UNIVERSIDAD DE CASTILLA-LA MANCHA

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

Implementación de Aplicación Web y Desarrollo de Técnicas de Visualización para la Gestión y el Diagnóstico de Patologías

Sergio García Muñoz

Junio, 2024

UNIVERSIDAD DE CASTILLA-LA MANCHA

TRABAJO FIN DE GRADO

Departamento de Tecnologías y Sistemas de Información

Tecnología Específica de Web

Implementación de Aplicación Web y Desarrollo de Técnicas de Visualización para la Gestión y el Diagnóstico de Patologías

Autor: Sergio García Muñoz

Tutor Académico: Félix Albertos Marco

Cotutor Académico: Juan Enrique Garrido Navarro

Junio , 2024

*D*edicado a mi familia y a todos
aquellos ...

Declaración de Autoría

Yo, …… con DNI ……, declaro que soy el único autor del trabajo fin de grado titulado "……" y que el citado trabajo no infringe las leyes en vigor sobre propiedad intelectual y que todo el material no original contenido en dicho trabajo está apropiadamente atribuido a sus legítimos autores.

Talavera de la Reina, a…..

Fdo: ……

Resumen

El proyecto DIPAMIA (DIagnóstico de PAtologías a través del Análisis del Movimiento utilizando Inteligencia Artificial) se centra en el análisis del movimiento para diagnosticar patologías como, por ejemplo, la depresión. Uno de los elementos clave de este proyecto es la plataforma Web encargada de recoger, gestionar y mostrar la información relativa al proyecto. Esta aplicación Web será utilizada tanto por especialistas sanitarios, especialistas en movimiento, así como personal técnico que trabaje con el procesado de los datos. En última instancia, los pacientes también utilizan la plataforma para acceder a la información relativa a su intervención. En este contexto, el desarrollo de la aplicación Web es el objetivo de este Trabajo Final de Grado. Se realizará un estudio previo de las tecnologías más adecuadas para el desarrollo del proyecto. También, de cara a implementar una metodología centrada en el usuario, se realizarán entrevistas a los stakeholders con el fin de obtener la información necesaria para el desarrollo del proyecto. A continuación se analizarán las técnicas de visualización más adecuadas según los pasos previos para implementar la aplicación Web. Finalmente, se realizarán estudios de usabilidad para mejorar el uso del sistema.

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Agradecimientos

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Índice general

1. Introducción **1**

1.1. Contexto y Motivación 1

1.2. Objetivos 2

1.3. Competencias de la Intensificación 2

1.4. Estructura del Documento 3

2. Estado del Arte **5**

2.1. Metodologías de Desarrollo Software 5

2.1.1. Ciclo de Vida del Desarrollo del Software 5

2.1.2. Metodologías de Desarrollo 6

2.1.3. Metodologías Ágiles 7

2.2. Tecnología Web 8

2.2.1. BackEnd 8

2.2.2. FrontEnd 11

2.2.3. Stacks Tecnológicos 14

2.3. Técnicas de Visualización 15

2.3.1. Tipos de Visualización 16

2.3.2. Mejores Prácticas 17

2.3.3. Principios de Gestalt 18

2.4. Interacción Persona-Ordenador 19

2.4.1. Interacción Persona-Ordenador 19

2.4.2. Usabilidad 19

2.4.3. Experiencia de Usuario 21

3. Propuesta de Solución **23**

3.1. Propuesta de Solución 23

3.1.1. Tecnología SPA 23

3.1.2. Arquitectura MVC 24

3.1.3. Usabilidad . . . 25
3.2. Metodología de Desarrollo . . . 26
3.2.1. Justificación . . . 26
3.2.2. Scrum . . . 27
3.2.3. Adaptación . . . 30
3.3. Herramientas . . . 30
3.3.1. Gestión de Proyecto . . . 30
3.3.2. Documentación . . . 32
3.3.3. Desarrollo . . . 32

4. Resultado **35**

5. Evaluación **37**

6. Conclusiones **39**

Bibliografía **41**

Anexo A. Resumen de Sprints **47**
A.1. Fase Inicial . . . 47
A.1.1. Primera reunión . . . 47
A.1.2. Segunda reunión . . . 48
A.1.3. Tercera reunión . . . 48
A.2. Sprint 1 . . . 48
A.2.1. Planificación del Sprint . . . 48
A.2.2. Desarrollo del Sprint . . . 49
A.2.3. Revisión del Sprint . . . 49
A.2.4. Retrospectiva del Sprint . . . 49
A.3. Sprint 2 . . . 50
A.3.1. Planificación del Sprint . . . 50
A.3.2. Desarrollo del Sprint . . . 50
A.3.3. Revisión del Sprint . . . 51
A.3.4. Retrospectiva del Sprint . . . 51
A.4. Sprint 3 . . . 51
A.4.1. Planificación del Sprint . . . 51
A.4.2. Desarrollo del Sprint . . . 52
A.4.3. Revisión del Sprint . . . 52
A.4.4. Retrospectiva del Sprint . . . 52

A.5. Sprint 4 . . . 53
A.5.1. Planificación del Sprint . . . 53
A.5.2. Desarrollo de Sprint . . . 54
A.5.3. Revisión Sprint . . . 55
A.5.4. Retrospectiva del Sprint . . . 55
A.6. Sprint 5 . . . 55
A.6.1. Planificación del Sprint . . . 56
A.6.2. Desarrollo de Sprint . . . 56
A.6.3. Revisión Sprint . . . 58
A.6.4. Retrospectiva del Sprint . . . 58
A.7. Sprint 6 . . . 58
A.7.1. Planificación del Sprint . . . 59
A.7.2. Desarrollo de Sprint . . . 60
A.7.3. Revisión Sprint . . . 60
A.7.4. Retrospectiva del Sprint . . . 60
A.8. Sprint 7 . . . 60
A.8.1. Planificación del Sprint . . . 60
A.8.2. Desarrollo de Sprint . . . 61
A.8.3. Revisión Sprint . . . 61
A.8.4. Retrospectiva del Sprint . . . 61

Índice de figuras

2.1. Despliegue con docker vs con maquinas virtuales [18] 11
2.2. TypeScript vs JavaScript 12
2.3. Principios de Gestalt [41] 18
2.4. Flujo de la disciplina de la Interacción Persona-Ordenador [26] 20

3.1. Flujo de una SPA 24
3.2. Arquitectura MVC 25
3.3. Flujo de la metodología Scrum [60] 29
3.4. Plantilla para la creación del BC 31
3.5. Arquitectura tecnológica 34

A.1. BC del Sprint 1 49
A.2. BC del Sprint 2 51
A.3. BC del Sprint 3 53
A.4. BC del Sprint 4 55
A.5. BC del Sprint 5 58

Índice de Tablas

2.1. Tecnología en relación al capítulo 2.2 15

A.1. PB del Sprint 1 48
A.2. SB del Sprint 1 49
A.3. PB del Sprint 2 50
A.4. SB del Sprint 2 50
A.5. PB del Sprint 3 52
A.6. SB del Sprint 3 52
A.7. PB del Sprint 4 54
A.8. SB del Sprint 4 54
A.9. PB del Sprint 5 56
A.10.SB del Sprint 5 56
A.11.PB del Sprint 6 59
A.12.SB del Sprint 6 59

Índice de Listados

A.1. Archivo makefile antes del cambio 57
A.2. Archivo makefile después del cambio 57

Acrónimos

TFG Trabajo Final de Grado

NoSQL Not Only Structured Query Language (No Solo Lenguaje de Consulta Estructurada)

HTML HyperText Markup Language (Lenguaje de Marcas de Hypertexto)

CSS Cascading Style Sheets (Hojas de Estilo en Cascada)

DOM Document Object Model (Modelo de Objeto de Documento)

SPA Single Page Application (Aplicaciones de una Sola Página)

Wasm WebAssembly

PWA Progressive Web Applications (Aplicaciones Web Progresivas)

HTTP Hypertext Transfer Protocol (Protocolo de Transferencia de Hipertexto)

SO Sistema Operativo

BD Base de Datos

IPO Interacción Persona-Ordenador

SIGCHI Special Interest Group on Computer-Human Interaction (Grupo de Interés Especial sobre la Interacción Persona-Ordenador)

ISO International Organization for Standardization (Organización Internacional de Normalización)

MVC Model-View-Controller (Modelo-Vista-Controlador)

HU Historia de Usuario

PB Product Backlog

SB Sprint Backlog

BC Burndown Chart

Capítulo 1

Introducción

Este proyecto tiene como meta principal la creación de una aplicación Web dedicada a la gestión y diagnóstico de patologías, con un enfoque especial en la fibromialgia. La aplicación busca presentar la información de manera efectiva mediante el uso de técnicas de visualización.

La implementación de esta aplicación tiene como objetivo principal centrarse en el FrontEnd, destacando la importancia de la capa visual y funcional; y abstrayéndose de la capa del servidor. Esta decisión busca asegurar que la interfaz ofrezca una representación clara y efectiva de la información médica, facilitando el análisis y diagnóstico de diversas patologías.

Antes de profundizar en los detalles del proyecto, es crucial presentar el contexto y la motivación detrás de este, así como los objetivos, las competencias de la intensificación y la estructura del documento.

1.1. Contexto y Motivación

Como estudiante de Ingeniería Informática, mi experiencia previa ha abarcado diversas herramientas y lenguajes de programación, pero he sentido un particular interés por el desarrollo interactivo en el FrontEnd, un área en la que he tenido menos oportunidades de trabajar hasta este último año de la carrera.

La motivación para este proyecto surge no solo de mi interés en el desarrollo web, sino también de la relevancia social y médica que aborda. Después de todo, la fibromialgia es una enfermedad crónica, la cual puede tener un impacto significativo en la calidad de vida de millones de personas en todo el mundo. Los síntomas debilitantes pueden afectar la capacidad de llevar a cabo actividades diarias, influenciando también la salud mental y emocional de quienes la padecen.

Es en este contexto, donde se origina el proyecto DiPAMIA (Sistema de Gestión y Diagnóstico de

Patologías basado en el Análisis de Movimiento a través de Inteligencia Artificial). Como indica su nombre, la premisa fundamental de este sistema es utilizar la inteligencia artificial para detectar patologías mediante el análisis del movimiento, ofreciendo una herramienta eficaz y no invasiva para el diagnóstico y seguimiento de condiciones de salud mental.

Para que este sistema sea accesible y útil, es esencial desarrollar una aplicación Web eficiente y amigable. Aquí es donde entra en juego la tecnología Web del lado del cliente y por consiguiente el desarrollo de este proyecto. Después de todo, la interfaz de usuario desempeña un papel crucial en el éxito de cualquier herramienta tecnológica, pues actúa como la ventana a través de la cual los usuarios interactúan con la información y funcionalidades proporcionadas. Además, la visualización de datos es esencial para presentar de manera clara y comprensible los resultados del análisis de movimiento, permitiendo a profesionales de la salud y pacientes tomar mejores decisiones.

1.2. Objetivos

El objetivo principal es la **implementación de una aplicación Web y el desarrollo de técnicas de visualización para la gestión y el diagnóstico de patologías**. Para alcanzar este objetivo se proponen los siguientes subobjetivos:

- **O1:** Estudio de las tecnologías Web
- **O2:** Estudio de técnicas de visualización en la Web
- **O3:** Obtención de las necesidades de los “stakeholders” del sistema
- **O4:** Desarrollo de la Aplicación Web de Soporte
- **O5:** Propuesta e implementación de técnicas de visualización
- **O6:** Evaluación de técnicas de visualización
- **O7:** Estudio de usabilidad y validación de la aplicación

1.3. Competencias de la Intensificación

Además de los objetivos mencionados previamente, también se pretende hacer uso de las competencias de la intensificación cursada, en este caso de “Ingeniería de Informática de Sistemas de Información”. Las siguientes son algunas de estas competencias:

- **BA5:** Conocimiento de la estructura, organización, funcionamiento e interconexión de los sistemas informáticos, los fundamentos de su programación, y su aplicación para la resolución de problemas propios de la ingeniería [67].

- **CO8:** Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados [67].
- **CO13:** Conocimiento y aplicación de las herramientas necesarias para el almacenamiento, procesamiento y acceso a los Sistemas de información, incluidos los basados en Web [67].
- **CO16:** Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería de software [67].

1.4. Estructura del Documento

Este documento se organiza de la siguiente manera para proporcionar una visión completa y estructurada del proyecto:

1. **Introducción:** Se brinda una visión inicial del proyecto, donde se presenta la motivación que impulsó su desarrollo, así como los objetivos y competencias del mismo.
2. **Estado del arte:** Se lleva a cabo un análisis exhaustivo de las diferentes metodologías de desarrollo, tecnologías web y técnicas de visualización relevantes en el campo.
3. **Propuesta de solución:** Se presenta la propuesta para la creación de la aplicación Web, detallando la metodología a seguir, las herramientas a utilizar en su desarrollo, así como las técnicas de visualización.
4. **Resultado:** Se muestra el producto final de la creación de la aplicación Web, acompañado de distintas capturas de pantalla que muestran su funcionalidad y diseño.
5. **Evaluación:** Se realiza una evaluación crítica del resultado final de la aplicación Web, analizando su usabilidad y eficacia.
6. **Conclusiones:** Se presentan las conclusiones finales del proyecto, incluyendo la verificación de sí se han alcanzado los objetivos establecidos inicialmente.

Finalmente, se encuentra la **Bibliografía**, donde se enumeran las diversas fuentes de información consultadas y utilizadas durante la realización del proyecto. Además, se incluyen los respectivos **Anexos** que complementan y enriquecen la información presentada en la memoria final.

Capítulo 2

Estado del Arte

En el contexto del desarrollo de una aplicación Web, es esencial considerar diversos factores, incluyendo la metodología de desarrollo, las herramientas y tecnologías disponibles y la usabilidad de la aplicación. A continuación, es realizado un análisis que abarca todas estas áreas con el propósito de determinar las mejoras prácticas y enfoques para implementar durante el proyecto.

2.1. Metodologías de Desarrollo Software

Para que un software, como es una aplicación Web, llegue a producción debe pasar por un proceso de desarrollo de software, conocido también como “ciclo de vida del desarrollo del software”. A su vez, para organizar este ciclo es necesario utilizar cierta metodología de desarrollo. En este apartado es realizado un análisis de las distintas fases de dicho proceso, además de las metodologías más destacadas.

2.1.1. Ciclo de Vida del Desarrollo del Software

El ciclo de vida del desarrollo de software es un proceso estructurado que abarca desde la concepción de un proyecto hasta su despliegue y mantenimiento. A lo largo de este ciclo, se llevan a cabo una serie de fases y actividades. Cada fase del ciclo de vida del desarrollo de software tiene sus propios objetivos y entregables específicos. Este normalmente está dividido en las siguientes fases, tal y como expone Clark en [15].

- **Planificación y análisis:** Se elabora un plan que describe en detalle cómo se dará vida al proyecto desde la idea hasta la realización.
- **Definición de requisitos:** Se recopilan los diferentes requisitos para la aplicación final.
- **Diseño:** Se elabora un documento de diseño de software que incluye el diseño del siste-

ma, los lenguajes de programación, las plantillas, la plataforma a utilizar y las medidas de seguridad de la aplicación.

