Universidad del Valle de Guatemala

Facultad de Ingeniería

Departamento de Ciencia de la Computación

Ingeniería de Software I

FRAMEWORKS

DJANGO Y REACT

Pablo José Méndez Alvarado – 23975

Luis Fernando Palacios López – 239333

Roberto Samuel Nájera Marroquín – 23781

André Emilio Pivaral López – 23574

Catedrático: Erick Francisco Marroquín Rodríguez

Sección: 20

Nueva Guatemala de la Asunción, 12 de marzo de 2024

Índice

Resumen

El presente documento expone un análisis comparativo entre dos tecnologías ampliamente utilizadas en el desarrollo web moderno: Django y React. Django, un framework de backend basado en Python, permite la creación de aplicaciones web robustas, seguras y escalables, facilitando el trabajo con bases de datos, autenticación, administración y más. Por otro lado, React es una biblioteca de frontend basada en JavaScript, especializada en la construcción de interfaces de usuario dinámicas y eficientes mediante un enfoque declarativo y basado en componentes. A lo largo del documento se describen sus características, principios de funcionamiento, componentes clave, casos de uso recomendados, y se realiza una comparación que destaca sus similitudes y diferencias. El objetivo es proporcionar una visión integral que permita comprender cuándo y cómo utilizar cada tecnología de forma eficiente, tanto individualmente como en conjunto.

Introducción

En el desarrollo de aplicaciones web modernas, la elección de las tecnologías adecuadas es un factor crucial para alcanzar soluciones eficientes, escalables y mantenibles. Django y React representan dos herramientas fundamentales que, aunque abordan diferentes áreas del desarrollo (backend y frontend, respectivamente), pueden complementarse para crear aplicaciones completas y robustas. Django destaca por su enfoque integral y su estructura basada en el patrón Model-Template-View (MTV), lo que permite un desarrollo rápido y seguro en el lado del servidor. React, en cambio, ofrece una experiencia de usuario dinámica y fluida mediante el uso de componentes reutilizables y un Virtual DOM eficiente. Este trabajo tiene como finalidad explorar a fondo las características técnicas de ambos frameworks, evaluar sus ventajas y limitaciones, y establecer una comparación que facilite la toma de decisiones informadas en proyectos de desarrollo web.

Django

Nombre

Django

Descripción general

Es un framework web de código abierto escrito en Python, diseñado para facilitar el desarrollo rápido y eficiente de aplicaciones web robustas y seguras (Django Software Foundation, 2024).

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 del desarrollo que resuelve

- 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 de 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.

Tipo de framework

Django es un framework de desarrollo web de alto nivel basado en el lenguaje de programación Python. Está diseñado para ayudar a los desarrolladores a construir aplicaciones web de manera rápida y eficiente. Django sigue el principio de "la batería incluida", lo que significa que proporciona muchas características y funcionalidades integradas, reduciendo la necesidad de utilizar bibliotecas externas. Entre sus características destacan la gestión de bases de datos mediante un ORM (Object- Relational Mapping), un sistema de administración integrado, la autenticación de usuarios, y la gestión de formularios, entre otros. Django promueve el desarrollo rápido de aplicaciones, el diseño limpio y pragmático, y la reutilización de código, lo que facilita la creación de aplicaciones web escalables y mantenibles (Django Software Foundation, n.d.).

¿Cuánto del proceso de desarrollo cubre?

Django cubre una gran parte del proceso de desarrollo de aplicaciones web, desde la gestión de bases de datos hasta la interfaz de usuario. Estos pueden ser:

• Gestión de Bases de Datos:
Proporciona un ORM que permite a los desarrolladores interactuar con bases de datos utilizando código Python en lugar de SQL. Esto facilita la creación y manipulación de modelos de datos y relaciones entre ellos (Django Software Foundation, n.d.).

• Autenticación y Autorización:
Incluye un sistema de autenticación de usuarios integrado que permite gestionar el registro, inicio de sesión, permisos y roles de los usuarios (Django Software Foundation, n.d.).

• Enrutamiento y Vistas:
Facilita la creación de rutas URL y vistas que manejan las solicitudes del usuario y devuelven respuestas. Las vistas pueden ser genéricas, basadas en clases o funciones, lo que permite flexibilidad en el desarrollo (Django Software Foundation, n.d.).

