Curso de Introducción al Desarrollo Backend

Por: Facundo García Martoni

Platzi 2021

Autor: REM - Notas del Curso

1. Lo que se aprenderá

- Bases para ser desarrollador Backend
- Diferencia entre Framework y libreria
- Protocolos HTTPS
- Servidores y tipos de servidor
- Flujos de trabajo
- API y JSON

- Desarrolo de una API

Requisitos

El requisito recomendado es el Curso Básico de Python.

2. Yin y Yang de una aplicación: frontend y backend

Las aplicaciones tienen una interfaz y una lógica que hace que funcione. La interfaz se le llama Frontend y la lógica es el Backend.

El Frontend y el Backend se pueden construir con diversas tecnologías.

Frontend

Las teconologías usadas para el Frontend:

- **HTML**, para la creación de la estructura
- **CSS**, para dar estilos
- **JS**, para agregar funcionalidad e interacción

Librerias y Frameworks

Para CSS tenemos:

- Foundation
- Bootstrap
- Tailwind

Que nos ayudan a escribir código de manera mas rápida y sencilla.

Para JAVASCRIPT tenemos:

- REACT.JS
- ANGULAR
- SVELTE
- VUE

Nos permitén escribir código de manera fácil, rápida y segura para tener menos errores en el frontend de una aplicación.

Antes de empezar a crear una aplicación con estas tecnológias hay una etapa de diseño, sin esta etapa obtendríamos una aplicación poco estética.

Diseño de una aplicación

A esta tarea se dedican los diseñadores donde encontramos dos ramas importantes **UI DESIGN** y **UX DESIGN**, que se complementan.

Para UI Design tenemos algunas aplicaciones como:

- Adobe XD
- Sketch
- Figma

Backend

Para el backend tenemos gran cantidad de lenguajes:

- JavaScript, con Node.JS
- PHP, con Laravel
- Java, con Spring
- Go
- Rust
- Ruby, con Ruby On Rails
- Python, con Fast API, Flask y DJango

3. Framework y Librería

Aunque parezcan lo mismo, no lo son.

Framework

Es un conjunto de herramientas (librerías, reglas, estandares) que trabajan en un proyecto complejo bajo ciertas reglas.

- Tiene funcionalidades integradas para evitar el uso de librerías externas.
- La compatibilidad de sus funcionalidades esta asegurada.
- El *Framework define* la forma en que debes desarrollar el proyecto.

Librería

Es una herramienta con una sola utilidad específica.

- El desarrollador es libre de usar la librería en la estructura que desee.
- Se debe controlar la compatibilidad de una librería con las demás.
- Se puede usar varias librerías según las necesidades.

4. Conectando Frontend y Backend: API y JSON

El frontend y el backend se unen mediante un componente especial llamado **API**, cuyas siglas provienen del inglés Application Program Interface.

Existen dos estandares para las API's:

- SOAP: Simple Object Access Protocol

  Utiliza Extensible Markup Language o XML para enviar la información. Ha caido en desuso debido a que existe una mejor tecnología, muy superior y mas eficiente llamada REST.

- REST: Representational State Transfer

  Utiliza JavaScript Object Notation o JSON para el intercambio de información entre el frontend y backend. Los diccionarios de Python y los Objetos de JavaScrips son similares.

5. HTTP, el lenguaje qeu habla Internet

HTTP, del inglés HyperText Transfer Protocol.

Cliente, los dipositivos como tablet, smarthphone, etc. Hacen peticiones al servidor.

Servidor, un dispositivo que está disponible todo el tiempo, envía una respuesta cuando el cliente lo solicita.

Las peticiones tienen 3 variables o cabeceras.

- HOST: Servidor al que se hace la petición.
- Accept-language, indica el lenguaje.
- Método: Que puede ser POST, GET, PUT, DELETE, TRACE, etc.

```
# Request
GET / HTTP/1.1
Host: developer.mozilla.org
Accept-Language: fr
```

El servidor responderá tambien con un 'lenguaje HTTP', el cual tiene diversos componentes.

```
# Response
HTTP/1.1 200 ok
Date: Sat, 09 Oct 2010 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
ETag: "51142bc1-7449-479b075b2891b"
Accept-Ranges: bytes
Content-Length: 29769
Content-Type: text/html
<!DOCTYPE html...(here comes the 20769 bytes of the requested web page)
```

La respuesta tiene encabezados y un body que contiene la respuesta del servidor con los datos solicitados.