- **Desarrollo:** Se transforman los requisitos en código para desarrollar la propia aplicación.
- **Pruebas:** Se realizan pruebas de validación para asegurar que esté funcionando correctamente y haga lo que debe hacer.
- **Despliegue:** Se entrega al usuario final.
- **Mantenimiento:** Se arreglan los diferentes errores que encuentren los usuarios, teniendo que volver a la primera fase del ciclo si fuera necesario.

Una vez se conoce el ciclo de desarrollo, es esencial saber cómo organizar el mismo. Para ello, en el siguiente punto se analizan las diferentes metodologías de desarrollo.

2.1.2. Metodologías de Desarrollo

Para organizar cada una de las fases, como se ha mencionado previamente, es necesario implementar una metodología de desarrollo, con el fin de aumentar la productividad y la calidad del software. A continuación, se detallan algunas de las metodologías más destacadas, tal y como expone Alona en [3].

- **Cascada:** Se trata de una de las metodologías más antiguas. En este caso, las fases de desarrollo se realizan de una manera secuencial, es decir, una fase comienza solo cuando termina la fase anterior. Aunque esto hace que la estructura sea más clara y fácil de entender, tiene la desventaja de ser poco flexible y adaptable a los cambios de los requisitos, que tan a menudo se ve en un desarrollo de software.
- **Iterativa:** Esta metodología divide el proyecto en pequeñas partes, y cada parte es desarrollada, probada e implementada de manera iterativa. Cada una de estas iteraciones agrega funcionalidad al software.
- **En V:** En esta metodología se relaciona cada fase de desarrollo con su fase de pruebas correspondiente. De esta manera, pone un gran énfasis en la validación temprana de requisitos y la calidad del software. No obstante, similar a la metodología en cascada, su enfoque lineal puede resultar inflexible frente a cambios en los requisitos.
- **Espiral:** Se trata de una metodología que mezcla los elementos de la planificación en cascada con la flexibilidad de las metodologías ágiles. Utiliza ciclos repetitivos incluyendo planificación, evaluación de riesgos, ingeniería y evaluación del cliente. Aunque es bastante efectivo en la gestión de riesgos, puede acabar resultando muy complejo de gestionar, además de requerir más tiempo y recursos que otras metodologías.
- **Big bang:** Esta metodología tiene un enfoque único en el que los desarrolladores se lanzan directamente a la codificación sin mucha planificación. Esto significa que los requisitos

se implementan a medida que aparecen, es decir, sin ningún tipo de hoja de ruta clara. Sin embargo, a la hora de necesitar realizar algún cambio, puede llegar a requerir una renovación completa del software.

- **Ágil:** Esta metodología organiza las fases del ciclo de vida en varios ciclos de desarrollo, de forma que el equipo realiza pequeños cambios de software incrementales en cada uno de estos ciclos. Esto lo hace ideal para proyectos de desarrollo de software que requieran flexibilidad y capacidad de adaptarse a los cambios con el tiempo.

Al final, la elección de la metodología de desarrollo de software es esencial para el éxito del proyecto y debe ajustarse a las particularidades del mismo. Cada enfoque tiene sus propias fortalezas y debilidades. Sin embargo, la metodología ágil puede ser la que más destaque para el desarrollo de software debido a su agilidad, ciclos cortos de desarrollo y capacidad para adaptarse a cambios en los requisitos. En el siguiente punto, se analizará algunas de las metodologías ágiles más utilizadas en la actualidad.

2.1.3. Metodologías Ágiles

Como ya se ha mencionado las metodologías ágiles son las que más destacan en la actualidad debido a su flexibilidad para adaptarse a los cambios. A continuación, son analizadas tres de las más usadas hoy en día, tal y como expone Espírito en [20]:

- **Extreme programming:** Consiste en una metodología ágil que se centra en la mejora continua y la respuesta rápida a los cambios en los requisitos. Para ello, pone énfasis en la retroalimentación constante y la comunicación abierta, destacando prácticas como la programación en parejas, pruebas unitarias continuas y entregas frecuentes.
- **Kanban:** Con origen en la manufactura, esta metodología ágil se ha conseguido aplicar con éxito al desarrollo de software. Se basa principalmente en la visualización del flujo de trabajo mediante el uso de tableros Kanban, representando cada tarea mediante tarjetas que se mueven a través de columnas que representan diferentes estados del proceso. A pesar de la mejora continua y flexibilidad que ofrece, se puede notar la ausencia de una estructura para roles y eventos, lo que dificulta la gestión de proyectos más grandes.
- **Scrum:** Muy conocido y usado en la actualidad. En Scrum, los ciclos de desarrollo se llaman “sprints”, que generalmente duran entre dos y cuatro semanas. Además, utiliza roles definidos claramente como el Scrum Master, el Product Owner y el equipo de desarrollo. También, se enfoca en la entrega iterativa de incrementos de software y gracias a su énfasis en la comunicación constante y adaptabilidad al cambio se ha convertido en una elección popular en el desarrollo de proyectos software, como es el caso de la implementación de una aplicación Web.

Nuevamente, la elección de la metodología dependerá de las características del proyecto. Siendo en este caso, Scrum la que más destaca entre todas ellas. Sin embargo, una vez se conoce el proceso y las diferentes metodologías del desarrollo de software, es esencial conocer la tecnología Web que se puede utilizar para el desarrollo de dicho software, la cual se detalla en el siguiente punto.

2.2. Tecnología Web

En el proceso de implementación de una aplicación Web, resulta esencial no solo seguir una metodología específica, sino también utilizar la diversa tecnología Web disponible. Este campo de estudio abarca un extenso conjunto de herramientas, estándares y prácticas que han sido utilizados en el desarrollo, implementación y mantenimiento de aplicaciones y sitios Web. Desde los aspectos del lado del cliente hasta los del servidor, este panorama tecnológico tiene como objetivo primordial mejorar la experiencia del usuario y optimizar la eficiencia en el desarrollo. A continuación, se profundiza en todas estas áreas.

2.2.1. BackEnd

En el ámbito del desarrollo BackEnd, se puede encontrar un ecosistema diverso de lenguajes de programación, cada uno acompañado de sus propios frameworks. Este panorama se enriquece aún más con la presencia crucial de bases de datos y la adopción generalizada de contenedores. En este análisis, se explora detenidamente cada una de estas áreas tecnológicas.

Por un lado, los lenguajes de programación, los cuales desempeñan un papel significativo, definiendo la estructura y el rendimiento de las aplicaciones. Se destacan principalmente cuatro lenguajes, tal y como expone Osadchuk en [55]: JavaScript, Python, Ruby y PHP.

- **JavaScript:** Se trata de un lenguaje de programación versátil y ampliamente utilizado que funciona tanto en el lado del cliente como en el lado del servidor. En el contexto del desarrollo del lado del servidor, se emplea en el entorno de ejecución **Node.js [16]**, que utiliza el motor V8 de Google Chrome. JavaScript es destacado por su capacidad para manejar operaciones de entrada/salida de manera eficiente y su naturaleza asíncrona, lo que lo hace ideal para aplicaciones escalables y basadas en eventos, como aplicaciones Web en tiempo real.
- **Python [61]:** Consiste en un lenguaje versátil y de alto nivel que enfatiza la legibilidad del código y la productividad del programador. Con una sintaxis clara y concisa, Python es utilizado en una amplia gama de aplicaciones, desde desarrollo Web hasta inteligencia artificial y análisis de datos.

- **Ruby [40]:** Es conocido por su elegancia y simplicidad, priorizando la productividad del desarrollador. Aunque es menos ubicuo que otros lenguajes, ha ganado popularidad gracias a su enfoque en la facilidad de uso y la creación de código conciso y expresivo.
- **PHP:** Se trata de un lenguaje de scripting especialmente diseñado para el desarrollo Web. Ampliamente utilizado para construir aplicaciones Web dinámicas, PHP se integra fácilmente con HTML y se ejecuta en el lado del servidor.

Estos lenguajes, aunque poderosos por sí mismos, se ven potenciados cuando se combinan con frameworks específicos. Cada uno de los mencionados tiene su conjunto de frameworks que agilizan y estructuran el proceso de desarrollo. A continuación, se explora un framework representativo para cada uno de estos lenguajes respectivamente, tal y como expone nuevamente Osadchuk en [55]: Express, Django, Ruby on Rails y Laravel.

- **Express.js [53]:** Es un framework para Node.js que simplifica el desarrollo de aplicaciones Web y APIs. Con un enfoque minimalista, permite la creación rápida de servidores y rutas, facilitando la construcción de aplicaciones robustas con JavaScript del lado del servidor.
- **Django [17]:** Es un framework de alto nivel para Python, diseñado para maximizar la eficiencia y la reutilización del código. Con un conjunto integrado de herramientas y una arquitectura basada en el patrón de diseño Modelo-Vista-Controlador (MVC), Django simplifica la creación de aplicaciones Web complejas al proporcionar una estructura organizativa y características como la administración automática de bases de datos.
- **Ruby on Rails [32]:** Es un framework que sigue el principio de convención sobre configuración para el desarrollo rápido de aplicaciones Web en Ruby. Facilita la creación de aplicaciones mediante la automatización de tareas repetitivas y la adopción de convenciones predefinidas. Rails proporciona un entorno coherente que acelera el proceso de desarrollo y favorece la escritura de código limpio y conciso.
- **Laravel [56]:** Es un framework elegante y completo para PHP que aborda diversos aspectos del desarrollo Web. Ofrece una sintaxis expresiva, una gestión eficiente de bases de datos, y una amplia gama de herramientas para tareas comunes. Laravel fomenta la creación de aplicaciones seguras y modernas, con un énfasis en la legibilidad y mantenimiento del código.

Por otro lado, en el contexto del desarrollo del lado del servidor es esencial considerar no solo los lenguajes de programación y sus frameworks asociados, sino también las bases de datos, los cuales son elementos fundamentales para la persistencia y gestión de datos. Para una comprensión más detallada, se examinan tres bases de datos específicas, tal y como expone Ramotion en [59]: MySQL, MongoDB y Neo4j.

- **MySQL [54]:** Se trata de una BD relacional, reconocida por su fiabilidad y consistencia en

la gestión de datos estructurados. Ha sido una opción de confianza en el desarrollo Web, proporcionando un entorno robusto para aplicaciones que dependen de una estructura de datos clara y relaciones definidas.

- **MongoDB [48]:** Destaca como una BD no relacional orientada a documentos, ofreciendo flexibilidad en el esquema y una capacidad eficaz para manejar grandes volúmenes de datos no estructurados. Esta característica la convierte en una elección popular para aplicaciones que requieren escalabilidad y adaptabilidad a cambios en la estructura de datos.
- **Neo4j [51]:** Como una BD de grafos, está diseñada para almacenar y procesar datos en forma de nodos y relaciones. Esto la convierte en una opción adecuada para aplicaciones que necesiten modelar y analizar redes complejas, como redes sociales, sistemas de recomendación y análisis de relaciones en datos interconectados.

Finalmente, en el lado del servidor, no se debe pasar por alto la importancia de los contenedores, ya que optimizan la implementación, escalabilidad y gestión de aplicaciones. A continuación, se analizan dos tecnologías destacadas, tal y como expone Baryshevskiy en [6]: Docker y Kubernetes.

- **Docker [35]:** Se trata de plataforma de contenedores que posibilita el empaquetado, la distribución y la ejecución coherente de aplicaciones en diversos entornos. Su enfoque revolucionario ha transformado la consistencia y portabilidad en el despliegue de software, permitiendo una gestión eficaz de dependencias y configuraciones. En la Figura 2.1 se puede observar la diferencia entre un despliegue de contenedores que comparten el SO y un despliegue de maquinas virtuales que hacen una copia completa del SO cada una.
- **Kubernetes [25]:** Consiste en un sistema de orquestación de contenedores, simplificando la administración, escalabilidad y despliegue de aplicaciones contenidas en entornos distribuidos. Su capacidad para coordinar eficientemente la ejecución de contenedores a lo largo de múltiples nodos ha consolidado su posición como una herramienta fundamental en el despliegue de infraestructuras escalables y resilientes.

En resumen, la tecnología de BackEnd engloba un conjunto diverso de herramientas y tecnologías esenciales para el desarrollo y funcionamiento de aplicaciones Web y servicios. La elección de tecnologías específicas depende de factores como los requisitos del proyecto, la escala y complejidad de la aplicación, así como las preferencias del equipo de desarrollo. La combinación adecuada de estos elementos contribuye a la creación de sistemas BackEnd robustos, eficientes y escalables. Sin embargo, una vez se conoce el lado del servidor, es vital conocer el otro lado conocido como el FrontEnd, el cual se enfoca más en el cliente. En el siguiente punto, se detalla la diferente tecnología Web dentro de dicha área.

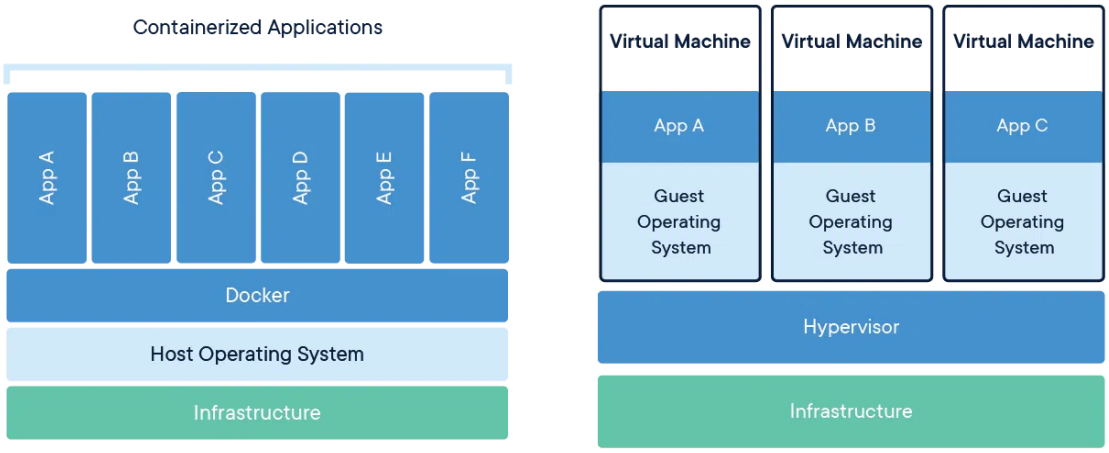


Figura 2.1: Despliegue con docker vs con maquinas virtuales [18]

2.2.2. FrontEnd

En el ámbito del Frontend, se pueden encontrar diferentes lenguajes tanto de marcado como de programación. Además, destacan los diferentes frameworks, así como de las diferentes librerías de visualización de datos que desempeñan un papel crucial en la construcción de interfaces de usuario atractivas. Tampoco se puede pasar por alto, la creciente popularidad del WebAssembly y las Progressive Web Apps, innovaciones que están transformando la experiencia del usuario dentro del ámbito del FrontEnd. A continuación, se realizará un análisis de estas tecnologías para comprender su impacto y aplicaciones específicas.

Por un lado, los lenguajes de marcado como de programación fundamentales que encontramos en el desarrollo del FrontEnd son la tríada formada de HTML, CSS y JavaScript, este último extendido mediante TypeScript. A continuación, se realiza una breve explicación de cada uno de estos lenguajes.

