**Principais Linguagens de Programação e *Frameworks* utilizados no Desenvolvimento *Front-End* e *Back-End*.**

O desenvolvedor *web* deve compreender o que são e como funcionam as principais linguagens de programação para a produção e manutenção de Soluções *Web*, para que saiba determinar qual é a melhor ferramenta de trabalho, em cada situação.

**Front-end** ou **interface frontal,** é um termo que se refere à etapa inicial de um processo.

**Back-end** ou **parte secundária, parte de suporte, parte de retaguarda**, é um termo que se refere à etapa final de um processo.

**Front-end** e **back-end** são dois termos empregados no desenvolvimento e utilização de tecnologias para a web. Os profissionais desenvolvedores *back-end* e *front-end* são os responsáveis pela realização do planejamento, produção e implementação de toda a estrutura de um site ou de uma aplicação web. O desenvolvedor ***back-end*** é o profissional que cuida da parte oculta da aplicação *web*. O desenvolvedor ***front-end*** é o profissional que atua na área responsável por programar a parte visível de uma solução web ou um aplicativo *web*.

**Linguagem de programação Python**

Uma linguagem de programação permite descrever, de forma compreensível para o computador, os passos a serem por ele realizados para que a tarefa requerida seja executada. As sequências de passos escritas nessas linguagens são chamadas de **programas**. A atividade de escrever um programa é denominada **programação**. (CARVALO E LORENA, 2017)

Existem inúmeras linguagens de programação, o que nos leva a preferências individuais quando da escolha da linguagem. Adotamos a linguagem de programação Python, por diferentes razões: ela foi assunto das aulas Introdução a Conceitos de Computação, Algoritmos e Programação de Computadores I e II e Formação Profissional em Computação; pela sua facilidade de uso e pelos recursos que ela oferece ao desenvolvedor. É um Código livre o que significa que pode ser utilizado e distribuído livremente. Python é uma linguagem de programação de alto nível e orientada a objetos, com tipos dinâmicos e que se adapta bem para o desenvolvimento de sistemas web. É uma linguagem cuja curva de aprendizagem o torna fácil de entender e aprender como funciona. Devido a sua aplicabilidade num grande número de áreas de desenvolvimento de software, principalmente aqueles solicitados no desenvolvimento deste projeto integrador, como sistemas gerenciadores de banco de dados, *SQL* e *MySQL* e desenvolvimento de programação para internet, optamos pela sua utilização na aplicação de soluções de problemas com a produção de uma solução Web com o uso de *Python*.

**Linguagens utilizadas para o desenvolvimento web**

**HTML5** (*Hypertext Markup Language, versão 5*) – é uma linguagem de marcação própria para o desenvolvimento de aplicações web na **WWW** (*World Wide Web*), que apresenta importantes funcionalidades como semântica e acessibilidade.

**CSS** (*Cascading Style Sheets*) – é uma linguagem de folhas de estilo, mecanismo de desenvolvimento web, usado para acrescentar estilo – cores, fontes, espaçamento, etc. – a uma aplicação web. Atualmente a versão em uso é a CSS3.

**Frameworks CSS** – trata-se de um conjunto de componentes de programação que fornecem uma estrutura básica de elementos reutilizáveis, com uma arquitetura consistente de funcionalidades sob a qual a aplicação web será desenvolvida. Frameworks CSS fornecem ferramentas de definição da folha de estilo, design responsivo, tipografia própria para web, botões, ícones, fontes e formulários, entre outras funcionalidades.

Dentre os inúmeros Frameworks CSS, optamos pelo uso do ***Bootstrap,*** pela sua popularidade e qualidades relativas ao Desenho Web Responsivo.

**Javascript** - é uma linguagem de programação interpretada estruturada, de script de alto nível com tipagem dinâmica fraca e multiparadigma (protótipo, orientado a objeto, imperativo e funcional). **ECMAScript 5** é a versão da linguagem básica. **V8** é o nome do interpretador Javascript, também chamado de **Máquina Virtual Javascript** ou **Engine**, desenvolvido pela Google Chrome. V8 é uma ferramenta desenvolvida na linguagem C++ e distribuída no regime de código aberto.

