**Yin y Yang de una aplicación: frontend y backend:**

Si imaginamos un auto, este va a tener muchos componentes que en conjunto forman el auto, pero principalmente existen dos, el primero es la carrocería, que es la parte visible del auto, y el otro componente importante es el motor, que es lo que el auto funcione.

Si aplicamos lo mismo en aplicaciones web, tienen una cara visible, o sea la interfaz, y tienen un motor, o sea una lógica que hay detrás, en programación la cara visible es el frontend, y el motor es el backend, juntos harán la aplicación.

Python es un lenguaje que sirve para construir backend, pero para construir el frontend existen otras tecnologías, una que compone el esqueleto de la palicacion (HTML), una para la disposición de los elementos (CSS) y una para la lógica que permite interactuar con la aplicación (JavaScript). Cada uno de los anteriores lenguajes tiene sus derivados, y estos se van a llamar librerías y frameworks.

Antes de empezar a hacer la cara de la aplicación web, es necesario hacer una etapa de diseño, ya que no es buena idea empezar a programar de 0, así que se hace antes un diseño de la aplicación:

* UI design: se crea la interfaz, la forma y el diseño de los botones
* UX design: donde debería estar ubicado el botón para que sea fácil de usar

En backend se pueden utilizar la gran mayoría de lenguajes que existen, incluso JavaScript, al igual que antes cada lenguaje tiene sus propios framework y librerias, por ejemplo JavaScript tiene un motor para correr JavaScript del lado del servidor llamado Node.JS

Diagrama

Descripción generada automáticamente

Python tiene tres librerías y frameworks para el desarrollo backend: Fast API, Flask y Django. Los tres tienen el mismo fin, construir el backend de una aplicación.

**Framework vs. Librería:**

En programación no se puede hacer todo desde 0, así que no hay que reinventar la rueda, para eso se crean las librerías, que es un código escrito por otra persona que hace el trabajo sin necesidad e saber como funciona, pero para poder utilizar las librerías, y en general para poder tener una marco de trabajo que se denominara framework, que cosa necesito específicamente para programar x cosa, o sea un framework son un conjunto de librerías, pasos, reglas y recetas para poder construir un producto digital.

En Python existen tres framework para construir una aplicación web: Fast API, Flask y Django.

**Cómo se conecta el frontend con el backend: API y JSON:**

Texto

Descripción generada automáticamenteCuando se está creando una aplicación web, hay que unir el frontend y el backend de alguna manera, esta unión se hace con una API (Application programming interface o interface de programación de aplicaciones) y es simplemente una sección del backend que permite que el frontend se comunique con el backend, y que pueda haber mensajes de ida y de vuelta.

Existen dos grandes estándares: SOAP (Simple Object Access Protocol) y REST (Representational State Transfer), el primero mueve la información mediante XML (extensible markup lenguage) que es similar a HTML, pero SOAP está quedando obsoleto porque se está utilizando más REST, este utiliza JSON (JavaScript Object Notation) que es un lenguaje que sirve para comunicarse entre backend y frontend, básicamente un JSON es un diccionario de Python, se ve de la misma manera, y los diccionarios en Python son lo mismo que los objetos en JavaScript, de ahí el nombre, simplemente cuando no depende del lenguaje se denomina JSON.

**El lenguaje que habla Internet: HTTP**

Hypertext transfer protocol o protocolo de transferencia de hipertexto. En el internet siempre hay clientes y servidores que se comunican, los clientes son los dispositivos que se conectan, y el servidor es una computadora que siempre esta prendida y contiene la aplicación web.

El cliente hace una petición a un servidos que traiga la aplicación web, el servidor envía una respuesta y el cliente la ve en el navegador, la transferencia de información se hace con una petición del cliente y una respuesta del servidor (request y response),

Un protocolo es un conjunto de reglas para comunicar dos partes, la computadora tiene que hablar el idioma del servidor para hacer la petición, y ese idioma es HTTP. El protocolo http se compone de cabeceras y el método, existen muchos métodos como el get, post, put, parch, la respuesta del servidor tiene además un cuerpo, que es un json, y sirve para tener los datos que necesitamos.

La primera línea de la cabecera de la respuesta es un código de estado, que dice como salió esa respuesta:

* 200 salió bien
* 400 algún recurso no fue encontrado
* 500 hay un problema en el código

Texto

Descripción generada automáticamente

El protocolo no funciona por sí mismo, sino que esta dentro de una escala jerarquica de protocolos.

Gráfico, Gráfico de rectángulos

Descripción generada automáticamente

**¿Cómo es el flujo de desarrollo de una aplicación web?:**

Lo primero que se necesita es el editor de código como Visual Studio Code o Atom, además hay que tener un sistema control de versiones como git, y en un proyecto web es necesario un navegador, que sirve para sebar si las cosas están saliendo bien.

La aplicación web tiene que compartirse, por lo que se coloca en el servidor para que pueda ser vista por cualquier persona, entonces hay un entorno local donde se trabaja, y un entorno global que es el servidor (produccion). Deeploit es el proceso de llevar el código de la computadora al servidor, y lo recomendable es primero hacer push a un repositorio remoto como github, que también es un servidor pero que contiene el código del proyecto, y sirve para que ese código pueda ser compartido a otras personas, pero cuando uno trae código de GitHub se denomina pull.