- **HTML:** Lenguaje de marcado estándar utilizado para estructurar el contenido de las páginas Web. Define la jerarquía y organización de los elementos en una página.
- **CSS:** Lenguaje de estilo que complementa a HTML. Permite definir el diseño, la presentación y la apariencia visual de los elementos HTML, proporcionando control sobre colores, tipografías y disposición.
- **JavaScript:** Se trata del lenguaje de programación del lado del cliente que permite la creación de interactividad en las páginas Web. Es esencial para manipular el DOM y responder a eventos del usuario.
- **TypeScript [42]:** Es un lenguaje de programación fuertemente tipado que se basa en JavaScript. A diferencia de este último, TypeScript añade un sistema de tipos estático, pro-

porcionando beneficios como la detección temprana de errores y una mejor mantenibilidad en proyectos a gran escala. Esta diferencia se puede ver ilustrada en la Figura 2.2.

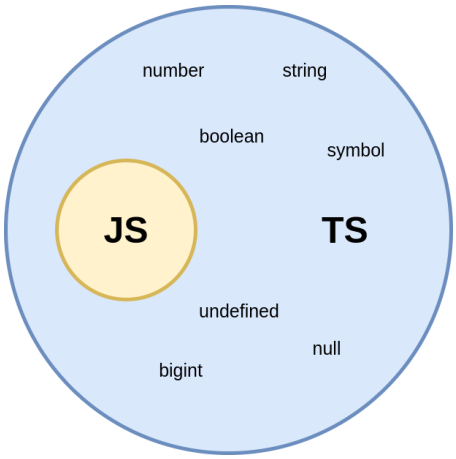


Figura 2.2: TypeScript vs JavaScript

Además de estos lenguajes fundamentales, los frameworks de SPA han revolucionado la forma en que se desarrollan las interfaces de usuario. Estos frameworks permiten la creación de experiencias más dinámicas al cargar solo los componentes necesarios a diferencia de recargar toda la página como lo haría una aplicación Web tradicional. Aquí se presentan cuatro de dichos frameworks, tal y como expone Baryshevskiy en [6]: React, Angular, Vue y Svelte.

- **React [22]:** Desarrollado por *Facebook*, React es una biblioteca de JavaScript que se centra en la construcción de interfaces de usuario reactivas y eficientes. Su enfoque en la creación de componentes modulares y su capacidad para gestionar el estado de manera efectiva lo convierten en una elección popular para aplicaciones dinámicas y escalables.
- **Angular [24]:** Respaldado por *Google*, es un framework completo de desarrollo que abarca desde la creación de componentes hasta la gestión del estado y el enrutamiento. Su estructura robusta y su integración con TypeScript ofrecen un enfoque amplio para el desarrollo de SPAs, siendo especialmente adecuado para proyectos empresariales complejos.
- **Vue.js [69]:** Se trata de un framework progresivo de JavaScript que destaca por su simplicidad y flexibilidad. Su diseño modular facilita la integración gradual en proyectos existentes, y su curva de aprendizaje suave lo convierte en una opción popular para desarrolladores que buscan una alternativa accesible y potente.
- **Svelte [29]:** Adopta un enfoque diferente al trasladar gran parte del trabajo de construcción a tiempo de compilación. Esto resulta en aplicaciones más livianas y rápidas en tiempo de ejecución. Su sintaxis sencilla y su rendimiento eficiente lo hacen atractivo para desarrolladores que buscan una alternativa innovadora y eficaz en la construcción de SPAs.

Por otro lado, diversas librerías han surgido para facilitar la creación de visualizaciones atractivas e interactivas, después de todo la visualización de datos desempeña un papel fundamental en la comprensión y comunicación efectiva de información en entornos Web. A continuación, son presentadas algunas de las librerías más destacadas, tal y como expone Majorek en [39]:

- **D3.js [8]:** Destaca como una herramienta poderosa para la manipulación basada en datos en documentos . Su capacidad para crear visualizaciones altamente personalizables e interactivas lo convierte en una opción popular para desarrolladores que buscan flexibilidad en la representación gráfica de datos complejos.
- **Three.js [9]:** Aunque inicialmente diseñada para gráficos 3D, Three.js puede ser aprovechada para visualizaciones de datos tridimensionales impactantes. Su capacidad para crear experiencias visuales inmersivas lo convierte en una elección interesante para proyectos ambiciosos.
- **Chart.js [19]:** Aporta una solución simple y fácil de usar, ofreciendo una variedad de gráficos, como barras, líneas y radar. Esta librería en JavaScript permite a los desarrolladores incorporar rápidamente visualizaciones atractivas en sus aplicaciones Web.
- **ECharts [4]:** Ofrece una amplia variedad de gráficos, incluyendo líneas, barras, dispersión y mapas, entre otros. La capacidad de Echarts para manejar grandes conjuntos de datos y su enfoque en la interactividad hacen que sea una elección sólida para proyectos que requieren visualizaciones dinámicas y atractivas.
- **Highcharts [34]:** Con una amplia gama de opciones de personalización, Highcharts simplifica la creación de gráficos interactivos. Esta librería en JavaScript es adecuada para proyectos que buscan una solución robusta y fácil de implementar.
- **React-Vis [66]:** Diseñada específicamente para trabajar con React, React-Vis proporciona componentes listos para usar que facilitan la incorporación de visualizaciones de datos en aplicaciones React. Es una elección eficiente para proyectos que utilizan este marco de trabajo.

Por último, dentro del ámbito del FrontEnd destacan dos innovaciones particulares: el WebAssemby y las Progressive Web Apps. Ambas son analizadas más en detalle a continuación, tal y como expone Baryshevskiy en [6].

- **WebAssembly (Wasm):** Permite la ejecución de código de alto rendimiento, escrito en lenguajes como C++ o Rust, directamente en el navegador. Esto amplía significativamente las posibilidades para construir aplicaciones Web más potentes y rápidas, acercando el rendimiento de las aplicaciones Web al de las aplicaciones nativas.
- **Progressive Web Apps (PWA):** Ofrecen experiencias avanzadas que combinan lo mejor de las aplicaciones Web y nativas. Su capacidad para funcionar offline permite a los

usuarios acceder a contenido incluso en ausencia de conexión a Internet, mejorando la accesibilidad y la continuidad de la experiencia del usuario.

En conclusión, la tecnología del Frontend al igual que el BackEnd abarca un conjunto diverso de herramientas y tecnologías esenciales para la creación y operación de aplicaciones Web y servicios. La selección de tecnologías particulares está condicionada por diversos factores, incluyendo los requisitos específicos del proyecto y la complejidad de la aplicación, así como las preferencias del equipo de desarrollo.

Una vez se conoce la diferente tecnología Web que rodea tanto el lado del BackEnd como el lado del FrontEnd, es de gran ayuda conocer los stacks tecnológicos más utilizados y analizarlos. Lo cual se realizará en el siguiente punto.

2.2.3. Stacks Tecnológicos

En el contexto de los stacks tecnológicos, es decir, la elección de un conjunto específico de tecnologías, se trata de un aspecto crítico dentro del desarrollo Web, ya que afecta directamente la eficiencia del desarrollo, la escalabilidad y el mantenimiento del sistema. A continuación son explicadas brevemente cuatro stacks, tal y como expone Campana en [12]: LAMP, Python Django, MEAN y MERN.

- **LAMP (Linux, Apache, MySQL, PHP/Python/Perl):** Ha sido un pilar en el desarrollo Web durante muchos años y sigue siendo una opción confiable para una variedad de proyectos tradicionales. Linux proporciona el sistema operativo, Apache sirve como servidor Web, MySQL maneja la base de datos, y PHP, Python o Perl actúan como lenguajes de programación del lado del servidor. Este conjunto ofrece estabilidad, flexibilidad y una amplia base de conocimientos, siendo especialmente adecuado para aplicaciones Web convencionales.
- **Python Django (Python, Django, Apache, MySQL):** Stack tecnológico compuesto de Python como lenguaje principal de programación, Django como framework Web basado en el patrón MVC, Apache como servidor Web para la gestión de solicitudes HTTP, y MySQL como sistema de gestión de bases de datos. Este conjunto es especialmente adecuado para proyectos Web que buscan la combinación de la versatilidad de Python, la solidez de Django, y la confiabilidad de Apache y MySQL.
- **MEAN (MongoDB, Express.js, Angular, Node.js):** Es un stack tecnológico basado completamente en JavaScript. MongoDB es la base de datos NoSQL, Express.js actúa como el framework del lado del servidor, Angular es el framework del lado del cliente y Node.js proporciona el entorno de ejecución del servidor. Este stack ha ganado popularidad por su coherencia en el uso de un solo lenguaje de programación (JavaScript/TypeScript) en

todo el flujo de desarrollo, lo que simplifica la colaboración entre los equipos de desarrollo y permite una transición más fluida de los datos entre el servidor y el cliente.

- **MERN (MongoDB, Express.js, React, Node.js):** Similar a MEAN, pero reemplaza Angular con React en el lado del cliente. React ha ganado una enorme popularidad por su enfoque declarativo y su capacidad para construir interfaces de usuario altamente interactivas. MERN comparte las ventajas de MEAN, pero además permite a los desarrolladores aprovechar las características distintivas de React para crear experiencias de usuario más dinámicas y eficientes.

Con todo esto en mente, se puede concluir que la elección entre los diferentes stacks tecnológicos que existen depende de las necesidades específicas del proyecto y las preferencias del equipo de desarrollo. Teniendo cada stack sus propias ventajas y pudiendo ser la elección correcta según los requisitos particulares de la aplicación que se esté construyendo.

Finalmente, en la Tabla 2.1 se puede observar una visualización detallada de la tecnología abordada en este capítulo. Siendo destacable como cada stack tecnológico está compuesto por diferentes tipos de tecnologías, demostrando así la diversidad y complejidad de las soluciones presentadas.

Tabla 2.1: Tecnología en relación al capítulo 2.2

Stack	SO	Servidor	BD	Lenguaje	BackEnd	FrontEnd
LAMP	Linux	Apache	MySQL	PHP/Python/Perl	-	-
Python Django	-	Apache	MySQL	Python	Django	-
MEAN	-	-	MongoDB	JavaScript	Express.js	Angular
MERN	-	-	MongoDB	JavaScript	Express.js	React

Una vez se conoce toda la tecnología que se puede utilizar a la hora de construir una aplicación Web. Es fundamental, tener en mente las diferentes técnicas de visualización que existen para asegurar que el usuario entienda los datos que se muestran con dicha aplicación. Estas técnicas son analizadas en el siguiente punto.

2.3. Técnicas de Visualización

Como se ha mencionado previamente, para desarrollar la aplicación Web deseada es necesario mostrar datos de una manera atractiva al usuario y para ello hay que hacer uso de las técnicas existentes de visualización de datos. Al fin y al cabo, esto vendría a ser la traducción de una

gran cantidad de información en un contexto visual usando elementos gráficos. De forma que consiga comunicar relaciones de datos complejas para una mejor comprensión de los usuarios. A continuación es realizado un análisis de las diferentes técnicas.

2.3.1. Tipos de Visualización

Para conseguir mostrar toda esta cantidad de datos existen diferentes tipos de visualización de datos, algunos de estos son los que se presentan a continuación, tal y como expone Calzon en [11].

- **Gráficos de números:** Representan inmediatamente un dato clave.
- **Gráficos de líneas:** Muestran datos a lo largo de una línea continua, conectando puntos de datos. Se utilizan comúnmente para visualizar tendencias a lo largo del tiempo.
- **Mapas:** muestran los datos por una localización geográfica. Normalmente, la información se muestra con un mapa de áreas coloreado.
- **Gráfico de cascada:** Muestra los datos durante un periodo de tiempo determinado, ilustrando los valores positivos y negativos
- **Gráficos de barras:** Representan datos utilizando barras rectangulares de longitudes proporcionales a los valores que representan. Pueden ser horizontales o verticales. Además, pueden mostrar las barras individualmente, acumuladas o agrupadas entre ellas.
- **Gráficos de pastel (pie charts):** Dividen un círculo en sectores para representar proporciones. Son útiles para mostrar la distribución porcentual de un conjunto de datos.
- **Gráficos de calibre (gauge charts):** Utiliza agujas y colores para mostrar datos similares a la lectura en un velocímetro.
- **Gráficos radiales (radar charts):** Muestran datos en un formato circular, útiles para comparar múltiples categorías en función de varias variables.
- **Diagramas de dispersión:** Representan puntos en un plano cartesiano para mostrar la relación entre dos variables. Cada punto representa una observación.
- **Tablas:** Forma común de visualización de datos que organiza la información en filas y columnas. Cada fila representa un conjunto de datos y cada columna representa un atributo.
- **Gráficos de burbujas:** Similar a un diagrama de dispersión, pero con la adición de un tercer valor que se representa mediante el tamaño de las burbujas.
- **Diagramas de caja (boxplots):** Representan la distribución estadística de un conjunto de datos a través de cuartiles. Muestran la mediana, los cuartiles y los valores atípicos.
- **Gráfico de embudo (funnel chart):** Muestra como los datos se mueven a través de un proceso específico.
- **Mapas de calor:** utilizan colores para representar la intensidad de un valor en una matriz

bidimensional. Son útiles para mostrar la concentración de datos en áreas específicas.

Sin embargo, no solo es necesario conocer qué técnicas existen, sino también saber cuáles son las mejores prácticas a la hora de utilizarlas. En el siguiente punto, se tratarán estas diferentes prácticas.

2.3.2. Mejores Prácticas

Existen diferentes prácticas recomendadas que se emplean a la hora de mostrar los datos, entre las que encontramos el uso de la teoría de color, la importancia de la sencillez, el uso de comparaciones, entre otros muchas más. A continuación se analizan más en detalle todas estas prácticas y consejos expuestas nuevamente por Calzon en [10].

- **Conocer el caso de uso para cada tipo.** Es de vital importancia tener en mente que cada tipo de visualización de datos es mejor en diferentes situaciones.
- **Evitar mostrar datos que puedan engañar al usuario.** Para ello se aconseja principalmente que los ejes no empiecen por 0 y que tampoco se omitan datos.
- **Tomar ventaja de la teoría de color.** Al fin y al cabo, los colores no solo ayudan a destacar datos importantes, sino también está demostrado que distintos colores causan distintas emociones a las personas. Además, se pueden aprovechar asociaciones de colores que los usuarios den por hecho, como colores de tonos más claros para valores bajos.
- **Priorizar la sencillez.** Se trata de otra buena práctica para conseguir que el usuario entienda mejor los datos. Evitando muchas etiquetas, imágenes, colores demasiado brillantes, entre otras cosas.
- **Usar los diferentes elementos textuales con cuidado.** Haciendo que los subtítulos sean cortos y contengan las unidades de medida; etiquetas cortas; ordenar la leyenda por orden de aparición; adición de tooltips interactivos para mostrar texto adicional.
- **Incluir comparaciones.** Presentar dos gráficas juntas que muestren la misma información durante un periodo de tiempo particular proporcionará una guía del impacto de los diferentes datos.
- **Añadir interactividad a los datos.** De esta forma se consigue dar vida a la visualización de datos, permitiendo a los usuarios navegar por ellos, a través de filtros y widgets.