HTTP/1.1 es la versión HTTP, que es la misma que la petición.

200 ok: Es el **estado de la petición**, en este caso la respuesta fue exitosa. Existen diferentes estados de respuesta HTTP.

- 100 - 199: Respuestas informativas
- 200 - 299: Respuestas satisfactorias
- 300 - 399: Redirecciones
- 400 - 499: Errores de los clientes
- 500 - 599: Errores de los servidores

Tipicos errores que vemos al momento de hacer peticiones:

404 Not found: El servidor no pudo encontrar el contenido solicitado. Es el mas famoso y de alta ocurrencia en la web.

403 Forbidden: Cuando el cliente no posee los permisos para cierto contenido, por lo que el servidor rechaza otorgar la respuesta apropiada.

500 Internal Server Error: El servidor ha encontrado una situación que no sabe como manejarla.

502 Bad Gateway: Significa que el servidor, mientras trabaja como una puerta de enlace para obtener una respuesta necesaria para manejar la petición, obtuvo una respuesta inválida.

El protocolo HTTP se ubica dentro de una escala gerarquica de los protocolos que tenemos en la web.

A grandes rasgos se ubicaría entre los protocolos:

(IP, UDP, DNS, TCP, TLS) -> (HTTP) -> (HTML, CSS, Web API, JavaScript)

Donde:

- TCP: Transmission Control Protocol
- TLS: Transfer Layer Security
- DNS: Domain Name System
- IP: Internet Protocol
- UDP: User Data Protocol

> **RESUMEN**:
>
> El **Server** envía el html, css y javascript al **Client** para que el mismo mediante su navegador pueda representar la app web, a su ve el **Client** envía un *request http* al **Server** para que el mismo mediante su **API** pueda constestar (**response**) con los datos en formato **JSON** para que el **Client** pueda aprovecharlos.

6. Flujo de desarrollo de una Aplicación Web

Editor de Código: Es la aplicación que nos permite editar nuestro código, puede ser un editor de texto plano, sin embargo hay aplicaciónes específicas para el desarrollo. Un editor conocido y recomendado es *Visual Studio Code* .

Sistema de Control de Versiones: Nos permite controlar los cambios y la historia de el proyecto que estamos desarrollando, así podemos movernos por los diferentes puntos historicos de la aplicación, crear ramas de prueba, crear ramas para resolver errores, etc.

Browser: En desarrollo web el browser es el navegador, que nos permite ejecutar la aplicación, ver su apariencia y comportamiento. Entre los navegadores mas conocidos tenemos: Google Chrome, Mozilla Firefox, Edge, Opera, etc.

Servidor o Server: Es una computadora que contiene la aplicación o sirve a los dispositivos clientes.

Deploy: Es el proceso de colocar el código local hacia el servidor. El deploy no necesariamente se hace de manera directa, sino que primero nuestro código pasa por un repositorio remoto como GitHub.

PUSH: Es el proceso de pasar nuestro código hacia GitHub.

PULL: Es el proceso de traer los cambios desde GitHub hasta nuestro repositorio local (en nuestra computadora)

CI/CD: Continuous Integration y Continuous Delivery, toma el código del local y realiza pruebas, si todo funciona bien el código va al servidor, completándose así el Deployment.

Production: El código que está en el servidor y esta ejecutándose. La aplicación esta almacenada en un servidos y que tiene una dirección IP el cual está asociado a un Dominio.

Local Host: Es el servidor local, se usa para el desarrollo en nuestra computadora. Generalmente el servidor local funciona en un IP local 127.0.0.1 y el puerto 8000.

7. El servidor: Hogar del ćodigo

Un servidor es una computadora que contiene una aplicación y la distribuye en internet. El cliente a través del protocolo HTTP puede hacerle peticiones y obtener datos para ser representados en el navegador.

Cuando se habla de la ***Nube*** , se refiere a un conjunto de servidores ubicados en algún lugar del mundo y que almacenan aplicaciones web. A este conjunto de servidores ubicados en algún lugar se le denomina **Data Centers**, desde aquí envian datos a todo en todo el mundo para los diferentes clientes.

Hosting: Un espacio en un servidor donde podemos guardar una aplicación.

Los servidores o hosting pueden darse de diferentes formas, y vamos a tener en cuenta al momento de elegir un servicio de Hosting: IaaS, PaaS y SaaS.

IaaS: Infrestruture as a Service

Nos permite controlar la infraestructura del Servidor como CPU, RAM y tipo de disco SSD. Se debe configurar el servidor desde cero.