• Sistemas de Plantillas:

Utiliza un sistema de plantillas potente que permite separar la lógica del negocio de la presentación, facilitando la creación de interfaces de usuario dinámicas y reutilizables (Django Software Foundation, n.d.).

• Validación de Formularios:

Proporciona herramientas para crear y validar formularios HTML, simplificando la recolección y validación de datos del usuario (Django Software Foundation, n.d.).

• Internacionalización y Localización:

Soporta la creación de aplicaciones multilingües y permite la traducción de textos y la

adaptación de formatos según la localización del usuario (Django Software Foundation, n.d.).

• Interfaz de Administración:

Incluye una interfaz de administración lista para usar que permite gestionar modelos, usuarios y contenidos de manera eficiente (Django Software Foundation, n.d.).

• Seguridad:

Incluye medidas de seguridad integradas como protección contra ataques CSRF, XSS, inyección SQL, y más (Django Software Foundation, n.d.).

¿Es necesario combinarlo con otras tecnologías?

Aunque Django es un framework completo y poderoso, a menudo se combina con otras tecnologías para mejorar la funcionalidad y la experiencia del usuario. Algunas combinaciones podrían ser:

• Frontend Moderno:

Frameworks de frontend como React, Angular o Vue.js pueden integrarse con Django para crear aplicaciones de una sola página (SPA) con interfaces de usuario dinámicas e interactivas. Mientras Django maneja el backend, React o Vue.js pueden encargarse del frontend, proporcionando una experiencia más fluida y responsiva.

• APIs y Django REST Framework:
Django REST Framework (DRF) es una biblioteca que extiende las capacidades de Django para construir APIs RESTful robustas y escalables. Esto es ideal si necesitas que tu aplicación interactúe con otras aplicaciones móviles o web.

• Servicios de Terceros:

Django puede integrarse con servicios externos como servicios de pago (Stripe, PayPal),

servicios de envío de correos electrónicos (SendGrid, Mailgun), y servicios de almacenamiento en la nube (AWS S3, Google Cloud Storage).

• Bases de Datos:
Aunque Django incluye soporte para múltiples bases de datos, seleccionar la base de datos correcta depende de las necesidades específicas del proyecto. Por ejemplo, PostgreSQL es una opción popular para aplicaciones Django debido a sus características avanzadas y rendimiento robusto.
• Caché y Colas de Tareas:
Para mejorar el rendimiento, Django puede integrarse con sistemas de caché (Redis, Memcached) y colas de tareas (Celery) para manejar operaciones costosas y tareas en segundo plano.

Patrones de diseño y arquitectónicos que implementa y en que parte de este son implementados.

Django sigue varios patrones de diseño y arquitectónicos que contribuyen a su robustez y eficiencia:

• Model-Template-View (MTV):
Este patrón es similar al patrón Model-View-Controller (MVC) pero adaptado a la estructura de Django:

o Model (Modelo): Representa la estructura de los datos y las reglas de negocio. Se define en el archivo models.py y utiliza el ORM de Django para interactuar con la base de datos.

o Template (Plantilla): Se encarga de la representación de la interfaz de usuario. Los archivos de plantillas se guardan en el directorio templates y utilizan el sistema de plantillas de Django para generar HTML dinámico.

o View (Vista): Controla la lógica de negocio y las interacciones del usuario.

Se define en el archivo views.py y maneja las solicitudes HTTP, interactúa con los modelos y devuelve respuestas HTML o JSON.

• DRY (Don't Repeat Yourself):
Django promueve la reutilización de código y la reducción de redundancias mediante su diseño modular y la creación de aplicaciones reutilizables. Esto facilita el mantenimiento y la escalabilidad del código.

• Reusable Apps:
Django permite dividir la funcionalidad en aplicaciones reutilizables, lo que facilita la organización del código y permite reutilizar componentes en diferentes proyectos. Cada aplicación puede incluir modelos, vistas, plantillas y archivos estáticos.

• Convención sobre Configuración:
Django sigue el principio de "Convención sobre Configuración", proporcionando configuraciones predeterminadas sensatas y reduciendo la necesidad de configuraciones personalizadas. Esto acelera el desarrollo y minimiza errores de configuración.

Situaciones donde se recomienda su uso

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.

React

Nombre

React.js

Descripción general

React es una biblioteca de JavaScript de código abierto desarrollada por Facebook para construir interfaces de usuario, especialmente en aplicaciones web de una sola página (SPA). Se centra en la construcción de componentes reutilizables y en la actualización eficiente del DOM (Document Object Model) mediante un enfoque declarativo (Facebook, 2024).