Todas estas prácticas ayudan en gran medida a que el usuario comprenda mejor los datos que se muestran. Sin embargo, otra buena práctica que se recomienda es el uso de los principios de Gestalt, los cuales se detallan mejor en el siguiente punto.

2.3.3. Principios de Gestalt

Los principios de Gestalt describen cómo los humanos agrupan elementos similares, reconocen patrones y simplifican imágenes complejas cuando se perciben objetos. Los diseñadores utilizan estos principios para organizar el contenido en sitios Web y otras interfaces para que sea estéticamente agradable y fácil de entender. En este punto se tratan los principios de simplicidad, proximidad y cierre que pueden ser usados en la visualización de datos, tal y como expone Meeks en [41]. A continuación se detallan cada uno de ellos.

- **Simplicidad:** Indica que los elementos gráficos que tengan propiedades visuales compartidas se consideran como parte de un mismo grupo. En la Figura 2.3 se puede ver el uso de similitud de color para indicar dos clases de elementos: los rojos y los grises.
- **Proximidad:** Sigue la misma lógica que el anterior, este principio indica que los elementos que estén próximos entre ellos se consideran como un mismo grupo. En la Figura 2.3 se puede ver el uso de este principio.
- **Cierre:** Nuevamente sigue la misma lógica anterior, encerrando los elementos en un área. Como se puede ver en la Figura 2.3, junto a los otros dos principios, este último es el más fuerte visualmente.

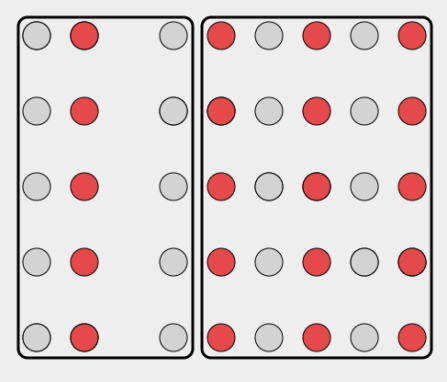


Figura 2.3: Principios de Gestalt [41]

Los principios que se han presentado aquí solo son tres de los muchos que existen y son aplicados. Sin embargo, con los ejemplos anteriores uno se puede dar cuenta de la gran utilidad que tienen estos principios a la hora de mostrar datos a los usuarios, y poder dar una mejor experiencia de usuario. Concepto el cual se profundizará más en el siguiente punto junto con el concepto de usabilidad.

2.4. Interacción Persona-Ordenador

Por último, para el desarrollo de una aplicación Web enfocada en el lado del cliente es indispensable detenerse para hablar de la disciplina de interacción Persona-Ordenador, de la usabilidad y de la experiencia de usuario. A continuación, se detallan cada uno de estos conceptos.

2.4.1. Interacción Persona-Ordenador

La IPO es definida por SIGCHI como “la disciplina relacionada con el diseño, evaluación e implementación de sistemas informáticos interactivos para el uso de seres humanos, y con el estudio de los fenómenos más importantes con los que está relacionado” [26].

Dentro de esta disciplina, se pueden encontrar distintos aspectos, los cuales se pueden ver reflejados en la Figura 2.4. Por un lado, se observa una persona con sus características de procesamiento de la información, tanto las de comunicación como las físicas que interactúa con un ordenador. El cual también tiene sus propias características. Por otro lado, en medio se encuentran los dispositivos de entrada y salida que relacionan a la persona con el ordenador, comunicándose mediante diferentes elementos de diseño.

Además, la figura muestra que la persona no está sola, sino que realiza el trabajo dentro de una organización social, siendo posible gracias a un proceso de desarrollo en el que cada uno de estos componentes debe ser abordado con igual grado de implicación y no caer en el error de obviar la parte humana, centrándose solamente en la parte tecnológica.

Con todo esto en mente, se puede decir que la disciplina de interacción Persona-Ordenador es la encargada de estudiar la usabilidad de la aplicación Web. Concepto que se detalla más en profundidad en el siguiente punto.

2.4.2. Usabilidad

El concepto de usabilidad fue introducido por J. Nielsen en [52], quien concluyó que un sistema software tiene dos componentes: el aspecto funcional y la forma en que los usuarios pueden usar este aspecto. Siendo este último, el que es tratado para mejorar la usabilidad de una aplicación. Por consiguiente, los aspectos que se tienen en mente al hablar de usabilidad serían la facilidad de aprendizaje, la efectividad de uso y la satisfacción con las que las personas son capaces de realizar sus tareas.

Teniendo esto en cuenta, la usabilidad se puede definir coloquialmente como “fácil de usar o de utilizar y de aprender” [27]. Definición que se considera correcta, pero incompleta, ya que el

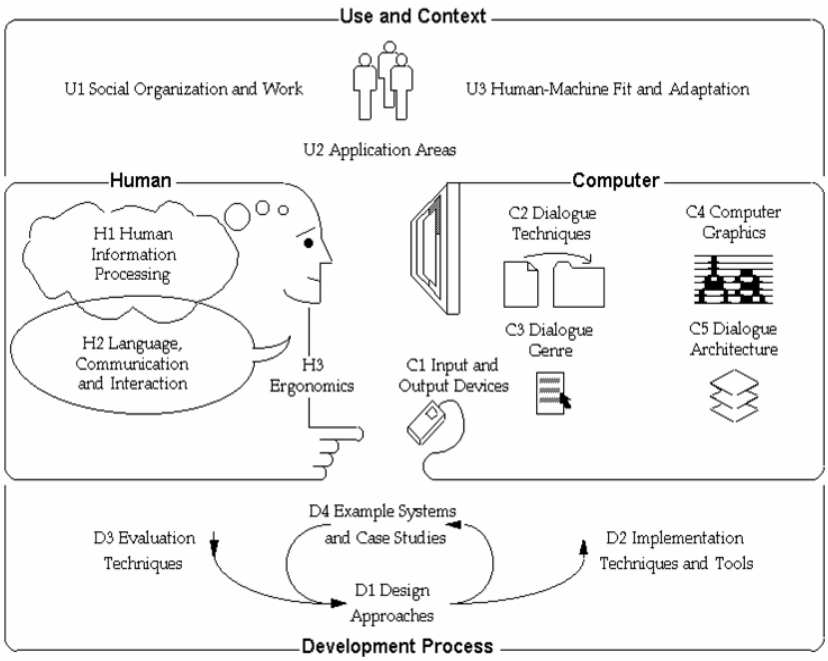


Figura 2.4: Flujo de la disciplina de la Interacción Persona-Ordenador [26]

concepto engloba muchos más aspectos, es por ello que el organismo de estandarización ISO propone dos definiciones:

- La medida en la que un producto se puede usar por determinados usuarios para conseguir objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso especificado [36].
- La capacidad que tiene un producto software para ser atractivo, entendido, aprendido, usado por el usuario cuando es utilizado bajo unas condiciones específicas [38].

Una vez se ha definido el concepto de usabilidad, se debe saber que este tiene un gran conjunto de atributos, los cuales Granollers en [27] los resumen en: facilidad de aprendizaje, sintetizabilidad, familiaridad, consistencia, flexibilidad, robustez, recuperabilidad, tiempo de respuesta, adecuación de tareas y disminución de la carga cognitiva.

Al final, se puede concluir que para conseguir un buen grado de usabilidad en una aplicación se requiere pensar en el usuario, lo cual se consigue diseñando una interfaz centrada en el usuario. Esto implica conocer las particularidades de los usuarios para cada caso y reflejarlas en la interacción de la interfaz. Aunque un equipo de desarrollo pueda implementar sistemas similares, es crucial involucrar a los usuarios desde el inicio para lograr una familiaridad que proporcione seguridad y relajación durante la manipulación del sistema.

El objetivo es lograr una interfaz fácil de usar y aprender, lo cual requiere una estrecha cola-

boración con los usuarios a lo largo de todo el proceso de desarrollo. Implicar a los usuarios desde el principio es fundamental, y no se debe confundir con simplemente diseñar pensando en el usuario sin su participación activa. Es crucial conocer la opinión de los usuarios de primera mano para garantizar la utilidad y la comodidad de la interfaz.

Además, los sistemas interactivos deben centrarse en todos los usuarios, considerando las diferencias individuales, incluso teniendo en cuenta a aquellos con discapacidades. Ignorar a los usuarios o dejar su participación a la fase final del proyecto puede resultar en una interfaz que no cumple con sus necesidades y expectativas. En resumen, el diseño de sistemas interactivos implica hacer del usuario el foco principal desde el inicio del proceso hasta la implementación final, abarcando la diversidad de usuarios y sus características específicas.

Por otro lado, dentro de la disciplina de la IPO, también se puede encontrar el término experiencia de usuario, el cual no debe ser confundido con usabilidad, después de todo este último es una faceta del primero. En el siguiente punto, se explica mejor el concepto de experiencia de usuario.

2.4.3. Experiencia de Usuario

El término de experiencia de usuario no tiene una definición consensuada, sin embargo, una de las más destacadas es la que propone el estándar ISO en [37], definiendo este concepto como "las percepciones y respuestas de una persona que resultan del uso y/o uso anticipado de un producto, sistema o servicio".

Además, la experiencia de usuario presenta diferentes facetas a considerar para el diseño o evaluación de un sistema interactivo. Sin embargo, nuevamente al igual que ocurre con su definición, todavía no están consensuadas ni por la comunidad científica ni por ningún organismo de estandarización. A continuación, se mencionan diferentes facetas para diferentes autores:

- El autor Peter Morville en [50] definió la colmena de la UX en la que se incluyen: usable, útil, deseable, valioso, creíble, encontrable y accesible como atributos a considerar para obtener una experiencia de usuario positiva.
- Autores como Hassenzahl y Tractinsky en [31] proponen tres facetas: más allá de lo instrumental; experimental; y emociones y afectos.
- Otros referentes como Hassan Montero y Ortego Santamaría en [30] señalan a usuario, contexto y contenido como los componentes más importantes dentro de la experiencia de usuario.

Para concluir, Granollers nos sugiere la siguiente definición en [28] cubriendo los diferentes aspectos: "la experiencia de usuario atiende a todos los factores, tanto internos como externos del usuario y del sistema interactivo, que causen alguna sensación a quien esté utilizando un siste-

ma interactivo concreto en un determinado contexto de uso” . Entre estos factores, encontramos la usabilidad, la cual ya ha sido explicada en el punto anterior, y será un factor a tener muy en cuenta a la hora de desarrollar la aplicación Web deseada. De la cual se hablará más en detalle en los próximos capítulos.

Capítulo 3

Propuesta de Solución

En este capítulo se proporciona la propuesta diseñada para abordar el problema presentado en el capítulo de introducción. Asimismo, se presenta una explicación detallada de la metodología de desarrollo seleccionada y las herramientas empleadas a lo largo del proyecto.

3.1. Propuesta de Solución

Para llevar a cabo la solución al problema planteado en la introducción, se propone el desarrollo de una aplicación Web moderna y eficiente con el fin de brindar una buena experiencia de usuario. Para conseguir esto se plantea la creación de una SPA con una arquitectura MVC teniendo un foco en la usabilidad. A continuación, se detallarán mejor todos estos puntos.

3.1.1. Tecnología SPA

Los SPA, como se detalla en el capítulo anterior y señala BasuMallick en [7], ofrecen la ventaja crucial de posibilitar la interacción del usuario sin tener que recargar la página. Esta característica contribuye de manera significativa a la reducción del tiempo de carga, lo que, a su vez, se traduce en una mejora sustancial en la velocidad de la aplicación. Al prescindir de la recarga de la página, como se ilustra en la Figura 3.1, solo es necesario intercambiar datos, mientras que recursos fundamentales como HTML y CSS se cargan de manera única al inicio.

Además, los SPA pueden incorporar las funcionalidades de los PWA, permitiendo su uso incluso cuando no hay conexión a internet. Esto lo consiguen los SPA enviando una única petición al servidor, para luego guardar y almacenar lo que recibe en el caché, permitiendo sincronizar los datos sel servidor cuando la conexión lo permita.

Asimismo, los SPA permiten crear aplicaciones Web con una compatibilidad multiplataforma,

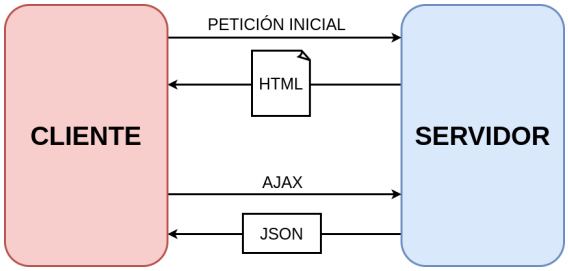


Figura 3.1: Flujo de una SPA

teniendo la posibilidad de utilizar la aplicación en cualquier dispositivo, navegador y sistema operativo.

Finalmente, con todas estas características se puede concluir que los SPA mejoran significativamente la experiencia del usuario gracias a la velocidad, la posibilidad de utilizarla offline y en cualquier dispositivo. Para la creación de dicha aplicación, se propone también el uso de una arquitectura MVC, la cual será detallada en el siguiente punto.

3.1.2. Arquitectura MVC

Con una arquitectura MVC, tal y como explica Hernández en [33], se conseguirá fortalecer la robustez y escalabilidad de la aplicación, al dividir el código en tres componentes principales, cada uno con responsabilidades específicas, lo que facilita la organización, el mantenimiento y la expansión esenciales en una aplicación moderna y eficiente. Estos tres componentes consisten en: Modelo, Vista y Controlador.

1. **Modelo:** Encargado de gestionar los datos y la lógica. En el contexto de la aplicación de diagnóstico de fibromialgia, el Modelo podría ser responsable de la manipulación de datos relacionados con los pacientes, resultados de diagnóstico y cualquier información relevante para el proceso de evaluación médica.
2. **Vista:** Representa la interfaz de usuario y se encarga de la presentación de datos. La Vista en este caso sería responsable de mostrar al usuario la información relevante de manera clara y comprensible, teniendo un gran enfoque en la usabilidad.
3. **Controlador:** Gestiona la interacción del usuario y actúa como intermediario entre el Modelo y la Vista. El Controlador sería responsable de procesar las acciones del usuario, actualizar el Modelo según sea necesario y coordinar la presentación actualizada a través de la Vista.

Esta división clara de responsabilidades, que se puede ver en la Figura 3.2, no solo facilita el desarrollo y la mantenibilidad del código, sino que también proporciona una base sólida para

futuras expansiones y mejoras en la aplicación.

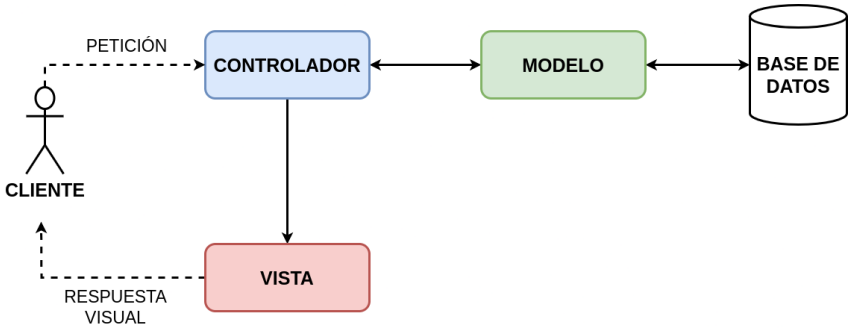


Figura 3.2: Arquitectura MVC

En adicción al uso de esta arquitectura para la escalabilidad de la aplicación, se plantea la elección de una **base de datos no relacional**, ya que estas ofrecen ventajas significativas en términos de flexibilidad y capacidad para manejar grandes volúmenes de datos.

