**Universidad del Valle de Guatemala**

Ingeniería Software I
Sección 20

# FRAMEWORKS

## DJANGO Y REACT

Investigación II

Pablo José Méndez Alvarado – 23975
Luis Fernando Palacios López – 23933
Roberto Samuel Nájera Marroquín – 23781
André Emilio Pivaral López – 23574

# django

# Datos Generales

**Nombre**: Django.

**Descripción**: Es un framework web de código abierto escrito en Python, diseñado para facilitar el desarrollo rápido y eficiente de aplicaciones web robustas y seguras.

**Propósito**: Simplificar la creación de sitios web complejos y basados en bases de datos, proporcionando herramientas y funcionalidades integradas que permiten a los desarrolladores centrarse en aspectos específicos de sus aplicaciones sin reinventar componentes comunes.

# Área Desarrollo y Principios Funcionamiento

**Área Desarrollo**:

- Gestión Bases Datos: Facilita la interacción con bases de datos relacionales sin necesidad de escribir consultas SQL manuales.
- Seguridad: Incorpora medidas de seguridad integradas para proteger contra amenazas comunes.
- Autenticación y Autorización: Ofrece sistemas integrados para la gestión de usuarios y permisos.

**Principios Funcionamiento**:

- Don't Repeat Yourself (DRY): Promueve la reutilización de código y la reducción de redundancias.
- Convención sobre Configuración: Establece convenciones predeterminadas para minimizar la necesidad de configuraciones explícitas.
- Desarrollo Rápido: Facilita la creación ágil de aplicaciones.

# Componentes y Estructura

**URL Dispatcher**: Gestiona las solicitudes entrantes del navegador web y las dirige hacia la vista correspondiente según la configuración de URL.

**Caching Framework**: Almacena temporalmente respuestas frecuentespara reducir el tiempo de carga.

**Template**: Define la estructura y presentación visual de los datos recibidos desde la vista.

**View**: Procesa las solicitudes recibidas, interactúa con el modelo para obtener datos, aplica lógica y envía la información a las plantillas.

**Model**: Gestiona y define la estructura de datos de la aplicación, interactuando directamente con la base de datos.

**Database**: Almacena y gestiona  los datos que la aplicación utiliza, consultados y manipulados mediante los modelos del framework.

# Aplicaciones Recomendadas

Django es especialmente recomendable en situaciones que requieren:

- **Desarrollo Rápido**: Proyectos con plazos ajustados que necesitan una implementación ágil y eficiente.
- **Aplicaciones Seguras**: Sistemas que manejan datos sensibles y requieren medidas de seguridad robustas integradas.
- **Escalabilidad**: Proyectos que anticipan un crecimiento significativo y necesitan una arquitectura que soporte una alta demanda.
- **Aplicaciones Basadas en Contenido**: Sitios que requieren una gestión eficiente de contenido, como blogs, portales de noticias o sistemas de gestión de contenido.

# Cuanto cumple en el proceso de desarrollo

**Gestion de bases de datos**

**Enrutamiento y vistas**

**Autenticación y autorización**

# Tecnologías combinables

**Fronted Moderno**

**APIs y Django Rest framework**

**Servicios de terceros**

# Patrones de diseño y arquitectónicos

**Model Template View (MTV)**

**DRY (Don’t Repeat Yourself)**

**Convención sobre Configuración**

# React JS

# Datos Generales

**Nombre**: React.js

**Descripción**: Es una biblioteca de JavaScript desarrollada por Facebook para construir interfaces de usuario (UI), especialmente para aplicaciones web de una sola página (SPA). Permite crear componentes reutilizables y dinámicos con una estructura declarativa y basada en estados.

**Propósito**: Facilitar la construcción de interfaces de usuario interactivas, eficientes y modulares, promoviendo la reutilización de componentes y una actualización eficiente del DOM.

# Área Desarrollo y Principios Funcionamiento

**Área Desarrollo**:

- React se enfoca en el frontend. Se usa para construir la interfaz gráfica (UI) y gestionar el comportamiento dinámico de los componentes visuales.
- En particular, mejora la experiencia del usuario y la eficiencia de desarrollo al manejar la vista de una aplicación.

**Principios Funcionamiento**:

- Componentes: Las UIs se dividen en pequeñas piezas reutilizables llamadas componentes.
- JSX: Sintaxis que permite escribir HTML dentro de JavaScript.
- Estado (State) y Propiedades (Props): Controlan la lógica y el flujo de datos en los componentes.
- Virtual DOM: React utiliza una representación en memoria del DOM que permite hacer actualizaciones eficientes.
- Unidireccionalidad de datos: Los datos fluyen de forma descendente (de padres a hijos), lo que facilita el control del flujo de información.

# Componentes y Estructura

**Componentes funcionales:** Simples funciones de JavaScript que retornan JSX.

**Hooks**: Funciones especiales (como useState, useEffect) que permiten manejar estado y efectos secundarios en componentes funcionales.

**Componentes de clase**: Antiguamente más comunes, permiten manejar estado y ciclos de vida mediante clases (menos usados con la introducción de hooks).

**Context API**: Para compartir datos globales sin necesidad de prop drilling.

**React Router**: Para manejar la navegación entre páginas en SPAs.

**Redux / Zustand / Recoil**: Bibliotecas externas comúnmente usadas con React para manejar estados complejos.

.

# Aplicaciones Recomendadas

React es especialmente recomendable en situaciones que requieren:

- **Aplicaciones web interactivas y dinámicas**: Cuando se requiere construir interfaces de usuario interactivas y dinámicas.
- **Aplicaciones SPA**: En aplicaciones de una sola página (SPA) que necesitan un rendimiento alto.
- **Sitios con componentes reutilizables**: Cuando se busca un enfoque componente-reutilizable y mantenible.
- **Mantenimiento del DOM**: En proyectos que requieren una actualización eficiente del DOM.
- **Escala y soporte**: Para equipos que desean trabajar con una gran comunidad, soporte y ecosistema (librerías, herramientas, etc.).

# Tipo de Framework

# ¿Cuánto del proceso de desarrollo cumple?

🔹 **Cubre:**

✅ **Creación de interfaces de usuario (UI)**
✅ **Gestión del estado (con Context API, Redux, etc.)**
✅ **Interactividad en el cliente**

🔸 **No cubre directamente:**

❌ **Backend (APIs, bases de datos)**
❌ **Gestión avanzada del estado global (sin herramientas adicionales)**
❌ **Seguridad y autenticación**

# ¿Es necesario combinarlo con otras tecnologías?

# Patrones de diseño y arquitectónicos

**Render Props:**

**FLUX/REDUX**

**Component-Based Architecture**

# Semejanzas

Componentes Reutilizables:
Tanto Django como React fomentan la creación de componentes reutilizables.

Soporte de Comunidad:
Ambos tienen comunidades de desarrolladores activas y robustas.

Rendimiento y Escalabilidad:
Ambos frameworks son capaces de manejar aplicaciones de gran escala y alto rendimiento.

**Tipo de Framework**

**Arquitectura**

**Uso Principal**