Cuando se trabaja en github se hace CI/CD (continuous interation/delivery ) y lo que hace es tomar el código escrito en el entorno local y lo prueba para ver si funciona bien, y finalmente se envía al servidor.

(La nube por detrás es un servidor) El servidor va a guardar la aplicación en un dominio, (<http://myapp.com>), los dominios hay que comprarlos como en <https://www.namecheap.com/> pero también hay una seri de dominios gratis que se pueden usar.

En la computadora las cosas viven en el entorno local, y esta formado por una dirección ip y un puerto, la combinación de ambas es el local host, lo mas normal es ponerle el puerto 8000, y hay muchas aplicaciones que se trabajan ahí, no importa cual se ponga siempre que esté disponible.

**El hogar de tu código: el servidor:**

Un servidor es una computadora que contiene una aplicación y la distribuye por internet, además por medio de HTTP se le puede pedir la aplicación y traerla al navegador, cuando se habla de la nube se habla de servidores.

Un DataCenter es un espacio gigante donde se tiene un conjunto de servidores trabajando en paralelo y contienen datos que corresponden a muchas cosas, como los datos de Google, estos DataCenters existen en todo el mundo. Cuando se publica una aplicación, hay muchas maneras de llevar la aplicación a producción, el local al servidor.

Hosting es cuando se guarda una aplicación en un servidor, un espacio en un servidor donde la aplicación va a ser guardada, cada vez que se vaya a publicar la aplicación hay que escoger una de las siguientes:

* Iaas (infrastructure as a service): control sobre las cosas mas importante del servidor donde va a vivir la aplicación: CPU, RAM, SSD
  + Amazon Web Services
  + Micorsoft Azure
  + Digital Ocean
* Paas (platform as a service): el servidor se encarga de actualizar todas las aplicaciones que hacen que viva la aplicación: bases de datos, firewall. Permite escoger que necesita la aplicación para funcionar. (just deploy)
  + Google app engine
  + Firebase
  + Heroku
* Saas (software as a service): Una aplicación que un proveedor te presta para hacer funcionar el negocio. No hay que hacer código ni programar.
  + Google docs
  + Slack
  + Word Press

Hay dos tipos de Iaas, Share hosting es alojamiento compartido, o sea si se escogen los recurso pero hay que compartirlo con otras personas, además es mas barato. VPS da un mejor rendimiento, ya que es un servidor privado para la aplicación.

**Proyecto: diseño y bosquejo de una API**

Si por ejemplo queremos una API para un tweet de twietter, hay que crear y borrar un usuario, crear, modificarlo, y borrar tweet, además tiene que poderse ver los usuarios y los tweets, o sea en esencia se necesita CRUD (créate, read, update, delete), cada ves que se diseña una aplicación se necesita un crud que funciona por debajo, y se puede traer a la vida median una API.

Hay mucho frameworks y librerías que permite crear API, en Python esta fastapi, djando con rest framewor, y flask.

un endpoint /route/path es una sección de la url del proyecto, una url se conforma del protocolo, el nombre de dominio y el endpoint, y se podría crear un endpoint que traiga los tweets, pero lo más común es que haya antes un /api.

* <http://twitter.com/api/tweets>

**Proyecto: diseñando los endpoints de los Tweets:**

Para crear los endpoint hay que seguir las reglas que proporciona CRUD, todas las palabras que hay al final del dominio que permite acceder a los datos que se necesite.

* /tweets 🡪 Show all tweets
* /post 🡪 Publish a tweet
* /tweets/{tweet = id} (un numero único que corresponda a un tweet o parametro) 🡪 Show a tweet
* /tweets/{tweet = id}/update 🡪 Update a tweet
* / tweets/{tweet = id}/delete 🡪 Delete a tweet

**Proyecto: diseñando los endpoints para los usuarios:**

Tweets y user se denominan modelos, tipos de datos en particular que voy a manejar en la aplicación, en este caso usuario con diferentes características, y tweets. En SQL los modelos se denominan tablas, y tiene registros dentro de SQL, pero cuando se esta programando esos registros se les denomina atributos.

* /users 🡪 show all users
* /signup 🡪 register a user
* /users/{user = id}/ 🡪 show a user
* /users/{user = id}/update 🡪 Update a user
* /users/{user = id}/delete 🡪 Delete a user

Cada vez que entremos a un endpoint, el servidor contesta con un archivo en formato JSON, y este contendrá los datos que necesitamos, en este caso la información del usuario,

**Qué lenguaje y framework escoger para backend:**

En python

* Django tiene un panel de administración de datos muy completo, y una comunidad enorme.
* Flask sive para trabajar con aplicaciones simples y sencillas, se usa siempre en cosas personalizables
* Fast api es el freamework más rápido de Python, es más rápido que la mayoría de frameworks

En go: un lenguaje muy rápido

Diagrama, Logotipo, nombre de la empresa

Descripción generada automáticamente