Por último, en el siguiente punto se detallará la propuesta de un enfoque en la usabilidad con el objetivo de dar una gran experiencia al usuario final.

3.1.3. Usabilidad

Diseñar una interfaz intuitiva y usable que responda a las necesidades del usuario final, haciendo que sea tanto fácil de utilizar como de aprender, es esencial en una aplicación Web que quiera brindar una gran experiencia de usuario.

La creación de una interfaz intuitiva implica la adopción de elementos visuales y de diseño que reflejen la lógica y el flujo de trabajo esperado por el usuario. Se busca minimizar la curva de aprendizaje, permitiendo que tanto profesionales de la salud como pacientes puedan interactuar de manera natural con la aplicación desde el primer momento. La disposición lógica de los elementos, la claridad en las indicaciones y la consistencia en el diseño contribuirán a una experiencia de usuario sin complicaciones.

En paralelo, se plantea la implementación de un **diseño responsive** que se adapte fluidamente a diversos tamaños de pantalla. Después de todo, la gran variabilidad de dispositivos que existen, como ordenadores, tablets y teléfonos móviles, requiere que la interfaz sea flexible y se ajuste de manera óptima a cada contexto. Esta adaptabilidad no solo mejora la accesibilidad, sino que también proporciona una experiencia consistente y agradable independientemente del dispositivo utilizado.

En resumen, con una interfaz usable, tecnología SPA y arquitectura MVC se asegura que la aplicación sea moderna y eficiente, ofreciendo una buena experiencia al usuario. No obstante, para

lograr la implementación exitosa de dicha aplicación Web, es esencial seguir una metodología de desarrollo adecuada, la cual se detallará en el próximo punto.

3.2. Metodología de Desarrollo

En el contexto de la metodología de desarrollo a seguir, se ha optado por adoptar la metodología ágil, específicamente Scrum. En este apartado, se proporcionará una justificación exhaustiva de la elección de Scrum como marco de trabajo. Además, se explorará el flujo de trabajo característico de esta metodología, y se detallarán las adaptaciones específicas realizadas para asegurar la alineación con los objetivos y características particulares del proyecto.

3.2.1. Justificación

Para justificar la elección de la metodología de Scrum, es esencial conocer en primer lugar la decisión de adoptar una metodología ágil para el desarrollo de la aplicación Web. Esta elección se fundamenta principalmente en tres razones: flexibilidad ante cambios, entregas continuas y una colaboración activa con el cliente. A continuación, se detallan cada una de estas razones.

1. **Flexibilidad ante cambios:** Los proyectos de desarrollo de aplicaciones Web a menudo enfrentan cambios en los requisitos del cliente. Las metodologías ágiles están diseñadas para adaptarse fácilmente a cambios, permitiendo una respuesta rápida y eficiente a las necesidades cambiantes a lo largo del ciclo de vida del proyecto.
2. **Entregas iterativas y continuas:** Las metodologías ágiles promueven entregas iterativas y continuas de incrementos de software funcional. En el contexto de una aplicación Web, esto significa que se pueden entregar funcionalidades tangibles en intervalos cortos, lo que facilita la obtención de retroalimentación temprana y la posibilidad de ajustar el enfoque según las necesidades cambiantes del usuario.
3. **Colaboración activa con el cliente:** La interacción continua con el cliente es un pilar fundamental de las metodologías ágiles. En el desarrollo de aplicaciones Web, donde la experiencia del usuario es crucial, la colaboración directa con el cliente asegura una comprensión precisa de sus expectativas y permite ajustes rápidos para garantizar la satisfacción del usuario final.

Sin embargo, como se ha mencionado previamente, dentro de las metodologías ágiles, la elección específica ha sido Scrum. A continuación, se detallan las razones principales para elegir dicha metodología.

1. **Roles claros y definidos:** Scrum define roles específicos como el Scrum Master, Product

Owner y Desarrolladores, proporcionando claridad en las responsabilidades y contribuyendo a una gestión eficaz del proyecto.

2. **Estructura de Sprints:** La división del proyecto en Sprints permite entregas incrementales regulares, lo cual es beneficioso en el desarrollo Web para mantener un ritmo constante de desarrollo y despliegue de funcionalidades.
3. **Foco en la colaboración:** Scrum promueve la colaboración continua entre los miembros del equipo y el cliente, facilitando la resolución rápida de problemas y la adaptación a los cambios de manera conjunta.
4. **Gestión efectiva de los requisitos:** La gestión del PB en Scrum permite la priorización y adaptación de los requisitos, especialmente importante en el desarrollo Web, donde la variedad de funcionalidades puede ser extensa.
5. **Mejora continua a través de la retroalimentación:** Los eventos de revisión y retrospectiva en Scrum proporcionan oportunidades cruciales para la mejora continua, permitiendo ajustar enfoques y procesos según la retroalimentación del equipo y del cliente.

En conclusión, la elección de Scrum como metodología ágil para el desarrollo de la aplicación Web se fundamenta en sus principios sólidos, su enfoque estructurado y su capacidad para adaptarse a las complejidades del desarrollo de software en el entorno dinámico de la Web.

Una vez se conoce la justificación de la elección de Scrum, es esencial conocer exactamente cómo es el flujo de dicha metodología, lo cual se explicará en el siguiente punto.

3.2.2. Scrum

Scrum se destaca como una metodología ágil que brinda apoyo a individuos y equipos en la creación de soluciones adaptables para desafíos complejos. Su implementación eficaz demanda la colaboración de un equipo compacto, típicamente compuesto por 10 personas o menos. Este equipo se distingue por su estructura no jerárquica, su versatilidad multifuncional y su capacidad de autogestión. En este contexto, Scrum define tres roles específicos, cada uno desempeñando una función crucial: Desarrolladores, Product Owner y Scrum Master.

1. **Desarrolladores:** Este grupo asume la responsabilidad de llevar a cabo el desarrollo del producto, transformando elementos de la lista de requisitos en incrementos funcionales del software.
2. **Product Owner:** Representa a todas las partes interesadas del producto y se encarga de maximizar su valor para clientes, usuarios y demás involucrados. Su enfoque se centra en la toma de decisiones orientada a los objetivos.
3. **Scrum Master:** Juega un papel vital en garantizar el correcto funcionamiento de la metodología Scrum. Facilita la comprensión de la teoría y práctica de Scrum tanto dentro del

equipo como en toda la organización. Además, se esfuerza por eliminar obstáculos y fomentar un entorno favorable para la colaboración y mejora continua.

Esta estructura de roles en Scrum establece un equilibrio que promueve la eficiencia, la flexibilidad y la adaptabilidad, elementos clave para abordar la complejidad de los proyectos de desarrollo de software. No obstante, Scrum no se limita únicamente a roles específicos, sino que también incorpora una serie de eventos que tienen lugar en cada iteración, la cual es comúnmente llamada Sprint.

El **Sprint** funciona como un contenedor que abarca los distintos eventos, con una duración preestablecida de un mes o menos. Tan pronto como concluye un Sprint, el siguiente comienza de inmediato. Dichos eventos están diseñados para fomentar la colaboración y la transparencia, y desempeñan un papel esencial en el éxito de la metodología Scrum. Los eventos clave en Scrum incluyen: Planificación del Sprint, Scrum Diario, Revisión del Sprint y Retrospectiva del Sprint.

1. **Planificación del Sprint:** En este evento, el equipo define el trabajo a realizar durante el próximo Sprint. Se seleccionan elementos del PB para abordar en el Sprint y se establece un objetivo claro.
2. **Scrum Diario:** Este evento diario brinda la oportunidad a los miembros del equipo para sincronizar sus actividades, compartir actualizaciones sobre su progreso, identificar posibles obstáculos y ajustar la planificación según sea necesario.
3. **Revisión del Sprint:** Al final de cada Sprint, el equipo presenta el trabajo completado al Product Owner y otros interesados. Se obtiene retroalimentación y se ajustan las prioridades para el próximo Sprint.
4. **Retrospectiva del Sprint:** Después de la revisión del Sprint, el equipo reflexiona sobre su desempeño, identifica áreas de mejora y establece acciones para implementar en el próximo Sprint con el fin de mejorar la calidad y la eficacia.

Estos eventos proporcionan un marco temporal y oportunidades para la colaboración, asegurando que el equipo se adapte continuamente y entregue valor de manera consistente. La interacción entre los miembros del equipo y la participación activa en estos eventos son esenciales para un correcto flujo de trabajo en Scrum.

Finalmente, el equipo también interactúa con diferentes artefactos en el contexto de Scrum. Estos artefactos proporcionan transparencia y visibilidad sobre el trabajo realizado, el trabajo por hacer y el estado general del proyecto. Los principales artefactos en Scrum son: Product Backlog, Sprint Backlog, Incremento.

1. **Product Backlog:** Esta lista priorizada de elementos describe las funcionalidades, mejoras y correcciones deseadas para el producto. Los desarrolladores junto con la ayuda del

Product Owner son responsables de gestionar y priorizar este backlog, asegurándose de que siempre refleje las necesidades más importantes del cliente y del negocio.

2. **Sprint Backlog:** Durante la planificación del Sprint, el equipo selecciona elementos del PB para abordar en el Sprint. Estos elementos se trasladan al SB, que sirve como una lista detallada de las tareas que el equipo planea completar durante el Sprint.
3. **Incremento:** Este es el resultado del trabajo del equipo al final de cada Sprint. Es un producto potencialmente entregable que incluye todas las funcionalidades completadas durante el Sprint.

Estos artefactos proporcionan una representación clara de las metas y el progreso del equipo. La interacción efectiva con estos elementos asegura una comprensión compartida de las prioridades y una adaptabilidad continua durante el desarrollo del producto en el marco de Scrum.

La información presentada anteriormente se ha obtenido de “La Guía de Scrum” [62] por Schwaber y Sutherland. En conclusión, la interacción de todos estos elementos en la metodología Scrum se ilustra en la figura 3.3. Comienza con la reunión de planificación, en la que participan todos los roles, marcando así el inicio del Sprint. Este Sprint tiene una duración máxima de 4 semanas y se acompaña de reuniones diarias realizadas por el equipo de desarrollo. Durante el Sprint, se llevan a cabo las tareas del SB, seleccionadas previamente del PB, y culminan en un incremento al final del Sprint. A continuación, se llevan a cabo la reunión de revisión y la retrospectiva, nuevamente con la participación de todo el equipo Scrum.

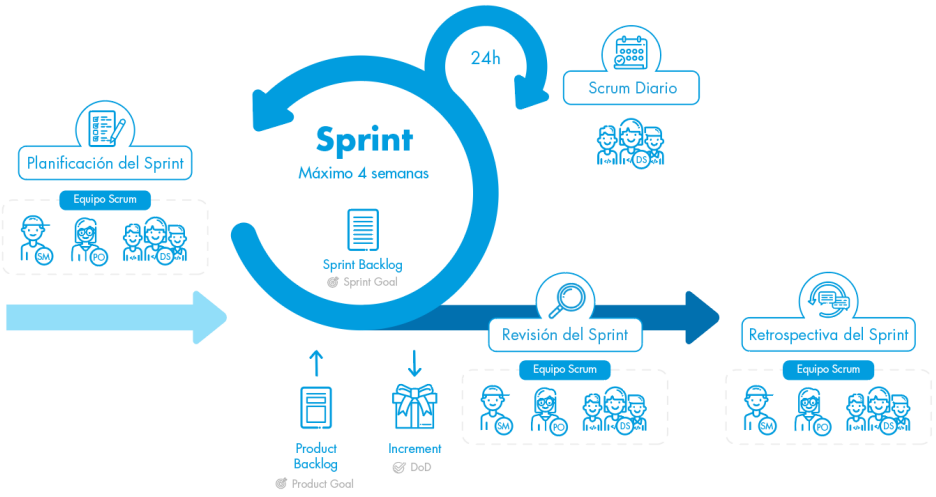


Figura 3.3: Flujo de la metodología Scrum [60]

No obstante, para la ejecución de este proyecto, ha sido esencial realizar adaptaciones específicas de la metodología, las cuales se detallan en el siguiente apartado.

3.2.3. Adaptación

Como se ha expuesto anteriormente, Scrum proporciona una estructura bien definida con roles específicos, eventos y artefactos; no obstante, su flexibilidad permite adaptaciones según las particularidades de cada proyecto. En el caso de este proyecto, se enfrenta a limitaciones y condiciones específicas que han influido en la configuración tradicional de Scrum.

Por un lado, una consideración clave radica en la **reducida composición del equipo**, conformado exclusivamente por el autor del proyecto y los dos tutores. Ante esta restricción, se han realizado ajustes en la asignación de roles:

- El equipo de desarrollo estará compuesto únicamente por un desarrollador, en este caso, el propio autor del proyecto.
- Los roles de Scrum Master y Product Owner se han asignado a ambos tutores, quienes desempeñarán estas funciones de manera colaborativa.

Por otro lado, otra consideración del proyecto es la **limitación de disponibilidad diaria**, la cual impide la realización de las reuniones diarias. Sin embargo, a pesar de esta baja disponibilidad, las reuniones de planificación, revisión y retrospectiva sí serán realizadas, llevándose a cabo en el mismo día.

Estas adaptaciones han sido cuidadosamente diseñadas para mantener la integridad del marco de trabajo Scrum mientras se ajusta a las circunstancias particulares del proyecto. Aunque las modificaciones se han realizado por necesidad, se busca preservar los principios fundamentales de Scrum, como la transparencia, la inspección y la adaptación, para lograr un desarrollo efectivo y un producto final de alta calidad.

Una vez se conoce la metodología que guiará el desarrollo del proyecto, resulta esencial conocer las herramientas que serán utilizadas durante el mismo. Estas herramientas serán expuestas en la siguiente sección.

3.3. Herramientas

Existe una amplia variedad de herramientas disponibles para facilitar el desarrollo de un proyecto. A continuación, se detallarán las distintas herramientas que se emplean en este proyecto, organizadas en tres categorías principales: gestión de proyectos, documentación y desarrollo.

