.py** - implementa a camada model e as validações personalizadas dos dados que serão guardados no banco de dado;

- **views.py** - implementa a camada view, que é responsável pela interação com a model e por processar todos os dados;
- **urls.py** - endpoints que permitem acesso às views;
- **tests.py** – arquivo para testes.

8. Tamanho e Desempenho

A aplicação deve rodar através de um site próprio na versão web, sendo responsiva para dispositivos mobile e disponibilizada em um provedor de serviços na internet, atendendo o padrão de disponibilidade de 99.99% em regime de 24X7.

9. Qualidade

A arquitetura adotada por camadas busca dar mais produtividade ás implementações e otimizar a manutenção do sistema, garantindo uma melhor organização do código fonte e sua portabilidade.

10. Referências

[Django Book] https://djangobook.com/

[Django Documentação] https://docs.djangoproject.com/pt-br/3.2/faq/

[Modelo MVT em Django] https://github.com/fga-eps-mds/A-Disciplina/wiki/Padr%C3%B5es-Arquiteturais---MVC-X-Arquitetura-do-Django