Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática

D01 – WIS Architecture Report

Grado en Ingeniería Informática – Ingeniería del Software

Diseño y Pruebas II

Curso 2023 – 2024

Fecha	Versión
15/02/2024	v1r1

Grupo de prácticas: C1.002	
Autores	**Correo Corporativo**
José Ramón Baños Botón	josbanbot@alum.us.es
Manuel Palacios Pineda	manpalpin@alum.us.es
Rubén Pérez Garrido	rubpergar@alum.us.es
Javier Ramírez Núñez	javramnun@alum.us.es
Sonia María Rus Morales	sonrusmor@alum.us.es

Link repositorio: https://github.com/Manpalpin/Acme-SF-D01-24.1.0.git

Índice

1. Tabla de versiones

Fecha	Versión	Descripción
14/02/2024	v1r0	Creación del documento
15/02/2024	v1r1	Entrega

2. Introducción

En este documento, vamos a compartir nuestros conocimientos previos sobre la arquitectura de un Sistema de Información Web (WIS), en el contexto de la asignatura Diseño y Pruebas II. Discutiremos diferentes tipos de arquitectura que conocemos y sus aplicaciones. Es importante destacar que esta información es conocida por el grupo antes de comenzar esta asignatura, lo que nos proporciona una base sólida para comprender y aplicar los conceptos discutidos en este curso. Nuestro objetivo es ofrecer una visión general de los conceptos clave y las mejores prácticas que hemos aprendido en proyectos anteriores, enriqueciendo así nuestro proceso de aprendizaje y fomentando la discusión dentro del grupo y con el profesorado.

3. Contenido

Algunas de las arquitecturas conocidas por el grupo previas a la asignatura son:

- **Arquitectura por capas**:

  Estas arquitecturas dividen nuestra aplicación en varias capas cuyas responsabilidades están bien distinguidas. Cada una de estas capas ofrecerán un servicio a sus capas inferiores y superiores y recibirán el servicio de otra capa. El principal beneficio de esta arquitectura es el bajo acoplamiento y alta cohesión debido a la separación distinguida de responsabilidades de cada una de las capas. Pero, uno de los problemas que puede llegar a tener esta arquitectura es que suele ser poco eficiente.

  Una de las divisiones más comunes es: Capa presentación, una capa de reglas de negocios y una capa de acceso a datos.

  Otra ventaja de esta arquitectura es que la mayoría de las capas de la aplicación las podemos tener en servidores propios, ahorrando así que el cliente tenga que gastar recursos en hacer todo el trabajo interno y simplemente interactúe con los datos que nuestras capas le envían a la capa de Cliente o Presentación.

- **Arquitectura MVC**:

  MVC es un patrón que se centra, al igual que la arquitectura por capas, en la separación entre la lógica de negocios y su visualización. Esta separación proporciona una mejor división del trabajo y una mejora de mantenimiento.

Las tres partes en las que se divide son:

- Modelo: define qué datos deben contener la aplicación.
- Vista: define cómo se deben mostrar los datos de la aplicación.
- Controlador: contiene una lógica que actualiza el modelo y/o vista en respuesta a las entradas de los usuarios de la aplicación.

La principal diferencia con el modelo de capas es la forma de como interactúan los elementos entre sí. En el modelo por capas, como hemos comentado, cada una de estas ofrece un servicio a sus capas inferiores y superiores, mientras que en la arquitectura MVC se puede decir que el Controlador manipula el Modelo, el Modelo es representado por la Vista y la Vista es usada para llamar al Controlador.

Otro beneficio, a parte de los ya comentados, es que podemos presentar múltiples vistas utilizando el mismo modelo y controlador. Mientras que la principal desventaja, es que suele requerir de una curva de aprendizaje mucho más alta que con otras arquitecturas más simples.

- **Arquitectura Microservicios**:

Esta estructura tiene un enfoque más expansivo a diferencia que las otras dos arquitecturas mencionadas anteriormente. El principal enfoque es el mismo, tener una separación de responsabilidades bien definida. En este caso lo que hacemos es realizar una “aplicación” a pequeña escala que solo ofrezca un servicio atómico de nuestra aplicación web, obteniendo así múltiples aplicaciones (que llamaremos microservicios) cuya responsabilidad está bien definida y separada.

Una de las mayores ventajas de esta arquitectura es que podemos adaptar el microservicio a los requerimientos de cada uno de estos. Podemos utilizar distintas tecnologías en cada microservicio, dar más rendimiento a algunos microservicios, etc.

Por otra parte, la gran desventaja de esta arquitectura es el manejo, gestión y despliegue de los microservicios, aparte de que la curva de aprendizaje puede ser aún más alta que la arquitectura MVC. Además, esta arquitectura solo tiene sentido si nuestra aplicación es muy grande, ya que no rentaría consumir muchos recursos por cada microservicio cuando la aplicación monolítica puede consumir mucho menos.

4. *Conclusiones*

Como conclusión final, este documento refleja nuestro conocimiento acerca de varias arquitecturas que podemos aplicar al desarrollo de una aplicación web. También hemos visto algunas ventajas y desventajas de estas arquitecturas y en qué casos es conveniente usar cada una de ellas.

5. *Bibliografía*

https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/capas

https://developer.mozilla.org/es/docs/Glossary/MVC

https://www.cloudmasters.es/sabias-que-que-son-los-microservicios-y-por-que-son-el-futuro-en-la-arquitectura-de-software/