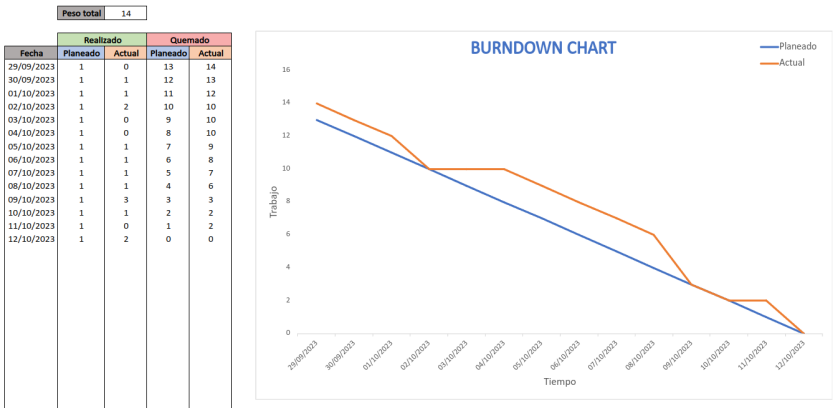
3.3.1. Gestión de Proyecto

En el ámbito de la gestión de proyectos, se han seleccionado las siguientes herramientas:

- **Planificación del proyecto:** GitHub Projects y Excel [44]
- **Comunicación y reuniones:** Outlook [47], Microsoft Teams [45] y OneDrive [46]
- **Control de versiones:** Git [65] y GitHub [58]

Para la planificación detallada del proyecto, se implementará GitHub Projects, una herramienta que simplifica la creación de tableros adaptados al flujo de trabajo de la metodología Scrum, detallada previamente. Esta plataforma será complementada con el uso de Excel, aplicación de hojas de cálculo proporcionada por *Microsoft*, la cual será utilizada para la creación de los diferentes BC destinados a evaluar el trabajo realizado en cada sprint.

Específicamente, se utilizará la plantilla diseñada siguiendo los pasos expuestos por Swam en [63]. La representación visual de esta plantilla se muestra en la Figura 3.4, la cual consiste en una tabla que incluye fechas, el peso planificado de las HU a realizar cada día, el peso real de las HU completadas diariamente y el peso "quemado" en relación con el peso total del Sprint.



Peso total	14			
	Realizado		Quemado	
Fecha	Planeado	Actual	Planeado	Actual
29/09/2023	1	0	13	14
30/09/2023	1	1	12	13
01/10/2023	1	1	11	12
02/10/2023	1	2	10	10
03/10/2023	1	0	9	10
04/10/2023	1	0	8	10
05/10/2023	1	1	7	9
06/10/2023	1	1	6	8
07/10/2023	1	1	5	7
08/10/2023	1	1	4	6
09/10/2023	1	3	3	3
10/10/2023	1	1	2	2
11/10/2023	1	0	1	2
12/10/2023	1	2	0	0

Figura 3.4: Plantilla para la creación del BC

En términos de comunicación y reuniones entre los miembros del equipo, se aprovechará el entorno proporcionado por *Microsoft*. Esto incluye el uso de Outlook para el envío de correos electrónicos, Microsoft Teams para facilitar reuniones virtuales y OneDrive para la compartición de archivos.

Finalmente, para gestionar el control de versiones tanto de la documentación como de la aplicación Web, se empleará Git junto con GitHub. En esta última, se creará una organización, la cual estará configurada con los distintos miembros del equipo, permitiendo la gestión de repositorios, así como del proyecto a través de la herramienta GitHub Projects mencionada anteriormente.

En conjunto, todas estas herramientas permitirán tener una gestión del proyecto adecuada desde la fase de planificación hasta el despliegue. No obstante, el éxito en el desarrollo de un proyecto no solo depende del uso adecuado de herramientas para gestionarlo, sino también de la correcta documentación. A continuación, en la siguiente sección, se detallarán las herramientas

específicas utilizadas para documentar de manera efectiva cada aspecto del proyecto.

3.3.2. Documentación

En lo que respecta a las herramientas destinadas a la documentación del proyecto, se han seleccionado las siguientes:

- **Lenguaje de marcado:** Markdown
- **Editor de texto:** Typora [5]
- **Generación de memoria:** Plantilla de Félix Albertos [1]
- **Diagramas:** Diagrams.net [2]
- **Diseño de interfaces:** Excalidraw [21] y Figma [23]

Por un lado, la documentación se redacta utilizando el lenguaje de marcado Markdown, que simplifica la aplicación de formato a un texto plano mediante caracteres especiales. Para esta tarea, se utiliza el editor Typora, elegido por su experiencia de escritura sin distracciones y su fácil manejo. En cuanto a la generación de la memoria final con el formato adecuado, se emplea la plantilla proporcionada por el tutor Félix Albertos. Esta plantilla, diseñada específicamente para la elaboración del TFG, elimina las posibles preocupaciones de formato al utilizar Markdown como entrada.

Por otro lado, en el caso de necesitar desarrollar diagramas, se recurrirá a Diagrams.net. Finalmente, para el diseño de las interfaces a implementar, se utilizará Excalidraw para bocetos con pocos detalles y Figma para diseños más elaborados.

Con la combinación de todas estas herramientas se conseguirá crear un flujo de trabajo documental eficiente y versátil, optimizando la redacción, formato, generación de memoria y diseño de interfaces en el proyecto. Sin embargo, una vez se han presentado las herramientas destinadas a la documentación, es esencial pasar a detallar las distintas herramientas específicas utilizadas en el desarrollo del proyecto.

3.3.3. Desarrollo

Dentro de las herramientas de desarrollo, se contemplan las que se enumeran a continuación, agrupadas en tres categorías: FrontEnd, BackEnd y Otras.

- **FrontEnd**
  - **Lenguaje de marcado:** HTML
  - **Lenguaje de diseño gráfico:** CSS
  - **Lenguaje de programación:** TypeScript [42]

    - **Framework:** React [22]
    - **Herramienta:** Vite [68]
    - **Librería destacada:** D3.js [8]
- **BackEnd**
    - **Lenguaje de programación:** TypeScript [42]
    - **Entorno de desarrollo:** Node.js [16]
    - **Framework:** Express.js [53]
    - **Base de datos:** MongoDB [48]
    - **Librerías destacadas:** Mongoose [49]
- **Otras**
    - **Editor de código:** Visual Studio Code [43]
    - **Gestor de paquetes:** pnpm [57]

Estas herramientas han sido seleccionadas cuidadosamente con el objetivo de implementar el stack MERN utilizando TypeScript como lenguaje central. El propósito es lograr el desarrollo de una aplicación web moderna y eficiente que abarque tanto el lado del cliente como el del servidor. A continuación, se detallan a fondo cada una de estas tecnologías.

En el FrontEnd, se ha optado por tecnologías líderes en la industria, como HTML, CSS, TypeScript y React. Estas elecciones proporcionan una estructura sólida para la creación de interfaces de usuario dinámicas y atractivas. Además, se ha incorporado Vite como herramienta de construcción, ofreciendo un entorno de desarrollo rápido y liviano que contribuye a la eficiencia del proceso de desarrollo. Finalmente, en el lado del cliente, se resalta la elección de la librería D3.js para la visualización de datos.

En el BackEnd, se repite la elección de TypeScript dentro del entorno de Node.js. Este último será respaldado por Express.js como framework, proporcionando un entorno ágil y escalable para la gestión eficiente de las operaciones del servidor. El stack MERN se solidifica al optar por MongoDB como base de datos, destacándose por su capacidad de almacenamiento flexible y escalable. La interfaz con esta base de datos se simplifica mediante el uso de la librería Mongoose, facilitando el acceso y la manipulación de los datos de manera eficaz.

Finalmente, se destaca la elección de Visual Studio Code como editor de código debido a su popularidad, extensibilidad y potentes características que mejoran la productividad del desarrollador. Además, se utilizará pnpm como gestor de paquetes, el cual añade eficiencia a la gestión de dependencias, facilitando la instalación y actualización de bibliotecas y herramientas de manera más rápida y consistente.

En conclusión, la selección de estas herramientas se alinea con el objetivo de desarrollar una

aplicación Web moderna, eficiente y atractiva para el usuario. Todas estas tecnologías están interrelacionadas, como se ilustra en la Figura 3.5. Esta figura no solo presenta la implementación de la arquitectura MVC previamente mencionada, sino que también muestra una clara división en tres secciones: Cliente, Servidor y Base de Datos.

- **Cliente:** En la sección del cliente, se incorpora toda la tecnología presentada anteriormente en el FrontEnd. Además, esta sección representa la Vista en la arquitectura MVC.
- **Servidor:** En esta sección, se observa la mayoría de la tecnología presentada en el BackEnd. Además, esta sección contiene tanto el Controlador, el cual sería la API utilizando el framework Express, como el Modelo utilizando la librería Mongoose.
- **Base de Datos:** Aquí se destaca el uso de la base de datos no relacional MongoDB, que sirve como el componente clave para el almacenamiento de datos.

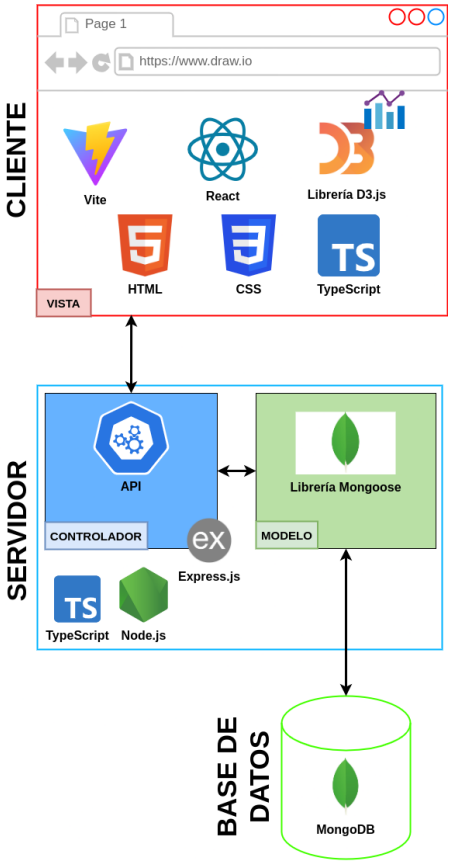


Figura 3.5: Arquitectura tecnológica

Una vez que se ha presentado las diversas herramientas que se utilizarán en el proyecto, junto con la metodología y la propuesta de solución, en el próximo capítulo se explicará el resultado final de la aplicación Web.

Capítulo 4

Resultado

Capítulo 5

Evaluación

Capítulo 6

Conclusiones

Bibliografía

[1] Albertos F. *Template for the Bachelor's Final Project (TFG)*. https://www.felixalbertos.com/resources/downloads/tfg_template.html.

[2] Alder G. *Diagrams.net*. https://app.diagrams.net/.

[3] Alona O. Software Development Life Cycle: An Ultimate Guide. https://qarea.com/blog/software-development-life-cycle-guide.

[4] Apache *ECharts*. https://echarts.apache.org/en/index.html.

[5] appmakes *Typora*. https://typora.io/.

[6] Baryshevskiy, A. The Latest Trends in Web App Development for 2023: What to Expect from the Industry. https://themindstudios.com/blog/web-app-development-trends/.

[7] BasuMallick, C. What Is a Single-Page Application? Architecture, Benefits, and Challenges. https://www.spiceworks.com/tech/devops/articles/what-is-single-page-application/.

[8] Bostock M. *D3.js*. https://d3js.org/.

[9] Cabello R. *Three.js*. https://threejs.org/.

[10] Calzon, B. 17 Essential Data Visualization Techniques, Concepts & Methods To Improve Your Business – Fast. https://www.datapine.com/blog/data-visualization-techniques-concepts-and-methods/.

[11] Calzon, B. Discover 20 Essential Types Of Graphs And Charts And When To Use Them. https://www.datapine.com/blog/different-types-of-graphs-charts-examples/.

[12] Campana, N. Tech Stack: Overview Of The Top Technologies of 2022. https://www.freelancermap.com/blog/tech-stack/.

[13] Chart.js Getting Started. https://www.chartjs.org/docs/latest/getting-started/.

[14] Cherni, B. 2019. *Programming TypeScript making your JavaScript applications scale*. O'Reilly.

[15] Clark, H. The Software Development Life Cycle (SDLC): 7 Phases and 5 Models. https://theproductmanager.com/topics/software-development-life-cycle/.

[16] Dahl R. *Node.js*. https://nodejs.org/en.

[17] Django Software Foundation *Django*. https://www.djangoproject.com/.

[18] Docker Use containers to Build, Share and Run your applications. https://www.docker.com/resources/what-container/.

[19] Downie N. *Chart.js*. https://www.chartjs.org/.

[20] Espírito, D. Top 5 main Agile methodologies: advantages and disadvantages. https://www.xpand-it.com/blog/top-5-agile-methodologies/.

[21] Excalidraw *Excalidraw*. https://excalidraw.com/.

[22] Facebook *React*. https://es.react.dev/.

[23] Figma, Inc. *Figma*. https://figma.com.

[24] Google *Angular*. https://angular.io/.

[25] Google *Kubernetes*. https://kubernetes.io/.

[26] Granollers, T. La Interacción Persona-Ordenador. https://mpiua.invid.udl.cat/la-interaccion-persona-ordenador/.

[27] Granollers, T. Usabilidad. https://mpiua.invid.udl.cat/usabilidad/.

[28] Granollers, T. User eXperience (UX). https://mpiua.invid.udl.cat/user-experience/.

[29] Harris R. *Svelte*. https://svelte.dev/.

[30] Hassan-Montero, S., Y. y Ortega-Santamaría 2009. *Informe APEI sobre usabilidad*. APEI, Asociación Profesional de Especialistas en Información.

[31] Hassenzahl, N., M. y Tractinsky 2006. User experience - a research agenda [editorial]. *Behavior & Information Technology*. 25, 2 (2006), 91–97.

[32] Heinemeier D. *Ruby on Rails*. https://rubyonrails.org/.

[33] Hernández, U. MVC (Model, View, Controller) explicado. https://codigofacilito.com/articulos/mvc-model-view-controller-explicado.

[34] Highsoft AS *Highcharts*. https://www.highcharts.com/.

[35] Hykes S. *Docker*. https://www.docker.com/.

[36] International Organization for Standardization 1998. Ergonomic requirements for office work with visual display terminals (VDTs)-part 11: Guidance on usability.

[37] International Organization for Standardization 2008. Ergonomics of human system interaction – part.

[38] International Organization for Standardization 1991. Software engineering-product quality.

[39] Majorek, J. 19 Best JavaScript Data Visualization Libraries. https://www.monterail.com/blog/javascript-libraries-data-visualization.

[40] Matsumoto Y. *Ruby*. https://www.ruby-lang.org/.

[41] Meeks, E. Gestalt Principles for Data Visualization. https://emeeks.github.io/gestaltdataviz/section1.html.

[42] Microsoft *TypeScript*. https://www.typescriptlang.org/.

[43] Microsoft *Visual Studio Code*. https://code.visualstudio.com/.

[44] Mircrosoft *Excel*. https://www.microsoft.com/es-es/microsoft-365/excel%0Ahttps://www.microsoft.com/es-es/microsoft-teams/log-in.

[45] Mircrosoft *Mircrosoft Teams*. https://www.microsoft.com/es-es/microsoft-teams/log-in.

[46] Mircrosoft *OneDrive*. https://www.microsoft.com/es-es/microsoft-365/onedrive/online-cloud-storage.

[47] Mircrosoft *Outlook*. https://www.microsoft.com/es-es/microsoft-365/outlook/email-and-calendar-software-microsoft-outlook.

[48] MongoDB, Inc. *Mongodb*. https://www.mongodb.com/.

[49] Mongoose *Mongoose*. https://mongoosejs.com/.

[50] Morville, P. 2005. Experience design unplugged. *ACM SIGGRAPH 2005 web program (SIGGRAPH '05)* (New York, USA, 2005), Article 10.

[51] Neo4j *Neo4j*. https://neo4j.com/.

[52] Nielsen, J. 1993. *Usability Engineering*.

