UNIVERSIDAD DEL VALLE DE GUATEMALA

Tarea Investigativa 2.

BRANDON JAVIER REYES MORALES - 22992
CARLOS ALBERTO VALLADARES GUERRA - 221164
GUSTAVO ADOLFO CRUZ BARDALES- 22779
PAULA REBECA BARILLAS ALVAREZ - 22764
RODRIGO ALFONSO MANSILLA DUBÓN - 22611

Catedrático: Lynette García Pérez

Ingeniería de Software
Guatemala, 2024

Frameworks de desarrollo

PLANIFICACIÓN DE ACTIVIDADES

1. Investigación de Frameworks
    a. Investigación Preliminar de frameworks.
        i. Encargado: Todos
            1. Fecha: 3-4 Marzo
    b. Investigación general que abarque los temas:
        i. Descripción del Framework
        ii. Principios de funcionamiento
        iii. tipo de framework y alcanc3e
        iv. Patrones de diseño y arquitectonicos implementados
        v. Recomendaciones de uso
            1. Encargado Framework 1: Paula Barillas, Rodrigo Mansilla
                a. Fechas: 7-11 Marzo
            2. Encargado Framework 2: Gustavo Cruz, Carlos Valladares
                a. Fechas: 7-11 Marzo
2. Comparación de Frameworks
    a. Analizar los frameworks y analizar semejanzas y similitudes.
        i. Encargado: Brandon Reyes
            1. 10 Marzo
3. Presentación.
    a. Realizar presentación que sintetice los puntos importantes del informe.
        i. Encargado: Rodrigo Mansilla
        ii. Encargado: Carlos Valladares
            1. Fecha : 12 Marzo

Planificación:

Actividad	Encargados	Fecha	Tareas/Objetivos
Investigación Preliminar de Frameworks	Todos los integrantes	3-4 de marzo	Investigación preliminar
Investigación General Framework 1 (Frontend)	Paula Barillas, Rodrigo Mansilla	7-11 de marzo	Investigación general FW 1
Investigación General Framework 2 (Backend)	Gustavo Cruz, Carlos Valladares	7-11 de marzo	Investigación General FW 2
Comparación de Frameworks	Brandon Reyes	10 de marzo	Comparación
Presentación	Rodrigo Mansilla, Carlos Valladares	12 de marzo	Presentación
Revisión Final	Todos los integrantes	13 de marzo 11:00 AM	

SOLID

Nombre

- SOLID

Descripción general (propósito)

- Representa cinco principios de diseño de software orientado a objetos. Teniendo como objetivo principal el crear un código extensible, flexible y mantenible. Donde es robusto y estable además permite escalabilidad donde acepta ser ampliado con nuevas funciones de manera ágil.

Área de desarrollo de las aplicaciones que resuelve

- Aborda aspectos fundamentales del diseño de software, especializándo se en la orientación a objetos. Sin importar un lenguaje específico de programación.

Principios de funcionamiento y componentes.

- Principio de Responsabilidad Única (SRP): Una clase debe tener una y solo una razón para cambiar.
- Principio de Abierto/Cerrado (OCP): Las entidades de software deben estar abiertas para su extensión, pero cerradas para su modificación.
- Principio de Sustitución de Liskov (LSP): Las clases derivadas deben ser sustituibles por sus clases base.
- Principio de Segregación de Interfaces (ISP): Los clientes no deben estar forzados a depender de interfaces que no utilizan.
- Principio de Inversión de Dependencias (DIP): Depende de abstracciones, no de implementaciones concretas.

Tipo de framework:

- SOLID no es un framework en sí mismo, sino un conjunto de principios de diseño que deben aplicarse en el proceso de desarrollo de software orientado a objetos.

¿El proceso de desarrollo cuánto cubre? , combinaciones con otras tecnologías

- SOLID no cubre el proceso de desarrollo completo, sino que proporciona una guía para diseñar y estructurar el código de manera más modular, flexible y mantenible.

Patrones de diseño y arquitectónicos que implementa Solid y en qué parte del mismo son implementados.

- SOLID no implementa patrones de diseño o arquitecturas específicas, sino que establece principios fundamentales para crear diseños de software más sólidos. Sin embargo, muchos patrones de diseño conocidos, como el Patrón de Estrategia, el Patrón de Observador y el Patrón de Fábrica, siguen los principios de SOLID. Estos principios se aplican en diferentes partes del diseño de software, como la definición de clases, la creación de interfaces, la gestión de dependencias y la estructura de la aplicación.

Situaciones donde se recomienda su uso

- En proyectos de software donde se oriente a objetos. Ya que esto hace que el código sea resistente a futuros cambios. Donde sus principios ayudan en los proyectos de largo plazo, también para sistemas complejos y entornos donde se prevén cambios frecuentes. Facilitando la prueba y el refactorizado del código.

FLASK

Nombre

- Flask

Descripción general (propósito)

- Es un framework web, de Python. Su propósito es brindar librerías o módulos, para que sea flexible y fácil de utilizar, manteniendo las operaciones de bajo nivel, como el protocolo, manejo de procesos, etc.