Propósito

Su objetivo principal es facilitar la creación de interfaces de usuario interactivas, rápidas y eficientes. React permite a los desarrolladores diseñar vistas que se actualizan dinámicamente cuando los datos cambian, sin necesidad de manipular directamente el DOM (Banks y Porcello, 2020).

Área del desarrollo que resuelve

React se enfoca en el frontend, es decir, en la capa de presentación de las aplicaciones web. 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 (Facebook, 2024).

Principios de 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.

(Banks y Porcello, 2020).

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.

(Facebook Open Source, 2024)

Tipo de framework

React.js es un framework de frontend (aunque técnicamente es una biblioteca, a menudo se le considera un framework por su ecosistema y forma de trabajo).

(Banks y Porcello, 2020).

¿Cuánto del proceso de desarrollo cubre?

Como se mencionó anteriormente, esta herramienta cubre principalmente la capa de frontend en el desarrollo de una aplicación, enfocándose en la interfaz de usuario y la interacción con el usuario. Permite diseñar interfaces a través de componentes reutilizables, gestionar el estado de la aplicación con herramientas como React Context o Redux, e integrar el frontend con el backend

mediante peticiones HTTP a APIs. Además, facilita la navegación en aplicaciones de una sola página (SPA) con React Router y optimiza el rendimiento a través del Virtual DOM.

(Banks y Porcello, 2020).

¿Es necesario combinarlo con otras tecnologías?

Sí, ya que para gestionar datos y la lógica de negocio, se requiere un backend desarrollado con tecnologías como el ya mencionado Django o Spring, junto con una base de datos como MongoDB, MySQL o PostgreSQL. Además, en muchos casos, se complementa con herramientas como Redux para la gestión del estado global, React Router para la navegación y bibliotecas de UI como Material-UI o Tailwind CSS para mejorar el diseño. Aunque React puede usarse solo para construir interfaces, su verdadero potencial se aprovecha cuando se integra con un ecosistema más amplio de tecnologías.

(Fedosejev, A, 2015)

Patrones de diseño y arquitectónicos que implementa y en que parte de este son implementados.

1. Composición de Componentes (Component-Based Architecture)
   React se basa en la reutilización de componentes independientes, fomentando una arquitectura modular donde cada componente maneja su propia lógica y representación visual.

2. Render Props
   Se utiliza para compartir lógica entre componentes mediante funciones pasadas como props, permitiendo un mejor control del renderizado y evitando duplicación de código.

3. Higher-Order Components (HOC)
   Un patrón que permite reutilizar la lógica de los componentes al envolver uno dentro de otro, comúnmente usado para manejar autenticación o permisos.

4. Hooks (Functional Programming)
   Introducen un enfoque declarativo basado en funciones, permitiendo el uso de estado y efectos secundarios en componentes funcionales sin necesidad de clases.

5. Flux/Redux (Unidirectional Data Flow)
   React promueve un flujo de datos unidireccional, lo que mejora la predictibilidad del estado de la aplicación. Librerías como Redux implementan este principio para gestionar el estado global.

6. Lazy Loading y Code Splitting
   Mejora el rendimiento cargando solo las partes de la aplicación necesarias en un momento dado, utilizando herramientas como React.lazy y Suspense.

7. MVVM (Model-View-ViewModel, parcialmente)
   Aunque React no sigue estrictamente este patrón, la separación entre la vista (UI) y la lógica (estado y controladores) se asemeja al modelo MVVM.

(Bertoli, M. 2017).

Situaciones donde se recomienda su uso

- 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.).

Semejanzas y diferencias de los Frameworks seleccionados.

Criterio	Django	React
Componentes Reutilizables	Usa aplicaciones modulares para organizar el código.	Utiliza componentes de UI reutilizables.
Soporte de Comunidad	Gran comunidad con documentación extensa y soporte.	Comunidad activa con muchos recursos y librerías.
Rendimiento y Escalabilidad	Capaz de manejar aplicaciones grandes con configuración adecuada.	Optimizado para aplicaciones dinámicas con renderizado eficiente.
Tipo de Framework	Framework completo de backend en Python.	Biblioteca de frontend en JavaScript.
Arquitectura	Sigue el patrón MTV (Model-Template-View).	Basado en componentes y flujo unidireccional de datos (Redux, Context API).
Uso Principal	Desarrollar aplicaciones web completas con backend y frontend.	Crear interfaces de usuario interactivas y dinámicas.
Lenguaje de Programación	Usa Python, con sintaxis clara y sencilla.	Usa JavaScript/TypeScript, flexible y potente en el frontend.
Templating vs. Componentes	Usa un sistema de plantillas para generar HTML dinámico.	Usa componentes reutilizables para renderizar la UI de forma declarativa.