[53] OpenJS Foundation *Express.js*. https://expressjs.com/.

[54] Oracle Corporation *MySQL*. https://www.mysql.com/.

[55] Osadchuk, S. Top Backend Technologies in 2023: choose the right one for your solution. https://doit.software/blog/backend-technologies#screen5.

[56] Otwell T. *Laravel*. https://laravel.com/.

[57] pnpm *Figma*. https://pnpm.io/.

[58] Preston-Werner T., Wanstrath C., Hyett P. *GitHub*. https://github.com/.

[59] Ramotion The Role of Database in Web Application Development. https://www.ramotion.com/blog/database-in-web-app-development/.

[60] Rodríguez, M. Scrum: el pasado y el futuro. https://netmind.net/es/scrum-el-pasado-y-el-futuro/.

[61] Rosssum G. *Python*. https://www.python.org/.

[62] Schwaber, K. y Sutherland, J. The 2020 Scrum Guide. https://scrumguides.org/scrum-guide.html#end-note.

[63] Swan, G. How to Create a Burndown Chart in Excel? (With Templates). https://clickup.com/blog/burndown-chart-excel/.

[64] Three.js Creating a Scene. https://threejs.org/docs/index.html#manual/en/introduction/Creating-a-scene.

[65] Torvalds L. *Git*. https://git-scm.com/.

[66] Uber Technologies, Inc. *React Vis*. https://uber.github.io/react-vis/.

[67] UCLM Memoria Oficial del Grado de Ingeniería Informática. https://www.uclm.es/-/media/Files/A01-Asistencia-Direccion/A01-124-Vicerrectorado-Docencia/grados/ingenieria-informatica/documentos-oficiales/Ing-Informtica-julio-2019.ashx.

[68] You E. *Vite*. https://vitejs.dev/.

[69] You E. *Vue.js*. https://vuejs.org/.

Anexo A. Resumen de Sprints

En este anexo, se proporciona un detallado registro de los diferentes Sprints llevados a cabo a lo largo del proyecto. Cada Sprint se inicia con una reunión de planificación, en la que se establecen las distintas HU a abordar durante el Sprint, con la participación de los tutores Félix Albertos Marco (FAM) y Juan Enrique Garrido Navarro (JEGN), quienes desempeñan los roles de Scrum Master y Product Owner. Es fundamental destacar que la reunión de planificación, donde se definen los objetivos del Sprint, se lleva a cabo el mismo día que la revisión y retrospectiva del Sprint anterior.

También, es relevante señalar que la fecha de inicio de cada Sprint es considerada automáticamente como la fecha de finalización del Sprint anterior, asegurando así una continuidad temporal en el desarrollo de las iteraciones del proyecto.

Sin embargo, antes de la primera reunión de planificación y por consiguiente del primer Sprint, tuvieron lugar tres reuniones iniciales, las cuales se explican brevemente en el siguiente punto.

A.1. Fase Inicial

Antes del comienzo del proyecto, tuvo lugar una “fase inicial”, en la que se realizaron varias reuniones con el fin de conseguir una línea base para realizar el proyecto. Las diferentes reuniones que tuvieron lugar se detallan a continuación.

A.1.1. Primera reunión

Esta reunión realizada el **12 de septiembre de 2023**, al tratarse de la primera de todas, tenía como objetivo simplemente de ponerse en contacto con el tutor del trabajo FAM. Además, se plantearon diferentes ideas para el proyecto, llegando a la conclusión que el proyecto se trataría de una aplicación Web relacionada con el campo de la medicina.

A.1.2. Segunda reunión

Tras la primera reunión, el tutor se puso en contacto con Cristina Bravo, con el fin de alinear el trabajo con un proyecto real. Teniendo en cuenta esto, el objetivo principal de esta segunda reunión realizada el **15 de septiembre de 2023**, fue tomar la decisión final de lo que trataría el proyecto, el cual consistiría en una aplicación Web para diagnosticar la depresión a través de inteligencia artificial.

A.1.3. Tercera reunión

Por último, el **20 de septiembre de 2023** tuvo lugar esta tercera reunión, en la cual los diferentes integrantes y personas de interés del proyecto DiPAMIA se pusieron en contacto, con el fin de alinear los diferentes objetivos del proyecto.

A partir de esta misma reunión, daría comienzo el proyecto, lo que significaría que la siguiente se trataría de la primera reunión de planificación, y por consiguiente del primer Sprint. El cual es detallado en el siguiente punto.

A.2. Sprint 1

Este Sprint tiene comienzo el **29 de septiembre de 2023** con una duración de 2 semanas. Su objetivo principal era la obtención de conocimiento sobre el lenguaje TypeScript para familiarizarse con este mismo. A continuación, se detallan las diferentes fases del Sprint.

A.2.1. Planificación del Sprint

En esta reunión, se tomó la decisión que el proyecto tendría un mayor enfoque en el lado del FrontEnd, abstrayéndose del lado del servidor y la parte de la inteligencia artificial. Finalmente, con esto en mente, se consiguió establecer un título tentativo para el proyecto, así como un primer esbozo de los diferentes objetivos principales. A continuación, en la Tabla A.1 se detallan las diferentes HU incluidas en el PB. Por otro lado, en la Tabla A.2 se presentan las HU trasladadas al SB, indicando así su planificación para ser ejecutadas durante el desarrollo del Sprint.

Tabla A.1: PB del Sprint 1

ID	Tarea	Peso
TSGM-1	Tener primer contacto con el lenguaje “TypeScript”	3

Tabla A.2: SB del Sprint 1

ID	Tarea	Peso
TSGM-1	Tener primer contacto con el lenguaje “TypeScript”	3

A.2.2. Desarrollo del Sprint

Durante este Sprint, se tuvo el primer contacto con el lenguaje de programación “TypeScript”. Para ello, se leyó el primer capítulo del libro recomendado por el tutor: “Programming TypeScript. Making Your JavaScript Applications Scale” [14].

A.2.3. Revisión del Sprint

Tras el desarrollo del Sprint, se realizó la reunión de revisión, en la cual se habló con el tutor FAM de lo aprendido sobre el lenguaje TypeScript, y las diferencias que tiene con el lenguaje JavaScript.

A.2.4. Retrospectiva del Sprint

Una vez terminado el Sprint podemos ver el BC en la Figura A.1, el cual refleja que el progreso del trabajo realizado ha ido por detrás del planificado inicialmente. Sin embargo, a pesar de esta variación, se logra completar el trabajo dentro del plazo establecido.

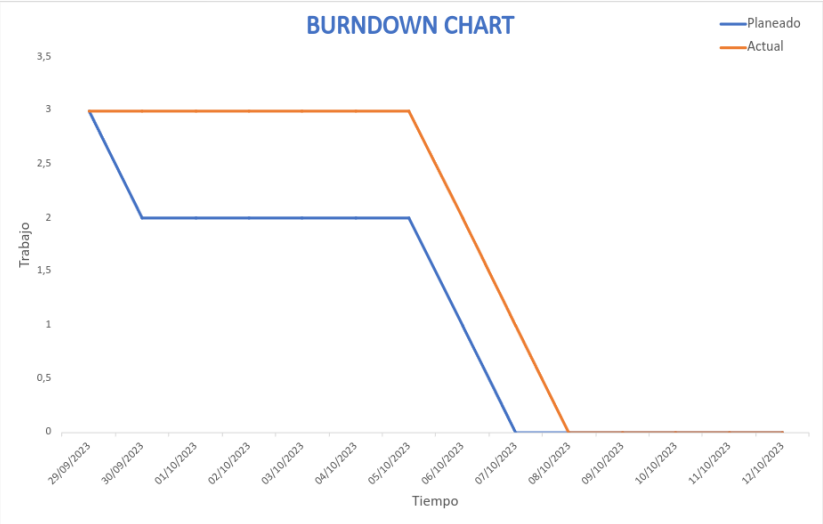


Figura A.1: BC del Sprint 1

A.3. Sprint 2

Este Sprint tiene comienzo el **13 de octubre de 2023** con una duración de 3 semanas. Su objetivo principal era familiarizarse con las diferentes librerías de visualización de datos. A continuación, se detallan las diferentes fases del Sprint.

A.3.1. Planificación del Sprint

En esta esta reunión, se concretó mejor el título del proyecto, así como de los diferentes objetivos principales. Además, se tomó la decisión final de realizar la memoria en español. A continuación, en la Tabla A.3 se detallan las diferentes HU incluidas en el PB. Por otro lado, en la Tabla A.4 se presentan las HU trasladadas al SB, indicando así su planificación para ser ejecutadas durante el desarrollo del Sprint.

Tabla A.3: PB del Sprint 2

ID	Tarea	Peso
TSGM-2	Seguir aprendiendo sobre el lenguaje TypeScript	3
TSGM-3	Usar librerías Web como Threejs y Chartjs	3

Tabla A.4: SB del Sprint 2

ID	Tarea	Peso
TSGM-2	Seguir aprendiendo sobre el lenguaje TypeScript	3
TSGM-3	Usar librerías Web como Threejs y Chartjs	3

A.3.2. Desarrollo del Sprint

Durante el Sprint, se continuó la lectura de los capítulos 2, 3 y 4 del libro de TypeScript [14]. Además, se tuvo el primer contacto con las diferentes librerías Web como Three.js para visualizaciones en 3D y Chart.js para visualizar datos en forma de gráficas. Se llegó a realizar una pequeña demo para comprender mejor el funcionamiento de estas mismas librerías empleando las guías oficiales [64] y [13].

A.3.3. Revisión del Sprint

Tras el desarrollo del Sprint, se realizó la reunión de revisión, en la cual se compartió con el tutor FAM la demo realizada en el Sprint, así como los diferentes aprendizajes obtenidos. Además se hizo una recapitulación de los distintos Sprints realizados hasta ese momento.

A.3.4. Retrospectiva del Sprint

Una vez terminado el Sprint podemos ver el BC en la Figura A.2, el cual refleja que el progreso del trabajo realizado ha coincidido con la planificación inicial.

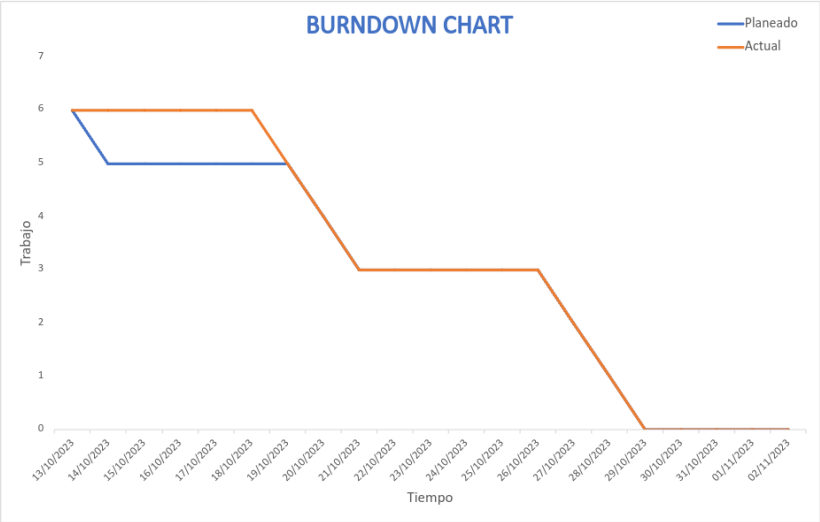


Figura A.2: BC del Sprint 2

A.4. Sprint 3

Este Sprint tiene comienzo el **3 de noviembre de 2023** con una duración de 4 semanas. Su objetivo principal era empezar la memoria realizando el primer capitulo de la misma. A continuación, se detallan las diferentes fases del Sprint.

A.4.1. Planificación del Sprint

En esta reunión, se realizó el anteproyecto, además se dio un repaso a los distintos capítulos que debería tener la memoria del proyecto. A continuación, en la Tabla A.5 se detallan las diferentes HU incluidas en el PB. Por otro lado, en la Tabla A.6 se presentan las HU trasladadas al SB, indicando así su planificación para ser ejecutadas durante el desarrollo del Sprint.

Tabla A.5: PB del Sprint 3

ID	Tarea	Peso
TSGM-4	Empezar anexo 1 sobre Sprints	3
TSGM-5	Realizar capítulo de introducción	5
TSGM-6	Realizar capítulo de estado de arte	7
TSGM-7	Realizar capítulo de propuesta de solución	7
TSGM-8	Realizar capítulo de resultados	11
TSGM-9	Realizar capítulo de evaluación	5
TSGM-10	Realizar capítulo de conclusiones	5

Tabla A.6: SB del Sprint 3

ID	Tarea	Peso
TSGM-4	Empezar anexo 1 sobre Sprints	3
TSGM-5	Realizar capítulo de introducción	5

A.4.2. Desarrollo del Sprint

Durante el Sprint, se empezó y finalizó el capítulo 1 de la memoria. Además, también se comenzó a realizar el primer anexo, el cual consistiría en el resumen de los diferentes Sprints que tuvieron lugar durante la realización del proyecto.

A.4.3. Revisión del Sprint

Tras el desarrollo del Sprint, se realizó la reunión de revisión, en la cual se conoció al segundo tutor del proyecto JEGN, y por lo tanto se hizo la recapitulación de todo el trabajo realizado para que estuviera al día.

A.4.4. Retrospectiva del Sprint

Una vez terminado el Sprint podemos ver el BC en la Figura A.3, el cual refleja que el progreso del trabajo realizado ha coincidido con la planificación inicial con algunas variaciones de tiempo.

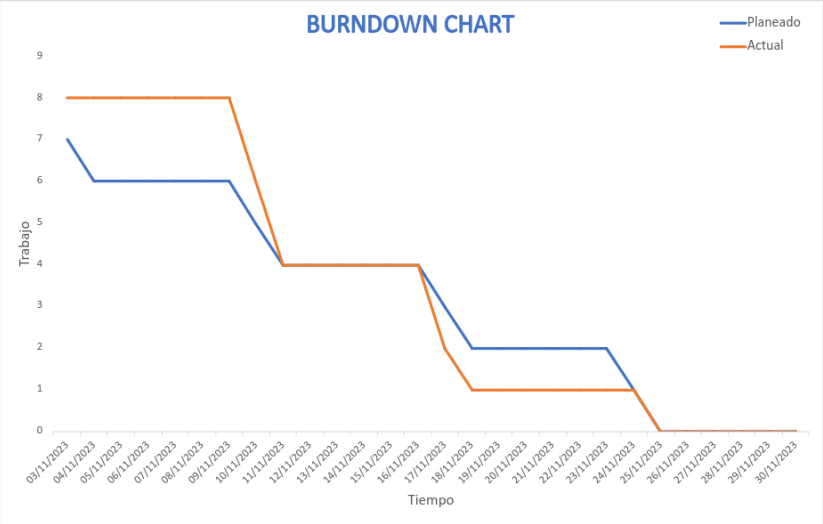


Figura A.3: BC del Sprint 3

A.5. Sprint 4

Este Sprint tiene comienzo el **1 de diciembre de 2023** con una duración de 2 semanas. Su objetivo principal era seguir progresando con la memoria, realizando el segundo capitulo de la misma. A continuación, se detallan las diferentes fases del Sprint.

