Introducción al Desarrollo de Software I - Lanzillotta

Capra - Chaves - Di Matteo - Sosa - Villegas - Palavecino

.UBAfiuba
FACULTAD DE INGENIERÍA

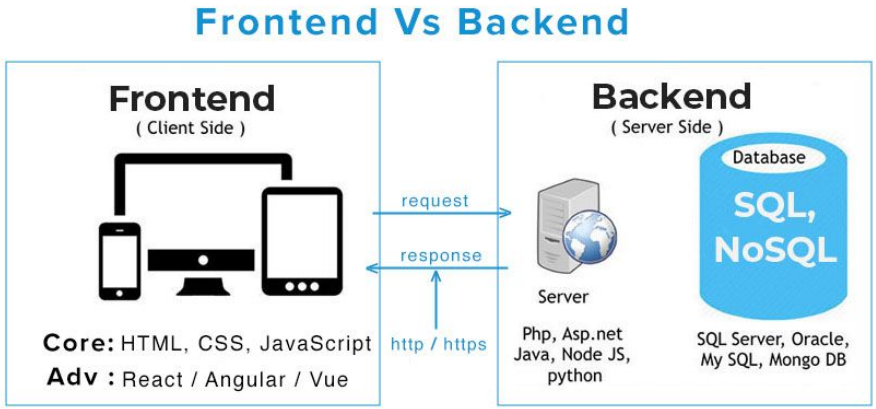
Agenda

- Introducción al desarrollo web
- Introducción al backend
- Introducción al frontend
- Ejemplo

Introducción al Desarrollo Web

Desarrollo Web

El desarrollo web es el **proceso de crear y mantener sitios y aplicaciones web**. Abarca una amplia gama de tareas, desde el diseño y la planificación hasta la codificación y la implementación. El desarrollo Web se suele separar entre **Front end** y **Back end**



Introducción al backend

¿Qué es el Backend?

El **back end (o backend)**, también conocido como desarrollo del lado del servidor, se refiere a la parte de un sitio web o aplicación web que no es visible para los usuarios. Se encarga de la lógica de la aplicación, el almacenamiento de datos y la comunicación con el servidor. Entre sus tareas se encuentran:

- Implementar la lógica de la aplicación.
- Almacenar y recuperar datos de una base de datos.
- Procesar solicitudes y enviar respuestas al usuario.
- Asegurar la seguridad del sitio web.

Frameworks de Back End

Existen muchas herramientas para armar el back end de una aplicación o sitio web, dependiendo de la tecnología que se esté utilizando.

Flask

Flask es un **framework web** para **Python**. Es minimalista y se enfoca en proporcionar las herramientas básicas para construir aplicaciones web. En lugar de incluir funcionalidades predefinidas como manejo de base de datos o validación de formularios, Flask permite a los desarrolladores elegir las librerías y herramientas adicionales que necesite para su proyecto.

Más adelante veremos cómo utilizar de manera básica ciertas funcionalidades de Flask y en las siguientes clases profundizaremos sobre aspectos del desarrollo back end.

Introducción al Front end

¿Qué es el Front end?

El **front end (o frontend)**, también conocido como desarrollo del lado del cliente, se refiere a la parte de un sitio web o aplicación web con la que los usuarios interactúan directamente. Es la cara visible del sitio, y se encarga de la **interfaz de usuario (UI)** y la **experiencia del usuario (UX)**. Entre sus tareas se encuentran:

- Diseñar la estructura y el layout de la página web.
- Implementar la lógica de la interfaz de usuario.
- Agregar elementos interactivos como botones, menús y formularios.
- Optimizar el rendimiento del sitio web para diferentes dispositivos.

Lenguajes del Front end

Para el desarrollo del front end de una aplicación se suele utilizar un **stack** básico de desarrollo, y éste cuenta de las siguientes 3 tecnologías:

HTML

HTML *(HyperText Markup Language)* es un **lenguaje de marcado** que define la estructura del contenido de una página Web. Este lenguaje consiste en una serie de elementos que usarás para encerrar diferentes partes del contenido para que se vean o comporten de una determinada manera.

Importante: **HTML no es un lenguaje de programación**. Es un lenguaje de marcado. Sirve para indicar una estructura de elementos, no para dar instrucciones a la máquina.

CSS

CSS *(Cascading Style Sheets)* u *Hojas de estilo en cascada* es un **lenguaje de hojas de estilo**, es decir, te permite aplicar estilos de manera selectiva a elementos en documentos HTML.

Importante: **CSS no es un lenguaje de programación. Tampoco de marcado.** Es un lenguaje de hojas de estilo. Sirve para dar estilo a elementos que ya fueron previamente indicados y ubicados en una estructura. Otros ejemplos de lenguajes de estilo son XSL y Sass.