***Framework* *front-end* para desenvolvimento *Web***

Na área de desenvolvimento de softwares, denominamos de *framework* a abstração que une os códigos comuns entre os vários projetos de softwares, permitindo uma funcionalidade genérica entre eles. Um *framework* atua onde ocorre funcionalidades comuns a várias aplicações. Desenvolver uma solução web completamente do zero é uma tarefa muito trabalhosa e demanda muito tempo. A utilização de softwares como os *frameworks* para desenvolvimento de soluções *web*, nos permitem gastar menos tempo e diminuir o nível de complexidade dos códigos empregados aplicando funcionalidades, comando e estruturas já prontas. Cada linguagem de programação possui características diferentes, o que leva a utilização de diferentes tipos de frameworks. Alguns exemplos de frameworks para aplicação web, que permitem suportar a criação de sites web dinâmicos e serviços web são: *Framework* de *Python*: ***Django***e ***Flask***, *Framework* de *Ruby*: ***RubyOnRails***, *Framework* de *Java*: ***Spring MVC***, *Framework* de PHP: ***Laravel***, ***CodeIgniter***, ***Symfony*** e ***CakePHP***,

**Vantagens de um *Framework***

* Produtividade – não requer que o usuário desenvolva os códigos primários, como *HTML*, *CSS* ou *Javascript* na produção de solução web, o que permite terminar um projeto em menos tempo.
* Padronização de codificação e processamento;
* Reaproveitamento de código;
* Segurança;
* Extensibilidade;
* Suporte e apoio por outros desenvolvedores.

**Desvantagens de um *Framework***

* Curva de aprendizagem – varia em função do *framework* escolhido.
* Dependência – Uma solução web executada com um *framework*, dificilmente será compatível com outro *framework*.
* Acomodação do programador – O uso constante de *framework* leva ao usuário a negligenciar o conhecimento da linguagem nativa.

Dentre os *frameworks* disponíveis no mercado, e baseados no conteúdo das aulas anteriores, e pelas características inerentes, tais como velocidade, linguagem de programação *Python*, demos preferência pela utilização do *framework* ***Django*** para Python em detrimento do *framework* ***Flask***.

***Framework* para *Python* – *Django***

***Django*** é um framework de alto nível, escrito na linguagem de programação *Python* que estimula o desenvolvimento de aplicações para solução web. *Django* é responsável pela parte mais complicada e trabalhosa do desenvolvimento web, como o tratamento de requisições, o mapeamento objeto-relacional, a preparações de resposta HTTP, permitindo ao desenvolvedor dispender menos tempo e menos energia na produção de soluções *web*.

O ***Framework*** ***Django*** foi desenvolvido para escrever aplicações *Web* dirigidas a bancos de dados, com a finalidade principal de tornar as tarefas comuns do desenvolvimento Web rápidas e fáceis. Ele permite o desenvolvimento rápido e a criação de código limpos.

Diferentemente dos outros frameworks para web, que são **MVC** (*Model View Controller)*, *Django* é declarado como um **MTV** (*Model-Template-View*). Nesse caso, a *View* representa ***qual*** informação o desenvolvedor vê e não ***como*** ele vê. Uma *View*, nesse caso, é uma forma de processar os dados de uma URL específica, visto que este método descreve qual é a informação apresentada.

Outra característica do Django são os *templates*, onde ocorre a separação entre o conteúdo e a apresentação. Uma *View* descreve ***qual*** é a informação apresentada, mas é o *template* que descreve **como** a informação é apresentada. O próprio *framework* *Django* é quem faz o trabalho de processar e rotear uma requisição para a *View*, de acordo com a configuração de URL declarada pelo desenvolvedor.

A figura (numerar) descreve o fluxograma de uma requisição vindo do *browser* do usuário e, passando para um servidor onde o *framework* *Django* está sendo executado, e retorna ao *browser* do usuário. O *framework* *Django* é compreendido por três camadas distintas:

* Camada de modelos – *(Model Layer)*,
* Camada de visualização – *(Model View)* e
* Camada de *templates* – *(Model Template View)*.

**Camada de modelos**

Modelos são a descrição dos dados que serão gerenciados pela solução *web* desenvolvida. Eles contêm todos os campos e os comportamento desses dados – equivalem à uma tabela de um banco de dados – o que facilita o gerenciamento do projeto de dados.

**Camada de visualização**

A camada de visualização é responsável por recepcionar e processar as requisições advindas dos usuários, conformar uma resposta e devolvê-la ao usuário. É a partir do endereço URL que o usuário quer acessar, que o framework Django irá servir de roteador para a requisição solicitada. Antes, o Django é informado para onde o usuário quer enviar esta requisição.

**Camada de templates**

A camada de template contém o código da linguagem Python, que irá renderizar as páginas *webs* criadas, e os arquivo em *HTML*, *CSS* e *Javascrip* que são responsáveis pela conformação da nossa aplicação. Um template é um arquivo de texto que será transformado em um outro arquivo, por exemplo, um HTML ou um CSS. A **DTL** – Django Template Language é a linguagem de programação padrão dos templates do *framework* *Django*.