A.5.1. Planificación del Sprint

En esta reunión, se pusieron las nuevas tareas para el siguiente Sprint, de las cuales, tratarán principalmente de realizar el capítulo 2 de la memoria y la adicción del siguiente Sprint al anexo. También, se habló sobre la importancia del concepto de usabilidad, el cual tendría que estar presente en todo momento durante el proyecto, lo que supone hablar sobre él en el capítulo 2 del estado de arte.

Además, se tomó la decisión que el Sprint duraría tan solo dos semanas, para no posponer la siguiente reunión después de fiestas, lo que supondría un Sprint de más de un mes de duración. A continuación, en la Tabla A.7 se detallan las diferentes HU incluidas en el PB. Por otro lado, en la Tabla A.8 se presentan las HU trasladadas al SB, indicando así su planificación para ser ejecutadas durante el desarrollo del Sprint.

Tabla A.7: PB del Sprint 4

ID	Tarea	Peso
TSGM-6	Realizar capítulo de estado de arte	7
TSGM-7	Realizar capítulo de propuesta de solución	7
TSGM-8	Realizar capítulo de resultados	11
TSGM-9	Realizar capítulo de evaluación	5
TSGM-10	Realizar capítulo de conclusiones	5
TSGM-11	Completar información de Sprints previos	3
TSGM-12	Añadir siguiente Sprint en anexo	1
TSGM-13	Compartir trabajo realizado a tutores	1
TSGM-14	Crear organización en GitHub	1
TSGM-15	Configurar “GitHub Project” para la gestión de Sprints	1

Tabla A.8: SB del Sprint 4

ID	Tarea	Peso
TSGM-6	Realizar capítulo del estado del arte	7
TSGM-11	Completar información de Sprints previos	3
TSGM-12	Añadir siguiente Sprint en anexo	1
TSGM-13	Compartir trabajo realizado a tutores	1
TSGM-14	Crear organización en GitHub	1
TSGM-15	Configurar “GitHub Project” para la gestión de Sprints	1

A.5.2. Desarrollo de Sprint

Durante el Sprint, se fueron realizando las diferentes tareas. Se decidió primero crear la organización en la plataforma GitHub, en la cual se crearán los repositorios donde se guardaría el código fuente del proyecto. También, se empezó a usar la herramienta de “GitHub Proyects” de la misma plataforma para gestionar mejor la metodología de scrum. Sin embargo, se tuvieron problemas para hacer esto último, ya que no se encontraba la manera de crear el BC, por lo que se decidió postergar la tarea para el siguiente Sprint y de esta forma poner más foco en progresar la memoria.

A parte de lo mencionado y de compartir la memoria a los tutores para que pudieran ver el progreso de esta misma. Se empezó a realizar el capítulo 2 de la memoria, el cual trataría sobre

el estado del arte. Por último, también se siguió completando el anexo de los Sprints.

A.5.3. Revisión Sprint

Tras el desarrollo del Sprint, se realizó la reunión de revisión, en la cual el tutor FAM dio el consejo de realizar la memoria en tercera persona, además de buscar ser más literario a la hora de explicar los diferentes apartados.

Por otro lado, también se discutió la manera correcta de organizar el anexo de los Sprints, en el cual de cada Sprint se detallarán por separada cada una de sus fases: planificación, desarrollo, revisión y retrospectiva. Además, se puso gran énfasis en la manera correcta de describir las diferentes HU. Las cuales deberían tener un determinado identificador, peso y un nombre que implique acción.

A.5.4. Retrospectiva del Sprint

Una vez terminado el Sprint podemos ver el BC en la Figura A.4, el cual refleja que el progreso del trabajo realizado ha ido por delante del planificado inicialmente. Además, se puede observar que no se ha quemado todo el peso de las HU debido al problema con la configuración del BC.

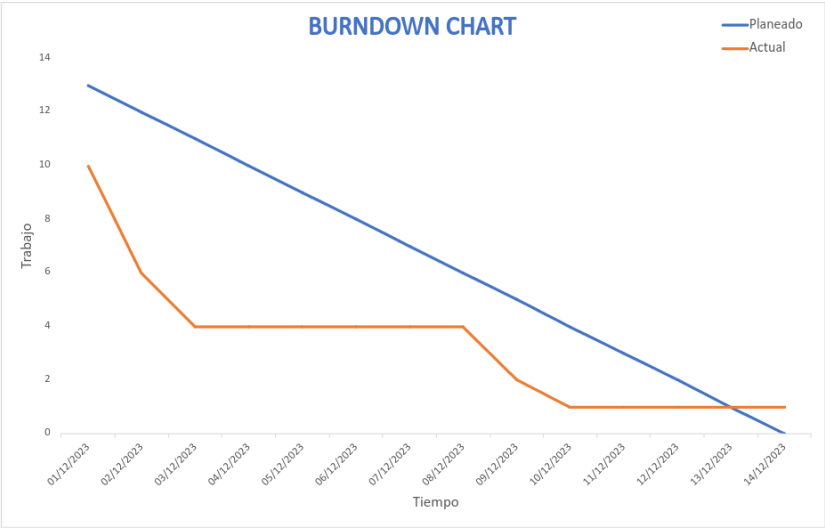


Figura A.4: BC del Sprint 4

A.6. Sprint 5

Este Sprint tiene comienzo el **15 de diciembre de 2023** con una duración de 5 semanas. Su objetivo principal era finalizar el segundo capitulo de la memoria. A continuación, se detallan las

diferentes fases del Sprint.

A.6.1. Planificación del Sprint

En esta reunión, se decidió que en el siguiente Sprint se terminaría de refinar tanto el capítulo 2 como el anexo de los Sprints, este último teniendo en cuenta lo hablado durante la reunión de revisión del anterior Sprint. Además, se empezó a discutir que en la siguiente reunión se podría realizar la primera entrevista con las personas de interés del proyecto. A continuación, en la Tabla A.9 se detallan las diferentes HU incluidas en el PB. Por otro lado, en la Tabla A.10 se presentan las HU trasladadas al SB, indicando así su planificación para ser ejecutadas durante el desarrollo del Sprint.

Tabla A.9: PB del Sprint 5

ID	Tarea	Peso
TSGM-7	Realizar capítulo de propuesta de solución	7
TSGM-8	Realizar capítulo de resultados	11
TSGM-9	Realizar capítulo de evaluación	5
TSGM-10	Realizar capítulo de conclusiones	5
TSGM-15	Configurar “GitHub Project” para la gestión de Sprints	1
TSGM-16	Refinar anexo Sprints	3
TSGM-17	Usar plantilla markdown	3
TSGM-18	Finalizar capítulo de estado de arte	3
TSGM-19	Añadir siguiente Sprint en anexo	1

Tabla A.10: SB del Sprint 5

ID	Tarea	Peso
TSGM-16	Refinar anexo Sprints	3
TSGM-17	Usar plantilla markdown	3
TSGM-18	Finalizar capítulo de estado de arte	5
TSGM-19	Añadir siguiente Sprint en anexo	1

A.6.2. Desarrollo de Sprint

Durante el Sprint, se realizaron las diferentes tareas planteadas en la reunión de planificación. Es decir, se acabó el capítulo del estado del arte y se mejoró el anexo de los Sprints. Además,

se empezó a usar la plantilla de markdown proporcionada por el tutor FAM.

Durante el uso de dicha plantilla, apareció un problema, el cual consistía en que no se encontraba el makefile correspondiente al ejecutar el comando “sudo make docker”. Esto se consiguió solucionarlo, cambiando “PWD” a “shell pwd” en la línea 24 del makefile. El archivo makefile previo al cambio se puede ver en el Listado A.1, mientras que en el Listado A.2 se puede ver el cambio realizado (en el caso de los listados, la línea 5 corresponde a la línea 24 del makefile original).

Listado A.1: Archivo makefile antes del cambio

```
1  tfg: deleteContainer
2      @echo "**********************************"
3      @echo "Compilando Trabajo Final de Grado"
4      @echo "**********************************"
5      -docker run -v "$(shell pwd)/templateAPP":/home/tfgii/templateAPP --name
           ↪ tfgii felix.albertos/tfgii
6  deleteContainer:
7      @echo "********************"
8      @echo "Borrando contenedor"
9      @echo "********************"
10     -docker rm tfgii
```

Listado A.2: Archivo makefile después del cambio

```
1  tfg: deleteContainer
2      @echo "**********************************"
3      @echo "Compilando Trabajo Final de Grado"
4      @echo "**********************************"
5      -docker run -v "$(PWD)/templateAPP":/home/tfgii/templateAPP --name tfgii
           ↪ felix.albertos/tfgii
6  deleteContainer:
7      @echo "********************"
8      @echo "Borrando contenedor"
9      @echo "********************"
10     -docker rm tfgii
```

A.6.3. Revisión Sprint

Tras el desarrollo del Sprint, se realizó la reunión de revisión, en la cual los tutores dieron algunos comentarios sobre la memoria poniendo especial atención en la escritura y el formato de la misma. Además, el tutor JEGN dio el consejo de añadir los diferentes PB en el anexo de Sprints con el objetivo de dar un mejor contexto a las HU que se seleccionan.

Por otro lado, se discutió la forma de realizar el BC utilizando la herramienta GitHub Projects, ya que está estaba dando problemas para crear dicho gráfico. Al final, se llego a la conclusión de hacer el burnup chart en vez del BC para no perder más tiempo con esta tarea.

Finalmente, se acordó modificar el enfoque del proyecto, pasando de diagnosticar la depresión a diagnosticar la fibromialgia.

A.6.4. Retrospectiva del Sprint

Una vez terminado el Sprint podemos ver el BC en la Figura A.5, el cual refleja que el progreso del trabajo realizado ha ido ligeramente por detrás del planificado inicialmente. Sin embargo, a pesar de esta variación, se logra completar el trabajo dentro del plazo establecido.

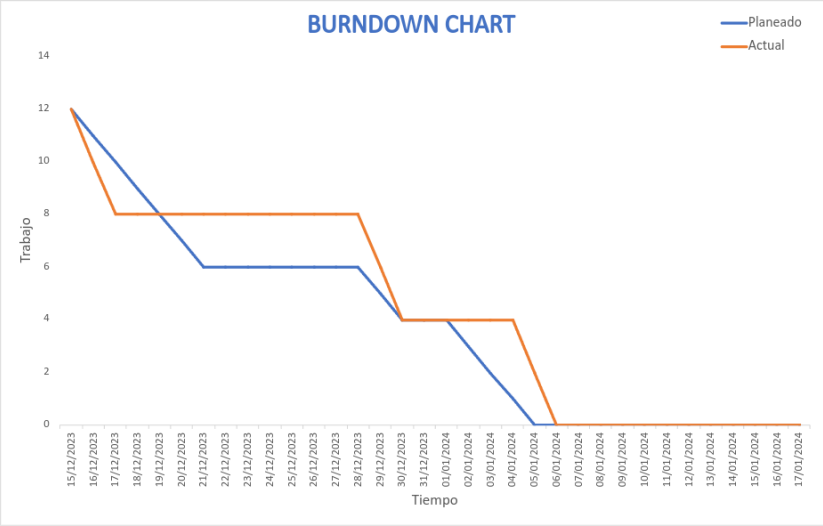


Figura A.5: BC del Sprint 5

A.7. Sprint 6

Este Sprint tiene comienzo el **19 de enero de 2024** con una duración de 2 semanas. Su objetivo principal era realizar el capítulo 3 de la memoria. A continuación, se detallan las diferentes fases

del Sprint.

A.7.1. Planificación del Sprint

En esta reunión, se planificaron las tareas a realizar durante el siguiente Sprint, acordando realizar el capítulo de propuesta de solución así como realizar las mejoras a la memoria según la revisión realizada anteriormente. Además, se discutió de una posible entrevista con Cristina para la siguiente reunión. A continuación, en la Tabla A.11 se detallan las diferentes HU incluidas en el PB. Por otro lado, en la Tabla A.12 se presentan las HU trasladadas al SB, indicando así su planificación para ser ejecutadas durante el desarrollo del Sprint.

Tabla A.11: PB del Sprint 6

ID	Tarea	Peso
TSGM-7	Realizar capítulo de propuesta de solución	7
TSGM-8	Realizar capítulo de resultados	11
TSGM-9	Realizar capítulo de evaluación	5
TSGM-10	Realizar capítulo de conclusiones	5
TSGM-15	Configurar “GitHub Project” para la gestión de Sprints	1 > 3
TSGM-20	Realizar cambios según revisión	5
TSGM-21	Crear el repositorio para la memoria	1
TSGM-22	Añadir el PB en el anexo de Sprints	3
TSGM-23	Ver que técnicas usar según test usuarios	5
TSGM-24	Preparar entrevista con Cristina	5
TSGM-25	Crear el burndown chart en Excel	5
TSGM-26	Añadir siguiente Sprint en anexo	1

Tabla A.12: SB del Sprint 6

ID	Tarea	Peso
TSGM-7	Realizar capítulo de propuesta de solución	7
TSGM-15	Configurar “GitHub Project” para la gestión de Sprints	3
TSGM-20	Realizar cambios según revisión	5
TSGM-21	Crear el repositorio para la memoria	1
TSGM-22	Añadir el PB en el anexo de Sprints	3
TSGM-23	Ver que técnicas usar según test usuarios	5

ID	Tarea	Peso
TSGM-24	Preparar entrevista con Cristina	5
TSGM-25	Crear el BC en Excel	5
TSGM-26	Añadir siguiente Sprint en anexo	1

A.7.2. Desarrollo de Sprint

Durante el Sprint, se llevaron a cabo las diversas HU planificadas en el Sprint Backlog, incluyendo los cambios propuestos en la última reunión de revisión y la redacción del capítulo de propuesta de solución, entre otras tareas detalladas en la Tabla A.12.

Es importante destacar que, en este período, se configuró finalmente el proyecto dentro de GitHub Project, incorporando todas las HU realizadas hasta ese momento. Además, se tomó la decisión definitiva de elaborar el BC utilizando la herramienta de Excel proporcionada por *Microsoft*. Esta elección se debió a las dificultades encontradas al intentar crear el gráfico mediante GitHub Project. Con el uso de Excel, se logró crear una plantilla siguiendo los pasos que explica Swan en [63].

A.7.3. Revisión Sprint

Tras el desarrollo del Sprint, se realizó la reunión de revisión, en la cual …

A.7.4. Retrospectiva del Sprint

Una vez terminado el Sprint podemos ver el BC en la Figura A.7.4, el cual refleja…

BC del Sprint 6

A.8. Sprint 7

Este Sprint tiene comienzo el **2 de febrero de 2024** con una duración de X semanas. Su objetivo principal era […]. A continuación, se detallan las diferentes fases del Sprint.

A.8.1. Planificación del Sprint

En esta reunión, …

A continuación, en la Tabla **??** se detallan las diferentes HU incluidas en el PB. Por otro lado, en la Tabla **??** se presentan las HU trasladadas al SB, indicando así su planificación para ser

ejecutadas durante el desarrollo del Sprint.

A.8.2. Desarrollo de Sprint

Durante el Sprint, …

A.8.3. Revisión Sprint

Tras el desarrollo del Sprint, se realizó la reunión de revisión, en la cual …

A.8.4. Retrospectiva del Sprint

Una vez terminado el Sprint podemos ver el BC en la Figura A.8.4, el cual refleja…

BC del Sprint 7