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

Descrevemos a seguir as principais tecnologias utilizadas no desenvolvimento da solução final – o desenvolvimento de um site para facilitar o contato dos adotantes com as ONGs ou comunidades interessadas no processo de adoção de animais, e a hospedagem gratuita do site, em uma plataforma na nuvem. Por tratar-se de uma atividade, quase que filantrópica, optamos por pesquisar e adotar aquelas tecnologias que não gerem custos ou despesas às comunidades que se dispuserem a implementar a solução final apresentada.

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.

**Visual Studio Code**

Visual Studio Code é um IDE – (Integrated Development Environment) ou Ambiente de Desenvolvimento Integrado, é um editor de código-fonte, um software *open source* – livre e de código aberto, de propriedade da Microsoft. Uma das suas principais qualidades é o grande número de extensões disponibilizadas, o que facilita em muito o seu uso nas mais diferentes ações de desenvolvimento front-end. Ele inclui suporte para depuração, controle de versionamento através do Git já incorporado, possui realce de sintaxe e permite a refatoração de código. Todas as linguagens citadas anteriormente fazem parte da enorme quantidade de extensões disponíveis para o Visual Studio Code: HTML5, CSS3, JavaScript e Typescript.

***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*.

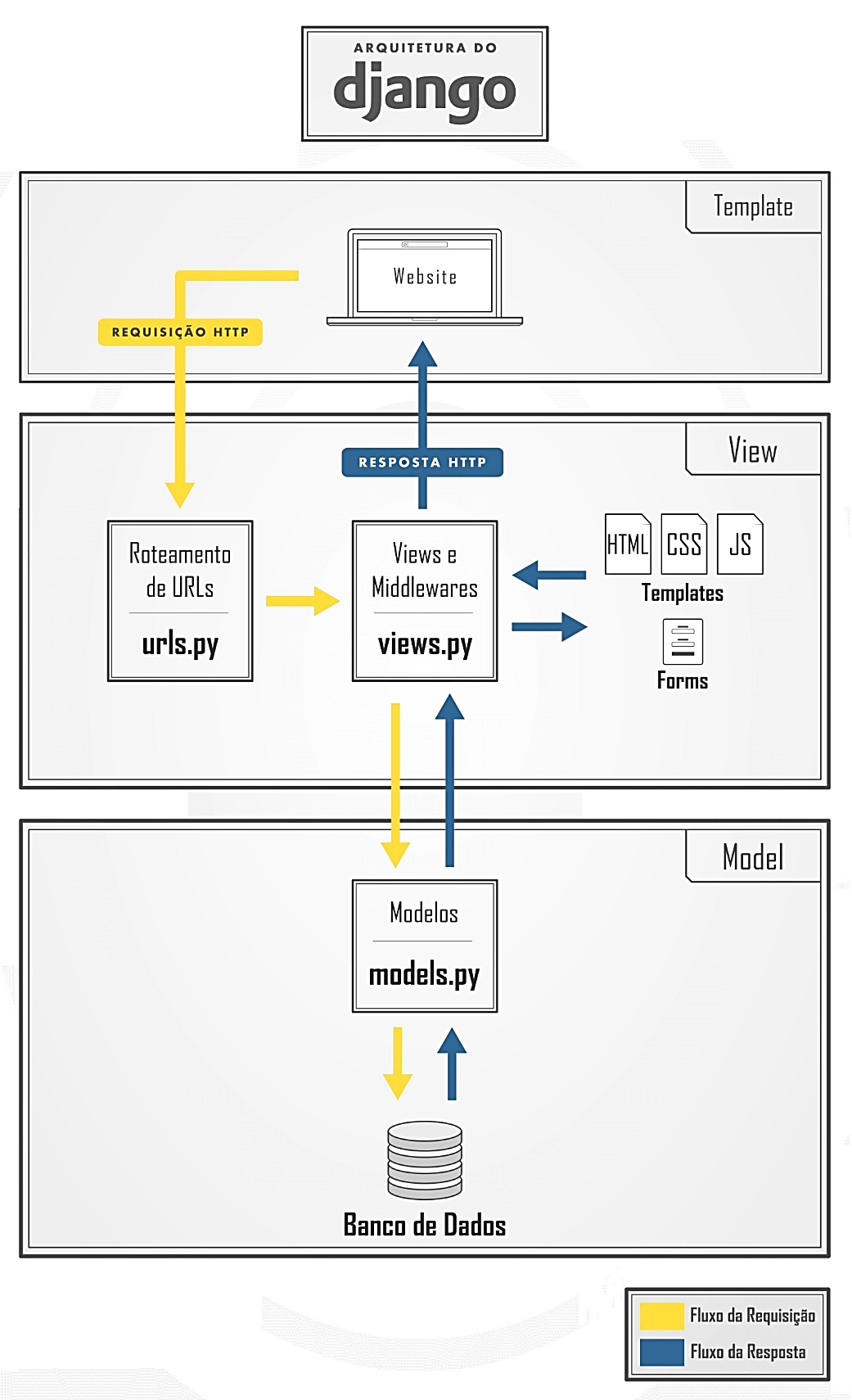


Figura (numerar) Arquitetura do framework Django. Fonte: Python Academy

**Framework para *Python* – *FLASK***

Trata-se de um micro (pequeno) *framework* web, multiplataforma, gratuito, de código aberto, também baseado na linguagem de programação *Python* para desenvolvimento de aplicativos e soluções *web*. *Flask* é caracterizado como um ***micro* *framework*** porque ele não necessita de biblioteca ou ferramentas específicas para o seu correto funcionamento – ele possui um núcleo simples, mas extensível de acordo com a necessidade do desenvolvedor. *Flask* não vem com as camadas de abstração de banco de dados, gerenciamento de usuários, validação de formulários ou outros componentes tais como as bibliotecas pré-existentes. Entretanto, o *Flask* suporta a adição de inúmeras extensões que adicionam recursos as suas funcionalidades, tais como, mapeamento de objeto-relacional, validação de formulários, manipulação de *upload*, tecnologias de autenticação, além de inúmeras ferramentas relacionadas ao *framework*, tornando-o um dispositivo desacoplado.

**Comparação entre os frameworks Flask e Django**

* Os dois *flameworks* utilizam a linguagem de programação *Python*, o que exige do desenvolvedor conhecimento prévio desta linguagem.
