UNIVERSIDAD CATÓLICA DE TEMUCO

Aplicación Práctica de Arquitectura de Software, Patrones de Diseño, Control de Versiones y Métricas de Calidad

Integrantes: Sebastian Ayenao Sebastian Victoriano
Carrera: Técnico Universitario en Informática
Curso: Diseño y Desarrollo de Software.

UNIVERSIDAD CATÓLICA DE TEMUCO

Índice

Integrantes: Sebastian Ayenao Sebastian Victoriano....1
Carrera: Técnico Universitario en Informática.... 1
Curso: Diseño y Desarrollo de Software.... 1
Índice....2
Introducción.... 3
Objetivos....4
Contexto del proyecto.... 5
Aplicación de Arquitectura de Software.... 6
a) Diagrama de Módulos.... 6
b) Diagrama de Despliegue....6
c) Cohesión, Acoplamiento y Decisiones Arquitectónicas.... 7
Patrones de Diseño Aplicados.... 8
Patron Factory Method....8
Patron Singleton....8
Operaciones, Mantenimiento y Ciclo de Vida....9
Tipo de mantenimiento aplicado....9
Proceso de operaciones....9
DevOps:.... 9
Métricas de Calidad y KPIs del Sistema.... 10
1.Complejidad ciclomática....10
2. Cobertura de pruebas.... 10
3.Frecuencia de despliegue.... 10
Control de Versiones y Buenas Prácticas en Git.... 11
Uso de Inteligencia Artificial.... 12
Herramientas utilizadas:....12
Prompts utilizados:.... 12
Impacto....13
Ajustes realizados:.... 13
Conclusión Técnicas.... 13
Aprendizajes centrales:.... 13
Limitaciones del diseño:.... 13
Riesgos del sistema:.... 14
Mejoras a futuro:.... 14

UNIVERSIDAD CATÓLICA DE TEMUCO

Contexto del proyecto

-Nombre del sistema:

- Kaeli

-Problema que resuelve:

- Kaeli permite comparar precios de productos básicos entre supermercados chilenos, ayudando a los usuarios a ahorrar tiempo y dinero

-Usuario final principal:

- Responsable de compras del hogar, estudiantes y jóvenes independientes

-Arquitectura general:

- Cliente-servidor con patrón MVC. Frontend en HTML, backend en Flask(Python), base de datos en PostgreSQL

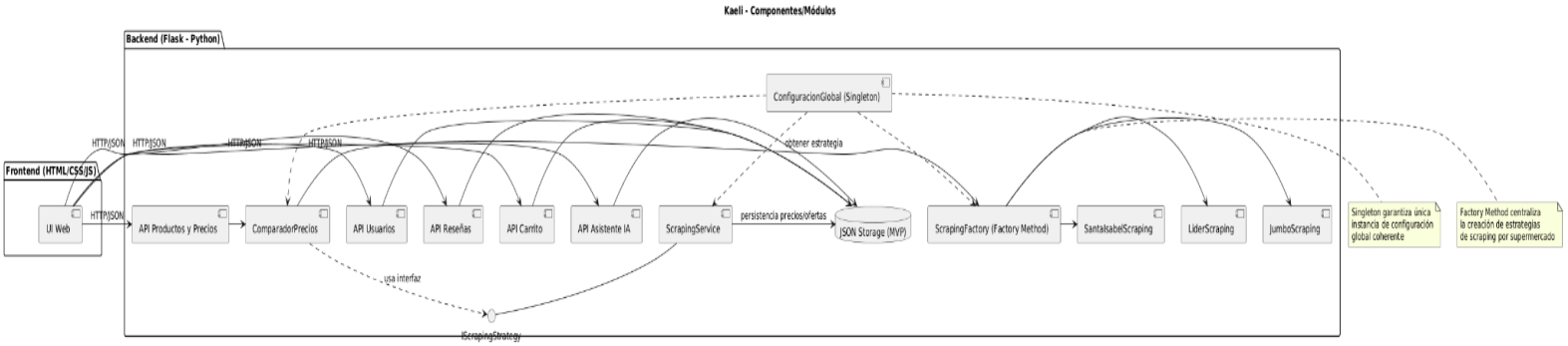
-Estado actual:

- Proyecto real en desarrollo, con MVP funcional que incluye registro/login, reseñas, carrito simulado y comparación de precios

