Introducción al Desarrollo Backend

lunes, 8 de agosto de 2022 21:41

FRONT Y BACK
Front: Carrocería, parte externa
{
Tecnologías principales: [HTML : Esqueleto, estructura
CSS: Estilos, colores, dispociones
JS: Lógica de interacción con la aplicación]
}
Back: Motor, corazón, lógica de la aplicación.

Antes de tirar código se hace un diseño de la aplicación
Ahí es donde entra el diseño UX que es donde se crea la interfaz, los botones y se establecen los colores
UI donde nos fijamos en el posicionamiento de los objetos para la usabilidad de la aplicación
usando tecnologías como [Adobe XD, Sketch, Figma]

FRAMEWORK VS LIBRERIA
Librería: Código escrito por otra persona que dará ciertas funcionalidades a mi software

Framework: Conjunto de librerías, reglas y estándares para el desarrollo de un producto digital

CONECTANDO FRONT CON BACK
El front y el back se unen mediante una sección del back llamada API(Application Program Interface) la cual habilita que hayan mensajes de ida y vuelta

Estándares para la construcción de APIS
SOAP (Simple Acces Object Protocol) mueve la info entre front y back mediante el lenguaje XML

REST (Representational State Transfer) mueve la info entre front y back mediante el lenguaje JSON

HTTP: Hyper**T**ext **T**ransfer **P**rotocol
Protocolo: Conjunto de reglas que sirven para comunicar a 2 partes.
Se transfiere *hypertexto* al transferir además de texto plano, links, enlaces, videos y demás.

Servidor: Contiene la aplicación
Cliente: Lanza una petición al servidor, le pide al servidor que traiga la aplicación

El Cliente lanza una petición (request) en HTTP, el servidor da respuesta (response) a la petición en HTTP; contiene un body, en caso de ser un API el body es JSON.

Métodos HTTP: Get (Traer la información), POST, PUT, DELETE, PATCH
Status code: En HTTP es el código de como salió la respuesta.

IP (Internet Protocol) Protocolo base con el que funciona internet

Sirven para transmitir datos a traves de la IP
{
TCP (Transmision Control Protocol)
UDP (User Data Protocol)
}
TLS (Transport Layer Security) Los datos transferidos por medio de la IP viajen de manera segura.
DNS (Domain Name Server) Convertir la IP en un dominio.

El ordenador hace una petición HTTP al servidor para que le devuelva el HTML y el navegador lo interprete y lo muestre.

FLUJO DE DESARROLLO DE UNA APLICACION WEB

Al ***servidor*** se le conoce popularmente como ***produccion / production***
Deploy es el proceso de llevar el codigo de local (tu pc) a un repositorio y del repo al servidor.

CI / CD = *Continuos Integration / Continuos Delivery*: Testear el código escrito en local y probar que todo esté funcionando bien, si todo está bien el código se va al servidor y se hace el deploy.

En local en vez de tener un dominio, va a tener una dirección IP junto con un puerto
127.0.0.1: 8000
-----IP---- -Port-

EL SERVIDOR
Una computadora que contiene una aplicación y la distribuye en internet.
Hay multiples formas de hacer deploy de una aplicación.
Cloud Computing: Como configurar y trabajar con los servidores para que las aplicaciones trabajen de la forma más eficiente posible.

Hosting es guardar una aplicación en un servidor.

IaaS: Infraestructure as a service, Cada vez que se quiera tener control sobre las cosas más importantes del servidor, tener control sobre CPU, RAM, SSD, todo el hardware del servidor.
2 tipos {
Share Hosting: Compartir el espacio / recursos del servidor con otras personas que también tienen su app.
VPS (Virtual Private Server) Servidor privado.
}
[AWS, Azure, Digital Ocean]

PaaS: Platform as a service, Se encarga de actualizar todas las aplicaciones que hacen que viva la aplicación, el firewall, bases de datos y demas. Se va a tener una interfaz gráfica en donde se coloca que es lo que necesita la aplicación para funcionar. Solo hacer el deploy del codigo.
[Google App Engine, Firebase, Heroku]

SaaS: Software as a service, Se necesita un software ya creado por otra persona, una aplicación ya creada; aplicación que un proveedor da.
[Google Docs, Slack]

DISEÑO DE UNA API
Endpoint / Rout / Path: Seccion de la URL del proyecto

HTTP:// Twitter.com / Api /Tweets
Protocolo Nombre dominio EndPoint

DESARROLLANDO LOS ENDPOINTS DE CRUD
Modelos / Models: Tipos de datos en particular que se van a manejar en la aplicación.
Los modelos se conocen como tablas en SQL.
A los registros se les denomina atributos.

Tweets EndPoints
{
Read: /Tweets; mostrar todos los tweets
Post: /Post; publicar un tweet
Read: /Tweets/{tweet_id}; Mostrar un tweet
Update: /Tweets/{tweet_id}/Update; Actualizar un tweet
Delete: /Tweets/{tweet_id}/Delete; Borrar un tweet
}

Users EndPoints
{
Read: /Users; mostrar todos los usuarios
Post: /SignUp; registrar un usuario
Read: /Users/{user_id}; Mostrar un usuario
Update: /Users/{user_id}/Update; Actualizar un usuario
Delete: /Users/{user_id}/Delete; Borrar un usuario
}

Cada vez que se entre a un endpoint el servidor va a contestar con un archivo JSON.

QUE LENGUAJE Y FRAMEWORK ESCOGER
Para desarrollar Back de aplicaciones

Python
{
DJango : Si se va a trabajar con muchos datos DJango es una buena opcion.
Flask : Sirve para trabajar con aplicaciones simples y flexibles, algo personalizable y a la medida de las necesidades.
FastApi : API más rápidas que hay.
}

JS
{
Express : Simple, fácil de escribir.
Nest : Mayor complejidad, más ventajas; un express con esteroides.
}

PHP
{
Laravel : Mas fácil, sintaxis más sencilla.
Symfony : Mas complejo, aplicaciones más escalables y complejas.
}

Java
{
Spring : Back de aplicaciones web.
}

GO
{
Gin : Framework por excelencia de GO
Beego : Framework en crecimiento
}

Ruby
{
Ruby on Rails : Rapido y flexible
}