Las mas populares IaaS:

- Amazon Web Services
- Microsoft Azure
- Digital Ocean

Tipos de IaaS:

- VPS: Virtual Private Server, donde el usuario tien mas control del servidor
- Shared Hosting: El usuario debe compartir el servidor con otros clientes.

PaaS: Platform as a Service

El servidor se encarga de actualizar todas las aplicaciones que mantienen el servidor. Solo se debe elegir que es lo que necesita la aplicación para funcionar: Bases de datos, Firewall, etc.

Los mas populares tenemos:

- Google App Engine
- Firebase
- Heroku

SaaS: Software as a Service

Ofrece software o una aplicación que funcione y ya exista para hacer funcionar un negocio. Esta diseñado para usuarios que recien empiezan.

Los más pupulares:

- Google Docs
- Slack
- WordPress

8. Proyecto: Diseño y bosquejo de una API

Creación de una API simple

Twitter, es una aplicación de micro blogging. Cada usuario escribe un twit y los seguidores pueden ver el mensaje.

Si fueramos a crear una aplicación similar o un clon de twitter, nuestra aplicaición tendría que ser capaz de crear: *Usuarios* , *Mensajes* , *Actualizar usuarios* . Así mismo también tendría que ser capaz de borrar usuarios y mensajes.

Los términos usados: Crear, Leer, Actualizar y Borrar, a estos verbos usados se le denomina como **CRUD** (create, read, update and delete)

A este CRUD se le puede traer a la vida mediante un API, que viene a ser el motor de la aplicación. De esta manera conectaremos el backend y el frontend mediante la API y nuestra aplicación estará funcionando.

Creación de Twitter API

API: Application Program Interface

Librerías para API en Python:

- FastAPI
- Django -> REST Framework
- Flask

EndPoint o Route o Path, es una sección de la URL de nuestro proyecto.

La URL del proyecto generalmente es: **http://twitter.com/API/tweets**

Componentes:

- **http://**, es el protocolo
- **twitter.com**, es el dominio
- **/tweets**, es el ***EndPoint o Route o Path*** . Viene despues del API

9. Proyecto: Diseñando EndPoints de los Tweets

Vamos a crear los Endpoints de los Tweets (models):

- /tweets -> Show all tweets
- /post -> publish a tweet
- /tweets/{tweet-id} -> show a tweet
- /tweets/{tweet-id}/update -> update a tweet
- /tweets/{tweet-id}/delete -> delete a tweet

> Nota:
>
> Cada sigla del CRUD tiene una representación en el mundo de las APIs y se llaman **verbos http**, que indican en la petición al servidor lo que se quiere hacer:
>
> - Create -> POST
> - Read -> GET
> - Update -> PUT
> - Delete -> DELETE
>
> De esta manera los endpoints quedarían:

> - Crate a tweet: */tweets/ :POST*
> - Read all tweets: */tweets/ :GET*
> - Read a tweet: */tweets/{id} :GET*
> - Update a tweet: */tweets/{id} :PUT*
> - Delete a tweet: */tweets/{id} :DELETE*

10. Proyecto: Diseñando los Endpoints para los usuarios

Vamos a crear los Endpoints para los usuarios (models):

- **/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 al endpoint, el servidor responderá con un archivo que contiene los datos, este archivo estará en formato JSON:

```
{
    "user-id": 121
    "username": facMartoni
    "email": facmartoni@mail.com
}
```

11. Que lenguaje y framework escoger para backend

Frameworks Python:

- **Django**, es el mas robusto. Tiene panel de administración completo para trabajar con muchos datos. Documentación de 3 mil páginas.
- **Flask**, es facil de aprender y para aplicaciones simples.
- **Fast API**, es el framework mas rápido y potente, mucho mas rápido de Django y Flask.

Frameworks JavaScript:

- **Express**, simple y facil de escribir
- **Nest**, esta escrito con un nivel de complejidad mayor con mas ventajas y codigo aprovechable.

Frameworks PHP:

- **Laravel**, mas simple
- **Synfony**, para apps mas complejas y mas escalables

Frameworks Java:

- **Spring**, sirve para backend de aplicaciones web

Frameworks Go:

- **Gin**
- **Beego**

Frameworks Ruby:

- **Ruby on rails**

12. Empieza tu camino

Lo que se aprendió:

- [x] HTTP
- [x] API
- [x] JSON
- [x] SERVIDORES
- [x] LA NUBE