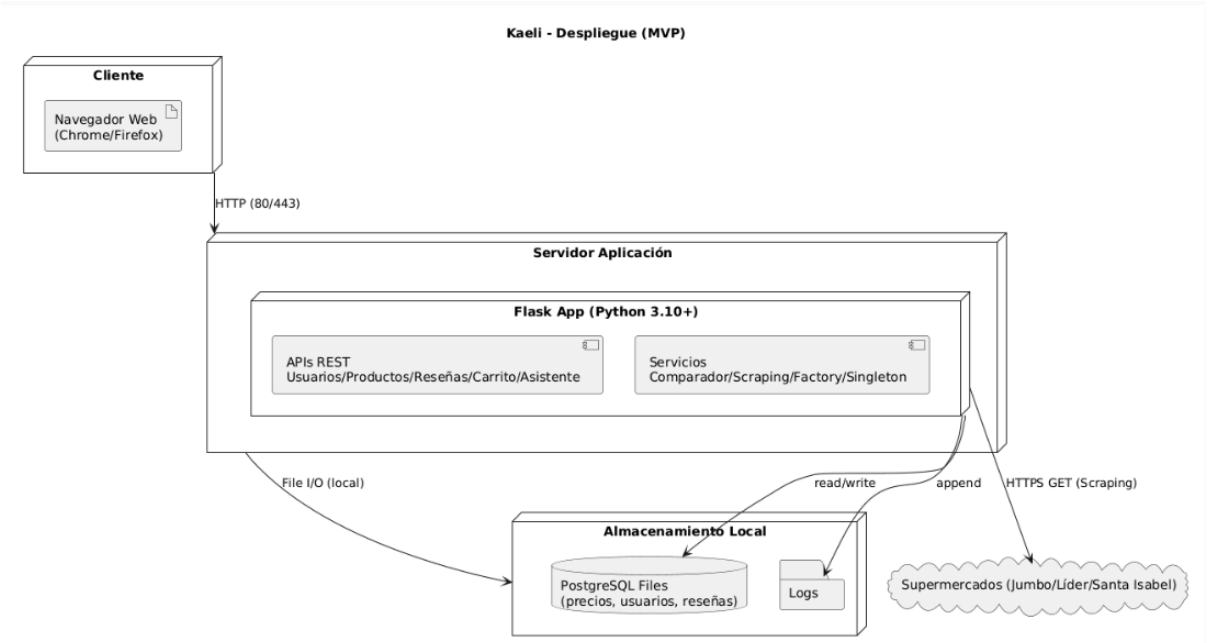
UNIVERSIDAD CATÓLICA DE TEMUCO

Aplicación de Arquitectura de Software

a) Diagrama de Módulos



b) Diagrama de Despliegue



- Nodo cliente: navegador web
- Nodo servidor: Flask API
- Contenedores: módulo de scraping, módulo de comparación, módulo de usuarios
- Base de datos: PostgreSQL
- Protocolo: HTTP

c) Cohesión, Acoplamiento y Decisiones Arquitectónicas

- Alta cohesión en módulos como Scraping y ComparadorPrecios
- Bajo acoplamiento logrado mediante interfaces (ScrapingStrategy) y patrón Factory
- Decisiones: uso de Singleton para configuración global, separación de lógica de scraping por supermercado

UNIVERSIDAD CATÓLICA DE TEMUCO

Patrones de Diseño Aplicados

Patron Factory Method

Aplicación:ScrapingFactory crea instancias de estrategias específicas (JumboScraping, LiderScraping).

Problema que resuelve: Permite manejar múltiples supermercados sin acoplar el código.

Justificación: Facilita agregar nuevos supermercados sin modificar el núcleo

```python
class ScrapingFactory:
    def crear_scraper(self, supermercado):
        if supermercado == "jumbo": return JumboScraping()
        elif supermercado == "lider": return LiderScraping()
        elif supermercado == "santa_isabel": return SantalsabelScraping()
```

Patron Singleton

Aplicación: ConfiguracionGlobal centraliza parámetros como API keys y rutas.

Problema que resuelve: Evita inconsistencias en configuración compartida.

