Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática

Grado en Ingeniería Informática – Ingeniería del Software

Diseño y Pruebas 2

Curso 2023 – 2024

Plantilla Reporte general

Grupo C1.013

Miembros	Información de contacto
David Fuentelsaz Rodríguez	davfuerod@alum.us.es
Miguel Galán Lerate	miggaller@alum.us.es
Antonio Jiménez Ortega	antjimort@alum.us.es
Josué Rodríguez López	josrodlop19@alum.us.es
Óscar Zurita Urpina	osczururp@alum.us.es

Repositorio: https://github.com/DP2-C1-013/Acme-SF-D01-24.1.0

Fecha	Versión
14/2/2024	v1.0

1. Resumen ejecutivo.

En este documento, todo lector podrá indagar sobre el conocimiento de los lectores acerca de las diferentes arquitecturas para sistemas de información web (WIS) que tienen los participantes del proyecto obtenido en las asignaturas previas a Diseño y Pruebas II.

2. Control de versiones.

Nº de revisión	Fecha	Descripción
1	16/02/2024	Desarrollo inicial del *chartering report*

3. Introducción.

La finalidad de este documento no es más que la de informar sobre las diferentes arquitecturas que nosotros, como participantes de este proyecto, conocemos. Principalmente, nos centraremos en los diferentes estilos arquitectónicos que conocemos, además de los patrones de diseño que más hemos empleado.

4. Contenidos.

4.1. Estilos arquitectónicos

En este apartado, se pueden encontrar algunos de los muchos estilos arquitectónicos que los participantes han visto en lo que llevan de carrera universitaria.

4.1.1. Capas

Este estilo arquitectónico ha sido aquel en el que más nos hemos centrado, mayormente en DP1. En este estilo, la funcionalidad del sistema está recogida en capas, donde cada una de ellas ofrece una funcionalidad común, y depende de la funcionalidad que ofrece la capa que se encuentra justo “debajo” de ella.

Cada una de las capas debe ofrecer una interfaz que sirva a la capa adyacente, y cada componente del sistema debe estar contenido en una única capa para mejorar así la cohesión. Normalmente, las aplicaciones cuentan con tres capas, como es el caso de la aplicación que vamos a desarrollar, pero pueden contener más. Es uno de los estilos arquitectónicos más utilizados.

Las tres capas más comunes son, ordenadas de mayor a menor nivel:

- Capa de presentación: se encarga de presentar la información visible al usuario a través de la interfaz gráfica de usuario.

- Capa de lógica de negocio: en ella se contiene toda la funcionalidad de la aplicación, usando datos recogidos de la capa de recursos para mostrarlos mediante la capa de presentación.
- Capa de recursos: contiene todos los datos y modelos de datos a utilizar por la aplicación.

Lo bueno de este estilo es que favorece la cohesión y minimiza el acoplamiento entre componentes, lo que viene siendo útil a la hora de mantener el sistema, y proporciona una buena separación de responsabilidades, pero a veces esto último se puede complicar dependiendo de la aplicación.

4.1.2. Microservicios

En cuanto a este estilo arquitectónico, el sistema en cuestión consta de un conjunto de servicios pequeños y no muy acoplados entre ellos, los cuales ejecutan sus propios procesos. La comunicación entre ellos es, normalmente, mediante HTTP.

Estos microservicios normalmente usan persistencia de datos independientes, es decir, cada servicio puede usar un tipo de base de datos distinto, no necesariamente con los mismos datos. Se interactúa con estos datos a través de APIs RESTful.

Las ventajas de este estilo es que proporciona modularidad, cohesión, separación de responsabilidades y ocultación de información, además de que cada servicio puede desarrollarse de forma independiente. Algunas desventajas son la lenta o no siempre fiable comunicación entre los servicios, además de que gestionar una multitud de servicios puede llegar a ser más complejo de lo normal.

4.1.3. Single Page Application (SPA)

Este estilo es conocido por nosotros, pero no hemos practicado con él. La aplicación se carga completamente de una sola vez en una única página HTML y no se vuelve a cargar en ningún momento; es decir, cuando el usuario usa la aplicación web, la página se actualiza de forma dinámica en vez de cargar otra página del servidor. Esto es posible de forma eficiente gracias a que ahora los clientes son los que renderizan el HTML, y el servidor solo manda JSON (que es menos pesado).

En base a lo anteriormente mencionado, las ventajas son: tiempo de respuesta rápido, y simple de construir y mantener. El principal aspecto negativo es la larga espera inicial de carga de la página.

4.2. Patrones de diseño que conocemos

4.2.1. Patrón Modelo-Vista-Controlador (MVC)

Este patrón de diseño es el que más hemos usado, ya que es fácilmente integrable con el estilo de capas. Estructura la aplicación en tres componentes principales:

- Modelo: representa la información propiamente dicha, incluyendo tanto los datos como tal como los esquemas que les da forma y coherencia (clases).
- Vista: es la presentación con la que el usuario puede interactuar, normalmente mediante una GUI.

- Controlador: responde a los eventos ocurridos en la GUI, haciendo que cambie el modelo y posiblemente la vista.

4.2.2. Front controller

Consiste en un conjunto de controladores o servicios que dependen de un controlador centralizado llamado front controller, que actúa como punto de entrada y salida de datos a la aplicación por medio de peticiones HTTP. Este controlador recibe una petición y la manda al servicio o controlador más adecuado para procesarla.

5. Conclusión.

Mayoritariamente, hemos visto principalmente tres estilos arquitectónicos, capas, microservicios y SPA, de los cuales el que más hemos empleado ha sido el de capas, por lo que tenemos cierto dominio con el mismo. Además, nos hemos centrado en el uso integrado del patrón de diseño MVC, ya que nos ha sido fácil de entender y de diseñar.

6. Bibliografía.

T2 - Architectural design of software applications, DP1.