Javascript

JavaScript es un **lenguaje de programación** que se utiliza para añadir interactividad a las páginas web. Permite crear animaciones, juegos, formularios dinámicos y mucho más. Es uno de los lenguajes de programación más populares del mundo y se utiliza en millones de sitios web.

Javascript - ejemplo de código

```
const title = document.getElementById("titulo")

title.addEventListener("mouseover", () => {
    alert("Hola")
})

const changeColor = document.getElementById("change-color")

changeColor.onclick = function (event) {
    const R = randomNumber().toString()
    const G = randomNumber().toString()
    const B = randomNumber().toString()
    title.style.color = `rgb(${R},${G},${B})`
}

function randomNumber() {
    return (Math.random() * 256).toFixed(0)
}
```

Más adelante explicaremos qué está ocurriendo en este script, no hace falta que lo entiendan ahora

Javascript

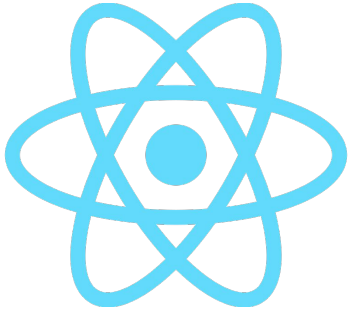
Tipos de datos

Number	Representa un número, no lleva comillas.
String	Cadena de texto, debe ir entre comillas.
Boolean	Tienen valor verdadero/falso (true/false). No llevan comillas
Array	Estructura que permite almacenar varios valores en una sola referencia
Object	Básicamente cualquier cosa. Todo en javascript es un objeto.

Operaciones

Suma/ Concatenación	Suma dos números, o junta dos cadenas en una	+
Resta, multiplicación, división	Mismo resultado matemático	-, *, /
Asignación	Asigna un valor a una variable	=
Identidad/ Igualdad	Comprueba si dos valores son iguales entre sí, y devuelve un booleano	===
Negación/ Distinto	Utilizado con el operador de identidad, la negación es en JS el equivalente al operador lógico NOT	!, !==

Frameworks de frontend



React

Angular

Vue.js

Ember.js

Svelte

No trataremos en profundidad con ninguno de estos frameworks en esta materia

Ejemplo práctico

Instalaciones necesarias

-Verificar que tenemos instalado Python.

-Actualizar pip -> Pip es un sistema de gestión de paquetes

Windows : python -m pip install -U pip

Linux: pip install -U pip

Mac: pip install -U pip

Entorno virtual

Virtual environment

Para comenzar con este ejemplo utilizaremos un **entorno virtual**. Los entornos virtuales nos permiten instalar y utilizar librerías y la versión de Python necesaria para nuestro proyecto.

Para esto utilizaremos pipenv en nuestra carpeta del proyecto

Luego instalaremos la libreria flask

Podemos verificar las dependencias instaladas

Entonces podemos comenzar nuestro proyecto

Flask - Estructura básica

```python
from flask import Flask

app = Flask(__name__)

if __name__ == "__main__":
    app.run(debug=True)
```

Este código nos permite levantar el servidor de nuestro sitio.
Por medio del parámetro debug podremos ver los cambios en tiempo real.

Por medio de funciones podremos renderizar contenido HTML tanto estática como dinámicamente.

```python
@app.route('/')
def index():
    return "Hola mundo"
```

HTML - Ejemplo de una página

```
<!DOCTYPE html>
<head>
    <title>Mi página web</title>

    <!-- Inserto estilos de CSS -->
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <h1 id="titulo">Título principal</h1>

    <!-- Cuerpo de nuestra página Web -->

    <!-- Inserto funcionalidad de Javascript -->
    <script src="script.js"></script>
</body>
</html>
```

Flask - Templates y Jinja

Flask nos permite utilizar y renderizar templates (básicamente archivos HTML que sirven de base a lo que se quiera mostrar). Para esto es necesario cargar los archivos HTML necesarios a la carpeta templates dentro del proyecto, e importar la función **render_template.** A partir de esta función también es posible pasar datos para que se rendericen.

Jinja nos permite agregar funcionalidad a estos templates, haciendo que puedan integrar data desde el template dinámicamente.

Flask - Routing

Con Flask es sencillo definir el enrutamiento de distintas páginas por medio de decoradores de Python que definan la ruta.

```python
@app.route('/')
def index():
    return "Hola mundo"
```

```python
@app.route('/about')
def about():
    return render_template('about.html')
```


.UBAfiuba
FACULTAD DE INGENIERÍA