Conclusiones

- Django y React se complementan eficazmente en el desarrollo de aplicaciones web modernas. Django, siendo un framework de backend, maneja la lógica del servidor, la administración de bases de datos y la generación de HTML. Por otro lado, React, como una biblioteca de frontend, se especializa en la creación de interfaces de usuario interactivas y dinámicas. Usados juntos, pueden ofrecer una solución completa y robusta.

- Django utiliza el patrón Model-Template-View (MTV) y promueve principios como "Don't Repeat Yourself" (DRY) y "Convención sobre Configuración". React se basa en un enfoque basado en componentes y promueve un flujo unidireccional de datos, utilizando herramientas como Redux para la gestión del estado global. Estas diferencias en enfoques y arquitecturas permiten a los desarrolladores seleccionar la herramienta adecuada según las necesidades específicas del proyecto.

- Django es especialmente recomendable para proyectos que requieren un desarrollo rápido y seguro, y que manejan datos sensibles o necesitan una arquitectura escalable. En cambio, React es ideal para aplicaciones que requieren interfaces de usuario interactivas, como aplicaciones de una sola página (SPA) y sitios web con componentes reutilizables.

Referencias

Banks, B., & Porcello, A. (2020). *Learning React: Functional Web Development with React and Redux* (2nd ed.). O'Reilly Media.

Django Software Foundation. (2024). *Django documentation*. https://docs.djangoproject.com/en/stable/

Facebook Open Source. (2024). *React – A JavaScript library for building user interfaces*. https://reactjs.org/

Melé, A. (2020). *Django 3 by example*. Packt Publishing.

Meta Platforms, Inc. (n.d.). React documentation. Recuperado de https://reactjs.org/docs/getting-started.html

Vincent, W. S. (2022). *Django for Beginners: Build websites with Python and Django* (4th ed.). WelcomeToCode.

Vincent, W. S. (2020). *Django for professionals*. Amazon Digital Services LLC.

Roy, A. (2023). *Mastering Django: Core*. Independently published.

Fedosejev, A. (2015). React. js essentials. Packt Publishing Ltd.

Bertoli, M. (2017). React Design Patterns and Best Practices. Packt Publishing Ltd.

Anexos

Cuadro 1. Planificación grupal de las actividades a realizar

Tarea	**Encargado**	**Fecha**	**Inicio**	**Fin**
Asignación de tareas y organización	André Pivaral y Roberto Nájera	10/03/2025	23:00	00:00

concerniente a la entrega				
Redacción de Resumen e Introducción	André Pivaral	11/03/2025	08:00	10:00
Primera parte de la investigación general del Framework de React	Roberto Nájera	12/03/2025	08:00	10:00
Segunda parte de la investigación de React	Luis Palacios	12/03/2025	08:00	10:00
Segunda parte de la investigación de Django	Pablo Méndez	12/03/2025	08:00	10:00
Primera parte de la investigación general del Framework de Django	André Pivaral	12/03/2025	8:00	10:00
Diseño de la presentación a utilizar en la exposición	Roberto Nájera, André Pivaral, Luis Palacios, Pablo Méndez	12/03/2025	12:00	14:00

Corrección final del documento de la investigación	André Pivaral y Luis Palacios	13/03/2025	12:00	14:00

Enlace a la presentación:
https://www.canva.com/design/DAGhYEfumUU/LzuyRSrrgEFlYpN9w5i0Ew/edit?utm_content=DAGhYEfumUU&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

Enlace al repositorio con los informes de trabajo:

https://github.com/Ultimate-Truth-Seeker/frameworks

Informe de gestión del tiempo grupal

Fecha	**Inicio**	**Fin**	**Tiempo Interrupción**	**Δ Tiempo**	**Fase**	**Comentarios**
10/03/2025	18:00	13/03/2025, 23:59	2d 12h	12h	Investigación de Frameworks	Positivo: Se tuvo buena comunicación, todos entregaron la parte que les correspondía Mejorar: puntualidad de los avances, evitar procrastinación