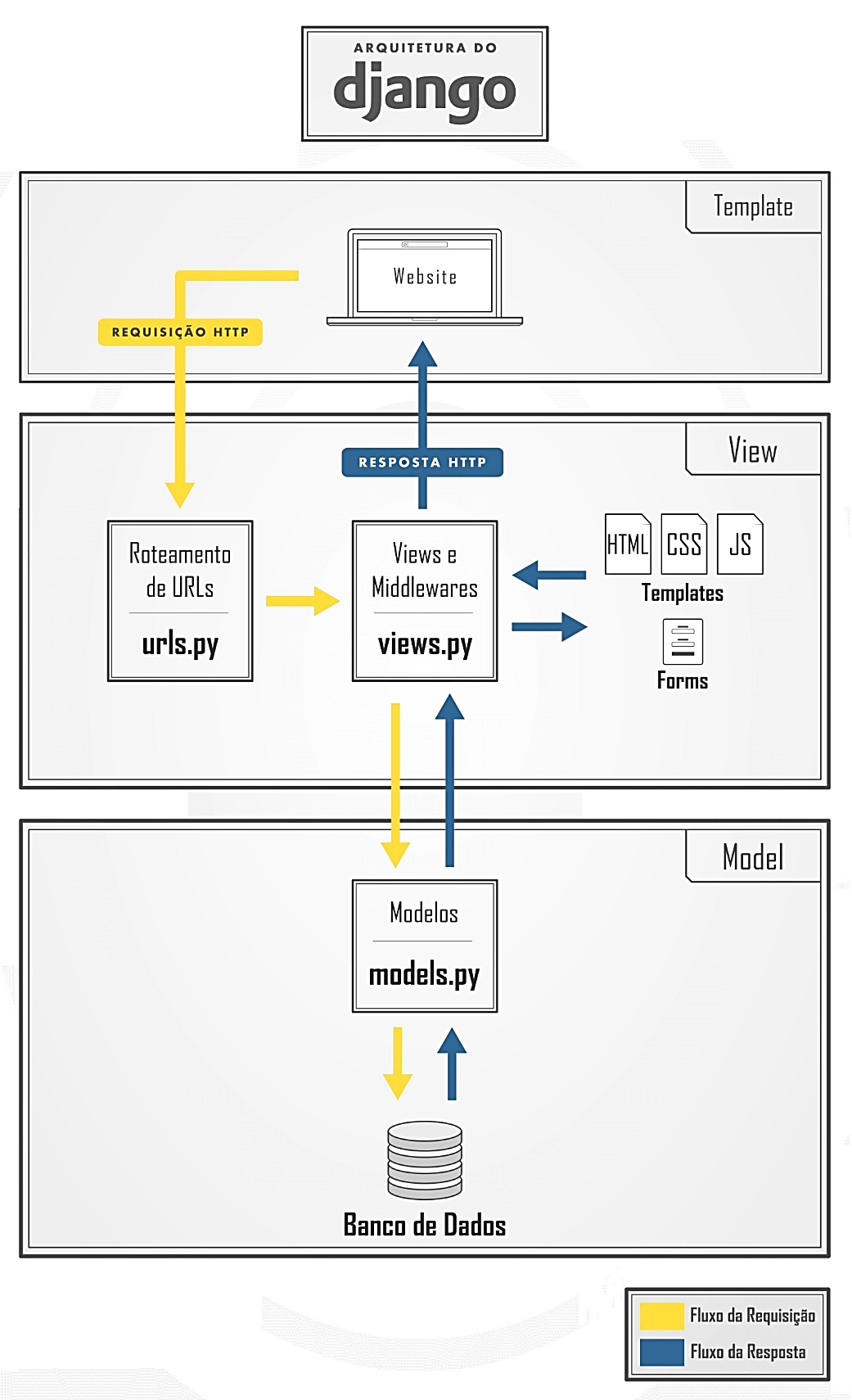


Figura (numerar) Arquitetura do framework Django. Fonte: Python Academy

**Bootstrap**

O ***Bootstrap*** é um *framework front-end* de código aberto, para desenvolvimento rápido com tecnologias *web*. Para utilizar é preciso ter conhecimentos básicos sobre HTML5 e CSS3. Uma das vantagens do *Bootstrap* sobre outros frameworks são os recursos responsivos disponíveis, que permitem o ajuste da solução web a telefones, tablets e desktops.

**Desenvolvimento Web**

**TCP/IP** – TCP (*Trans*mission Control Protocol – Protocolo de Controle de Transmissão) e IP (*Internet Protocol* – Protocolo de Internet) é um conjunto de protocolos de interconexão, de comunicação entre computadores em rede. O conjunto de protocolos TCP/IP é visto como um modelo de camadas ou níveis, onde cada camada é responsável por um bloco de tarefas, retornando um conjunto de serviços para o protocolo da camada superior. As camadas de aplicação, que são as camadas mais altas e estão mais próximas do entendimento do usuário, porque lidam com dados mais abstratos; as camadas mais baixas são deixadas para as tarefas de menor nível de abstração.

