

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 2024 – 2025

Fecha	Versión
17/02/2025	V1r1


Grupo: C2.006	
Autores	
Castaño Juan, Cynthia	cyncasjua@alum.us.es
Pereira Campos, Macarena	macpercam@alum.us.es
Pérez Franco, Laura	lauperfra@alum.us.es
Pérez Garrido, Rubén	rubpergar@alum.us.es

Repositorio GITHUB: <https://github.com/rubpergar/Acme-ANS-C2>

	<p>DISEÑO Y PRUEBAS II <Nombre documento></p>
---	--

Índice de contenido

1. Tabla de versiones	2
2. Resumen ejecutivo	2
3. Introducción	3
4. Contenido	3
5. Conclusiones	4
6. Bibliografía	5

	<p style="text-align: center;">DISEÑO Y PRUEBAS II <Nombre documento></p>
---	--

1. Tabla de versiones

Fecha	Versión	Descripción
17/02/2025	V1r0	Creación del documento
17/02/2025	V1r1	Finalización documento

2. Resumen ejecutivo

En este documento, recopilaremos nuestros conocimientos previos sobre la arquitectura WIS antes de cursar esta asignatura, con el objetivo de obtener un resumen de lo que sabemos actualmente y poder comparar estos conocimientos en el futuro, una vez completado el curso.


Nuestra experiencia en este tema proviene principalmente de la realización de diversos proyectos a lo largo de nuestra carrera universitaria, especialmente en asignaturas como IISSI (1 y 2), AISS, DP1, entre otras.

Además, abordaremos los diferentes tipos de arquitectura WIS que conocemos o sobre los cuales hemos oído hablar.

3. Introducción:

En el siguiente informe, se explora la arquitectura WIS (Web Information System), un campo fundamental en el diseño y desarrollo de sistemas de información basados en la web. A lo largo de nuestra carrera universitaria, hemos adquirido conocimientos sobre diferentes arquitecturas que permiten la creación de sistemas eficientes y escalables. Este análisis se centra en las arquitecturas de 2 capas, en capas, el patrón MVC, la arquitectura SPA y los microservicios, con el objetivo de proporcionar una visión general de sus características, usos y ventajas.

El propósito de este informe es no solo documentar lo que sabemos hasta ahora sobre estas arquitecturas, sino también establecer una base que permita evaluar nuestro progreso en el aprendizaje de estos conceptos a medida que avancemos en el curso. A lo largo del documento, se discutirá cómo cada tipo de arquitectura puede mejorar el rendimiento,

	<p style="text-align: center;">DISEÑO Y PRUEBAS II <Nombre documento></p>
---	--

escalabilidad y mantenimiento de los sistemas desarrollados, y cómo tomar decisiones informadas al seleccionar la mejor opción para un determinado entorno de desarrollo.

4. Contenido

Para comenzar, la arquitectura WIS es un enfoque para diseñar y desarrollar sistemas de información basados en la web. Estos sirven para gestionar, procesar y presentar información a través de interfaces accesibles desde navegadores web. Algunos ejemplos son los siguientes:


- **Arquitectura de 2 capas (Cliente-Servidor):**

En este tipo de arquitectura, el cliente interactúa directamente con el servidor de base de datos. Toda la lógica de negocio se maneja desde el lado del cliente o del servidor. Es una estructura simple y directa que es ideal para aplicaciones pequeñas, pero puede resultar limitada a medida que el sistema crece en complejidad.

- **Arquitectura en capas:**

Esta arquitectura se divide en varias capas, cada una con una responsabilidad clara:

- Capa de Presentación también conocida como interfaz de usuario, es la capa encargada de interactuar directamente con el usuario. Se encarga de mostrar la interfaz gráfica o la interfaz de usuario, capturando los datos que el usuario introduce y mostrándole los resultados o respuestas del sistema. Gestiona las solicitudes del usuario y las envía a la capa de Lógica de Negocio que se encargará de procesarlas.
- Capa de Lógica de negocio: contiene la funcionalidad central del sistema y las reglas de negocio. Esta capa es la responsable de procesar las solicitudes recibidas desde la capa de Presentación aplicando lo anterior. Se implementan las validaciones, cálculos y procesos principales del sistema.
- Capa de Persistencia: actúa como intermediaria entre la Lógica de Negocio y el SGBD, gestionando el acceso y almacenamiento de los datos de la aplicación. Su objetivo es realizar operaciones CRUD (crear, leer, actualizar y eliminar) en la base de datos de forma eficiente y segura.
- Sistema de Gestión de Bases de Datos (SGBD): es el sistema donde se almacenan los datos de la aplicación. Además, proporciona funcionalidades