* O *Flask* trata-se de um *framework* muito mais leve do que o *framework* *Django* o que o torna indicado para o desenvolvimento de pequenos projetos. Isso não o incapacita para o desenvolvimento de grandes projetos.
* O *Django* é um *framework* completo, indicado para a produção de grandes projetos de desenvolvimento *web*. Isso não o incapacita para o desenvolvimento de pequenos projetos.
* A curva de aprendizagem do *Flask* é mais rápida do que a curva de aprendizagem do *Django*, o que indica a utilizado do *Flask* para quem está começando a aprender a desenvolver soluções *web*.

**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.

**MySQL** – é um **SGBD** – Sistema Gerenciador de Banco de Dados, que emprega **SQL** (*Structure Query Language*) – Linguagem de Consulta Estruturada, como interface. O MySQL é um dos mais populares SGBD em uso atualmente. Foi escolhido como SGBD para utilização no desenvolvimento da solução web proposta no presente projeto integrador, por ter sido conteúdo das aulas da disciplina de Banco de Dados, e por suas principais características:

* É um software livre com base na GPL.
* Portabilidade;
* Excelente desempenho e estabilidade;
* Pouco exigência quanto ao uso de recursos de hardwares;
* Facilidade de manuseio;
* Ser aceito pela plataforma de hospedagem escolhida pelo grupo.

**API** (*Application Programming Interface*) – **Interface de Programação de Aplicação** trata-se de um conjunto de rotinas padrões pré-estabelecidos por um software, para permitir a utilização das suas funcionalidades por aplicativos que pretendem apenas usar seus serviços. **API Web** é um conjunto de interfaces utilizadas no contexto de desenvolvimento web. Trata-se de uma API cujo conjunto de interfaces é definida por mensagens de requisição e resposta HTTP.

O propósito das APIs é simplificar o trabalho de desenvolvimento de programas, por meio da abstração dos inúmeros elementos que fazem parte de um software. O uso de API facilita a implementação de todas as partes que compõe um sistema.

**HEROKU – plataforma de hospedagem em nuvem**

*Heroku* é uma plataforma de hospedagem em nuvem, que suporta várias linguagens de programação. Dentre as linguagens de programação com que a *Heroku* é compatível, está a linguagem *Python*, adotada para a execução da proposta de desenvolvimento *web* apresentada como solução do problema detectado pela comunidade pesquisada.

**Arquitetura do *Heroku*** – Os aplicativos executados na plataforma *Heroku*, possuem um domínio usado para rotear as requisições HTTP para um contêiner de aplicação correta. Cada um desses contêineres é espalhado por vários servidores. A plataforma de versionamento *Git* lida com os repositórios de aplicativos, de forma gratuita, para os usuários cadastrados.