Justificación: Asegura una única instancia accesible desde todos los módulos.

```python
class ConfiguracionGlobal:
    _instance = None

    def __new__(cls):
        if cls._instance is None:
            cls._instance = super().__new__(cls)

            cls._instance.api_key = "kaeli_2024"
```

UNIVERSIDAD CATÓLICA DE TEMUCO

Operaciones, Mantenimiento y Ciclo de Vida

Tipo de mantenimiento aplicado

- Correctivo: ajustes en el scraping por constantes cambios en el HTML
- Adaptativo: incorporación de nuevos supermercados o negocios locales
- Perfectivo: mejoras en interfaz y experiencia de usuario
- Evolutivo: app móvil y que los usuarios puedan subir ofertas

Proceso de operaciones

- Gestión de incidencias: vía GitHub Issues
- Monitorización: logs de scraping y errores
- Despliegue: en fases de pruebas será local
- Flujos operativos: revisión por pares, pruebas incrementales

DevOps:

Se promueve cultura DevOps mediante integración continua, revision de código y uso de GitHub

UNIVERSIDAD CATÓLICA DE TEMUCO

Métricas de Calidad y KPIs del Sistema

1.Complejidad ciclomática

- Definición: Número de caminos lógicos independientes en el código (if,while,for,case)
- Medición: Herramienta radon en Python
- Valor hipotético: 4 en módulo de scraping
- Interpretación: Moderada complejidad, mantenible
- Mejora**:** Refactorizar condiciones anidadas

2. Cobertura de pruebas

- Definición: Porcentaje de código cubierto por test (el porcentaje de código ejecutado al correr pruebas)
- Medición: pytest-cov
- Valor hipotético: 65%
- Interpretación: Aceptable para el MVP (Producto Mínimo Viable)
- Mejora: Añadir pruebas a módulos faltantes

3.Frecuencia de despliegue

- Definición: Número de veces que se publicará una nueva versión
- Medición: Generar commits con tag sobre la versión
- Valor hipotético: 1 despliegue cada 15 días
- Interpretación: Ritmo adecuado para proyecto universitario
- Mejora: Hacer que el proceso de despliegue sea automático mediante scripts

UNIVERSIDAD CATÓLICA DE TEMUCO

Control de Versiones y Buenas Prácticas en Git

El proyecto Kaeli utiliza Git como sistema de control de versiones, con repositorio alojado en GitHub. La gestión del código se organiza mediante ramas y commits, lo que permite mantener trazabilidad y colaboración ordenada entre los integrantes del equipo

Estructura del repositorio

El repositorio cuenta con las siguientes carpetas principales:

- kaeli/mvp: contiene el código fuente del sistema, incluyendo backend, scraping y lógica de negocio
- diagrams/: almacena los diagramas del sistema
- docs/: incluye toda la documentación que se ha realizado sobre el proyecto
- README.md : archivo de presentacion del proyecto, con descripcion, instalacion, uso, equipo y roadmap

Estrategia de branching

Se han definido dos ramas principales

- main: rama estable destinada a entregas finales (actualmente vacía)
- develop: rama activa donde se concentran todos los avances de los proyectos

Se recomienda implementar ramas features para cada módulo para aislar cada funcionalidad y hacer más fáciles las revisiones del código antes de entregarlo a develop

Commits semánticos

No se realizan commits semánticos y solo descriptivos, como:

- subida documentos kaeli
- subo diagramas

UNIVERSIDAD CATÓLICA DE TEMUCO

Uso de Inteligencia Artificial

Herramientas utilizadas:

- Copilot (Microsoft)
- Chat GPT
- Gemini

Prompts utilizados:

- “Sugiereme que supermercados utilizar para hacer web scraping para kaeli”
- “Como se hace web scraping y cómo lo implemento en mi proyecto”
- “Ayudame a diseñar una página web para kaeli con todo lo esencial basado en los requerimiento funcionales y no funcionales”

Impacto

La IA ayuda a elegir los supermercados, la implementación del web scraping y en la interfaz visual del proyecto kaeli

Ajustes realizados:

Se revisaron y adaptaron las respuestas para alinearlas con el contexto y requisitos del curso

UNIVERSIDAD CATÓLICA DE TEMUCO

Conclusión Técnicas

Aprendizajes centrales:

- La importancia de patrones para una escalabilidad eficiente (Singleton y Factory Method)
- Valor de arquitectura modular
- Relevancia de ciertas métricas para evaluar la calidad de lo que se está haciendo

Limitaciones del diseño:

- Dificultad elevada al intentar hacer web scraping por la seguridad que tienen la pagina de los supermercados
- Asistente IA tiene respuestas genéricas de momento debido a que no tiene aprendizaje automático

Riesgos del sistema:

- Cambios en el HTML de los supermercados elegidos
- Bloqueo por scraping excesivo en las páginas

Mejoras a futuro:

- Integrar IA con Machine Learning (ML)
- Desarrollar una app móvil o web móvil