Área de desarrollo de las aplicaciones que resuelve

- Flask, debido a su flexibilidad y facilidad de uso es utilizado en amplia gama de proyectos. Sin embargo, es más utilizado para desarrollar prototipos rápidos y aplicaciones simples que no requieren de estructuras complejas para su funcionamiento.

Principios de funcionamiento y componentes.

- Sigue el principio "WSGI" (Web Server Gateway Interface) para manejar las solicitudes HTTP. El cual tiene un diseño modular, con componentes como enrutadores para manejar las rutas URL.
- Además, utiliza jinja2, el cual es un template web que combina una estructura de data específica, que permite, posteriormente, convertir variables de python, en un template HTML.

Tipo de framework:

- Flask es un microframework que no impone herramientas o bibliotecas específicas.

¿El proceso de desarrollo cuánto cubre? , combinaciones con otras tecnologías

- Se utiliza para la creación de aplicaciones simples como también complejas. Donde al combinar con otras tecnologías de la libertada, ejemplo:
    - Base de Datos: Flask no impone una base de datos específica, lo que te da libertad para elegir entre SQL (usando SQLAlchemy) o bases de datos NoSQL (como MongoDB).
    - Frontend: Puedes combinar Flask con frameworks frontend como React, Angular o Vue.js. Flask proporciona endpoints API para que tu frontend se comunique con el backend.
    - Autenticación y Autorización: Flask se integra fácilmente con herramientas como Flask-Login o Flask-Security para gestionar la autenticación y la autorización.
    - Despliegue: Para implementar tu aplicación, puedes usar servidores web como Gunicorn o uWSGI, y servidores de aplicaciones como Nginx o Apache.

Patrones de diseño y arquitectónicos que implementa Flask y en qué parte del mismo son implementados.

- Flask, también implementa el patrón de diseño Modelo-Vista-Controlador. Donde este patrón separa la lógica de la aplicación siendo el modelo -> , la vista -> la interfaz del usuario y el controlador -> manejo de solicitudes y respuestas. También se destaca por utilizar componentes como Jinja2 para plantillas web, Flask se convierte en una herramienta versátil para construir aplicaciones web personalizadas.

Situaciones donde se recomienda su uso

- **Prototipado:** Flask es usado para el prototipado de aplicaciones web de manera rápida. Debido a su simplicidad, permite a los desarrolladores crear y desplegar aplicaciones web en poco tiempo. Esto permite validar ideas previamente al desarrollo con frameworks especializados.

- **Proyectos pequeños o medianos:** Proyectos que no requieren la complejidad de frameworks más grandes, gracias a la flexibilidad que Flask ofrece para no sobrecargar el desarrollo con componentes innecesarios.

- **Servicios Web Y APIs RESTful:** Por su facilidad de manejo en solicitudes HTTPS y la compatibilidad con extensiones que facilitan el desarrollo de interfaces , es una buena opción considerar su uso en el desarrollo de servicios web y APIs RESTful.

SEMEJANZAS Y DIFERENCIAS DE SOLID Y FLASK

Semejanzas	Diferencias
Tanto SOLID como Flask tienen como objetivo final mejorar la calidad, mantenibilidad y flexibilidad del código.	SOLID es un conjunto de principios de diseño, mientras que Flask es un framework web completo con una estructura y componentes predefinidos.
Con los principios de SOLID, se crean clases y componentes más coherentes y desacoplados, lo que facilita la reutilización de código en diferentes partes de la aplicación o en otros proyectos. De manera similar, Flask alienta la reutilización de código a través de extensiones y blueprints (módulos de aplicación), que permiten encapsular y reutilizar funcionalidades específicas.	SOLID aborda el diseño de software a un nivel más abstracto y genérico, aplicable a cualquier sistema orientado a objetos. Por otro lado, Flask se enfoca específicamente en el desarrollo de aplicaciones web.

Referencias:

Martín, M. J. (2018, noviembre 21). SOLID: los 5 principios que te ayudarán a desarrollar software de calidad. Profile Software Services. *https://profile.es/blog/principios-solid-desarrollo-software-calidad/*

Flask — Documentación de Flask (2.3.x). (s/f). Readthedocs.io. https://flask-es.readthedocs.io/

SolidJS. (s/f). Solidjs.com. https://www.solidjs.com/

Paúl Tutillo. (2015). Proyecto previo a la obtención del titulo de magister en gestión de las comunicaciones y tecnologías de la información, https://bibdigital.epn.edu.ec/bitstream/15000/10481/1/CD-6201.pdf

Ahmed Bahgat, (2023, agosto 25). Flask vs Django: Elijamos Tu Próximo Framework Python. https://kinsta.com/es/blog/flask-vs-django/

Garcia Eduardo Ismael. (2019, noviembre 11). ¿Por qué aprender Flask? #Python, https://codigofacilito.com/articulos/por-que-flask

Link Presentación
https://www.canva.com/design/DAF-9d-xswA/ECZA1Z0FyKYUV6Bg8rfTPA/edit?utm_content=DAF-9d-xswA&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

Link formulario:
https://docs.google.com/spreadsheets/d/1AVvDIN2bwq_1klPaG0dBAF55KKyZXlBBkL9XaL8q-3Q/edit#gid=0