***Deploy* para o *Heroku*** – *Deploy* compreende a fase em que, terminado o desenvolvimento de um aplicativo ou solução web, disponibilizamos o aplicativo, em uma plataforma de hospedagem, para que seja testado ou produção final. A nossa opção foi disponibilizar, fazer o *Deploy*, da nossa solução *web*, na plataforma de hospedagem, *Heroku*, através do *Git*, de forma gratuita.

**Softwares de controle de versões**

***Git*** – Software livre, de Controle de Versão VSC. 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, permite o trabalho em equipe, usado principalmente no desenvolvimento e versionamento de software permitindo a ramificação do projeto, é considerado seguro e organizado. Cada diretório de trabalho do *Git* é denominado como um repositório que contém o histórico completo das revisões efetuadas.

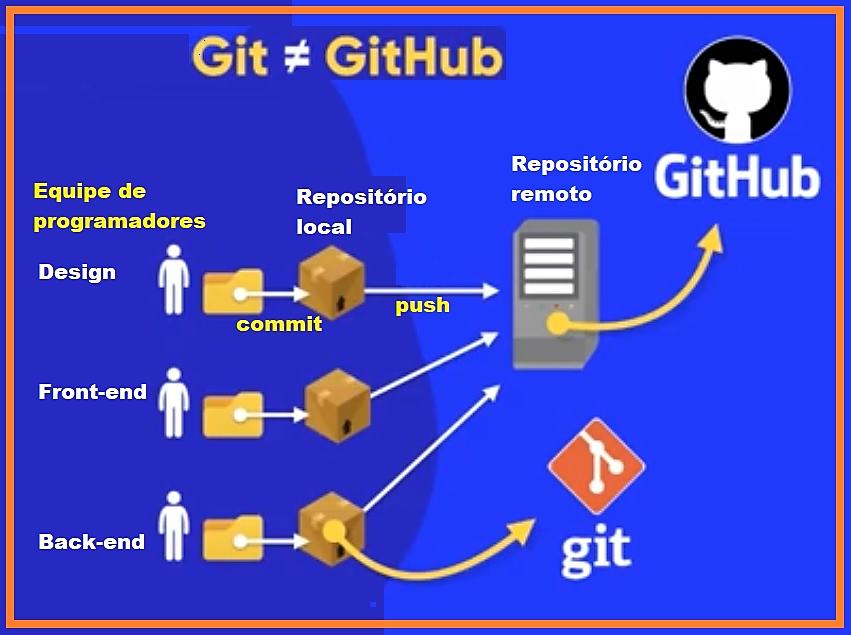


Figura ENUMERAR – Fluxograma comparativo entre os softwares de Controle de Versão Git e a plataforma social de programadores, GitHub, adaptado pelos autores. Fonte: <https://github.com/gustavoguanabara>

***GitHub*** – Software livre utilizado como uma plataforma social para programadores. 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 **Projeto Integrador Univesp**, com endereço: <https://github.com/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 software de controle de versão 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.

Elmasri, Ramez e Navathe, Shamkant B. **Sistemas de banco de dados**. 7.ed. São Paulo: Pearson Education do Brasil, 2018.

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

Grinberb, Miguel. **Flask Web Development – Developing Web Applications with Python**. Estados Unidos: O’Reilly, 2014.

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**

**MySQL Workbench Reference Manual** – **for versions 8.0 through** **8.0.26**. Disponível em: <https://www.mysql.com/support/supportedplataforms/workbench.html> Acesso em 16 Set 2021.

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

**Sites das tecnologias utilizadas no desenvolvimento da solução do Projeto Integrador**

**Bootstrap** <https://getbootstrap.com> Acesso em 21 set. 21.

**Django**. <https://djangoproject.com> Acesso em 21 set. 21.

**Flask**. <https://flask.palletsprojects.com> Acesso em 21 set. 21.

**Git**. <https://git-scm.com> Acesso em 12 set. 21.

**GitHub**. <https://github.com> Acesso em 12 set 21.

**Heroku**. <https://www.heroku.com> Acesso em 21 set. 21.

**MySQL**. <https://www.musql.com> Acesso em 21 set. 21.

**Python**. <https://www.python.org/> Acesso em 20 set. 21.

**Visual Studio Code**. <https://code.visualstudio.com/> Acesso em 20 set. 21.

**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.