	<p style="text-align: center;">DISEÑO Y PRUEBAS II <Nombre documento></p>
---	--

avanzadas como gestión de transacciones, seguridad y recuperación de datos en caso de fallos, asegurando la integridad y disponibilidad de la información.

- **Patrón MVC (Modelo-Vista-Controlador):**

Este patrón busca separar las responsabilidades dentro de la aplicación para mejorar la organización del flujo de trabajo y facilitar el mantenimiento. Se estructura de la siguiente manera:

- Modelo: representa los datos y la lógica de negocio de la aplicación. Contiene la información que la aplicación maneja y las operaciones necesarias para modificarla.
- Vista: es la interfaz de usuario y se encarga de mostrar los datos al usuario. Además, representa la forma en que los datos se presentan,
- Controlador: actúa como intermediario entre el modelo y la vista, gestionando las entradas del usuario y actualizando el modelo y la vista.

- **Arquitectura SPA (Single Page Application):**


La arquitectura SPA se caracteriza por cargar la aplicación una sola vez y gestionar todas las interacciones posteriores mediante llamadas AJAX. Esto permite actualizar únicamente las partes de la página que lo necesiten, sin recargar la página completa. Este enfoque mejora significativamente la experiencia del usuario, proporcionando una navegación fluida y rápida, además de reducir la carga en el servidor.

- **Arquitectura Microservicios:**

En esta arquitectura, se opta por dividir una aplicación monolítica en varios servicios pequeños, independientes y autónomos, llamados **microservicios**. Cada microservicio es responsable de una funcionalidad específica de la aplicación, lo que permite escalabilidad, flexibilidad y facilidad de mantenimiento. Los microservicios se comunican entre sí a través de APIs bien definidas, lo que permite desarrollar, desplegar y escalar cada parte del sistema de manera independiente.

5. Conclusiones

Para concluir, este documento refleja nuestros conocimientos sobre la arquitectura WIS, la cual ofrece diversos enfoques para el diseño y desarrollo de sistemas de información basados en

	<p style="text-align: center;">DISEÑO Y PRUEBAS II <Nombre documento></p>
---	--

la web. Cada tipo de arquitectura mencionado tiene sus particularidades que las hacen similares, pero, al mismo tiempo, distintas en su aplicación y funcionamiento. La elección de una u otra dependerá de las necesidades específicas del sistema que queramos implementar, así como de la complejidad y los recursos disponibles. Todas presentan ventajas e inconvenientes, lo que requiere una evaluación cuidadosa para seleccionar la más adecuada en cada contexto.

Algunas arquitecturas son más sencillas, como la arquitectura de 2 capas, que se basa únicamente en la interacción directa entre el cliente y el servidor. Aunque es simple, sigue siendo útil para sistemas poco complejos. Por otro lado, la arquitectura en capas proporciona una separación clara de responsabilidades, lo que favorece la escalabilidad y el mantenimiento del sistema.

El patrón MVC destaca por organizar de manera eficiente el flujo de trabajo, separando la lógica de negocio, la presentación y el control de la interacción del usuario, lo que facilita el mantenimiento y la reutilización de código. La arquitectura SPA mejora la experiencia del usuario al cargar la aplicación una sola vez, ofreciendo una navegación fluida. Finalmente, los microservicios aportan escalabilidad y flexibilidad al dividir la aplicación en servicios pequeños e independientes, lo que permite un despliegue y mantenimiento más ágil.

6. Bibliografía

- **Sierra, C. (1995).** *Arquitectura WIS*. Universidad Nacional de Colombia. Recuperado de <https://repositorio.unal.edu.co/bitstream/handle/unal/57768/cesarsierrawis.1995.pdf?sequence=1&isAllowed=y>