O TCP/IP é composto por quatro camadas, onde a primeira é a camada de aplicação. É nesta camada que atuam os protocolos SMTP (*Simple Mail Transport Protocol*) – responsável pela comunicação dos e-mails, o protocolo FTP (*File Transfer Protocol*) – responsável pela transferência e publicação de arquivos na internet e, o protocolo HTTP (*Hypertext Transport Protocol*) – que é o responsável pela publicação de sites na Web e pela navegação na internet. Cada programa utilizado pelo usuário, dependendo da finalidade do programa, comunica-se com um protocolo diferente dentro da camada de aplicação, processando uma requisição.

Após o processamento da requisição na camada de **aplicação**, ela é encaminhada, normalmente, para o TCP, outro protocolo, na camada de **transporte**. A camada de transporte pega os dados enviados pela camada de aplicação, divide os dados em pacotes e os envia para a camada inferior chamada **Internet**. A camada Internet pode receber os dados fora de ordem, e a camada **transporte** é quem coloca os pacotes de dados em ordem, além de checar a consistência dos dados dos pacotes recebidos. É na camada Internet que se situa o IP, responsável por captar os pacotes de dados recebidos da camada transporte e adicionar um endereço virtual, o endereço IP – que são, o endereço do computador que está enviando os dados e o endereço do computador que irá receber os dados. Este pacote de dados é transmitido para a camada Interface de Rede, que é a camada inferior – são chamados de datagramas. A camada interface de rede, após receber os pacotes de dados da camada internet, os envia através da rede. A camada interface de rede também é responsável por receber o pacote da rede, se no momento, o computador estiver recebendo dados. A configuração da camada interface depende do tipo de rede em que o computador está inserido.

**Softwares de controle de versões**

Para cumprir o determinado no tema centralizador: praticando controle de versão, optamos por adotar o sistema de controle de versões ***Git*** para ser empregado no desenvolvimento da solução web proposta para este projeto integrador. O *Git* é um software utilizado para registrar o histórico das edições de qualquer tipo de arquivo, usado principalmente no desenvolvimento e versionamento de software. Cada diretório de trabalho do *Git* é denominado como um repositório que contém o histórico completo das revisões efetuadas. Trata-se de um software livre.

Toda ação de versionamento, dentro do *Git*, ocorre na máquina do usuário. Necessitando, assim, de uma plataforma de hospedagem externa para a interação entre outros programadores. Para essa operação de controle de versão usando o *Git*, é recomendado o uso do ***GitHub***, plataforma de hospedagem de código-fonte e arquivos com controle de versão. O *GitHub* é um software livre, mundialmente usado.

Inicialmente abrimos uma conta no GitHub denominada **Projetointegradorunivesp**, local onde estão sendo criados todos os repositórios referentes ao desenvolvimento das atividades do projeto integrador. Cada participante do grupo foi orientado a abrir uma conta própria, também no GitHub, além de instalar o Git em suas máquinas particulares.

**REFERÊNCIAS**

Banin, Sérgio Luiz. **Python3: Conceitos e aplicações: Uma abordagem didática**. São Paulo: Érica, 2018.

Carvalho, André C. P. L. F. Carvalho e Lorena, Ana Carolina. **Introdução à Computação: hardware, software e dados.** 1.ed. Rio de Janeiro: LTC, 2017

E. Ferreira, Rubem. **Linux: Guia do Administrador do Sistema**. 8.ed. São Paulo: Novatec, 2013.

Flanagan, David. **JavaScript: O guia definitivo**. 6.ed. Porto Alegre: Bookman, 2013.

Matos, Ecivaldo e Zabot, Diego. **Aplicativos com Bootstrap e Angular – Como desenvolver apps responsivos.** São Paulo: Érica, 2020.

Perkovic, Ljubomir. **Introdução à Computação usando Python – Um foco no desenvolvimento de aplicações**. São Paulo: LTC, 2012.

Sommerville, Ian. **Engenharia de Software**. 9.ed. São Paulo: Pearson Prentice Hall, 2011.

William, Stallings. **Arquitetura e organização de computadores.** 8.ed. São Paulo: Pearson Pratice Hall, 2010.

**SITES**

Ramos, Vinícius. **Desenvolvimento Web com Python e Django**. Python Academy: Disponível em <<https://pythonacademy.com.br/blog/> >. [Acessado 16 Set 2021]

**GLOSSÁRIO**

**Framework de aplicações**

Um conjunto de classes abstratas e concretas reusáveis que implementam recursos comuns para muitas aplicações dentro de um domínio (por exemplo, interface de usuário). As classes do *framework*de aplicações são especializadas e instanciadas para criar uma aplicação.