UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

ANÁLISIS DE RENDIMIENTO DE LENGUAJES DE DESARROLLO DE APLICACIONES Y SERVICIOS WEB

VÍCTOR GALLEGO IZQUIERDO

Director/a: SILVIA LLORENTE VIEJO (Departamento de Arquitectura de Computadores)

Titulación: Grado en Ingeniería Informática (Tecnologías de la información)

Memoria del trabajo de fin de grado

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

Resumen

Hoy en día existen multitud de lenguajes de programación para distintos propósitos. Uno de ellos es el desarrollo de aplicaciones y servicios web. La diversidad de opciones disponibles en el mercado actual presenta un desafío y una oportunidad para los desarrolladores, ya que deben sopesar factores cruciales como la facilidad de desarrollo, la eficiencia y el rendimiento.

Este proyecto se embarca en una comparación y evaluación de diversos lenguajes de programación, con el objetivo de proporcionar una visión detallada y completa. En un escenario donde la evolución es constante y donde la demanda de aplicaciones y servicios web robustas y eficientes, comprender las fortalezas y debilidades de cada lenguaje se vuelve cada vez más esencial.

Para poder analizar los lenguajes, se desarrollará una serie de aplicaciones en cuatro de los lenguajes estudiados a lo largo del proyecto y sobre el cual se realizará una serie de pruebas para evaluar su rendimiento y eficiencia de desarrollo.

Se realizará un análisis de los distintos lenguajes de programación que se encuentran en el mercado actual para poder determinar cuáles son los más adecuados, todo ello desde el punto de vista del desarrollo web del lado del servidor (backend). Las principales características por comparar son el tipo de lenguaje, el rendimiento, la facilidad de despliegue, utilización por parte de los desarrolladores del sector TI y características de seguridad nativas.

Resum

Avui dia hi ha multitud de llenguatges de programació per a diferents propòsits. Un és el desenvolupament d'aplicacions i serveis web. La diversitat d'opcions disponibles al mercat actual presenta un desafiament i una oportunitat per als desenvolupadors, ja que han de sospesar factors crucials com ara la facilitat de desenvolupament, l'eficiència i el rendiment.

Aquest projecte s'embarca en una comparació i una avaluació de diversos llenguatges de programació, amb l'objectiu de proporcionar una visió detallada i completa. En un escenari on l'evolució és constant i on la demanda d'aplicacions i serveis web robustes i eficients, comprendre les fortaleses i les debilitats de cada llenguatge esdevé cada vegada més essencial.

Per poder analitzar els llenguatges, es desenvoluparà una sèrie d'aplicacions en quatre dels llenguatges estudiats al llarg del projecte i sobre el qual es realitzaran una sèrie de proves per avaluar-ne el rendiment i l'eficiència de desenvolupament.

Es realitzarà una anàlisi dels diferents llenguatges de programació que es troben al mercat actual per poder determinar quins són els més adequats, tot això des del punt de vista del desenvolupament web del costat del servidor (backend). Les principals característiques per comparar són el tipus de llenguatge, el rendiment, la facilitat de desplegament, la utilització per part dels desenvolupadors del sector TI i les característiques de seguretat natives.

Abstract

Today there are many programming languages for different purposes. One of them is the development of web applications and services. The diversity of options available in today's market presents a challenge and an opportunity for developers as they must weigh crucial factors such as ease of development, efficiency and performance.

This project embarks on a comparison and evaluation of various programming languages, with the aim of providing a detailed and complete overview. In a scenario where evolution is constant and where the demand for robust and efficient web applications and services, understanding the strengths and weaknesses of each language becomes increasingly essential.

In order to analyze the languages, a series of applications will be developed in four of the languages studied throughout the project and on which a series of tests will be carried out to evaluate their performance and development efficiency.

An analysis of the different programming languages found in the current market will be carried out to determine which ones are the most suitable, all from the point of view of server-side (backend) web development. The main characteristics to compare are the type of language, performance, ease of deployment, use by developers in the IT sector and native security features.

Agradecimientos

Quiero agradecer a mi directora del proyecto, Silvia Llorente por el tiempo, consejos y dedicación en la realización de este TFG, han sido imprescindibles para poder hacerlo lo mejor posible.

También me gustaría agradecer a mis padres, hermano, amigos y familia en general por el apoyo y comprensión durante todo el tiempo. Han sido esenciales para poder continuar y acabar las asignaturas y el proyecto del grado. Gracias por formar parte.

Índice

Índice de figuras

Índice de tablas

1. Introducción y contexto

1.1 Introducción

En un mundo en el que cada vez consumimos más tecnología y estamos cada vez más conectados, el desarrollo web se ha convertido en una de las disciplinas más importantes de la informática. Como es un mundo en constante evolución, con nuevas tecnologías y tendencias, es importante mantenerse actualizado con aquellas que supongan una mejora en el campo, por lo que este estudio se centrará en identificar y analizar las mejores prácticas del desarrollo web para estar actualizado y ser competitivo.

1.2 Contextualización

Este Trabajo de Final de Grado (TFG) forma parte del Grado en Ingeniería Informática (GEI) con especialización en Tecnologías de la Información que se imparte en la Facultat d'Informàtica de Barcelona (FIB) de la Universitat Politécnica de Catalunya (UPC), realizándose en el primer cuatrimestre del curso 2023-2024. El proyecto está realizado en modalidad A, siendo dirigido por Silvia Llorente Viejo.

El proyecto pretende realizar un estudio comparativo entre los diferentes lenguajes de programación que existan en el mercado, determinando cuál es mejor para desarrollar aplicaciones y servicios web con criterios de rendimiento, facilidad de aprendizaje y despliegue, utilización en el sector TI y características de seguridad nativas.

1.3 Terminología y definiciones

En este apartado se hará una pequeña explicación de algunos conceptos o términos importantes que son recurrentes a lo largo del TFG, para así poder entenderlo de la forma más clara posible:

- **Aplicación web:** Una aplicación web es un software que se ejecuta en un servidor remoto. Para acceder a las funcionalidades de esta no es necesario instalar un programa en el sistema, si no que con disponer de un navegador web ya es suficiente para acceder [1].
- **Servicio Web:** Un servicio web es un software diseñado para que dos o más aplicaciones se comuniquen entre sí a través de protocolos de red, ya sean de manera local o a través de Internet [2]. Generalmente se suele utilizar el protocolo HTTP, o su equivalente seguro HTTPS, para la comunicación entre los servicios web.

- **HTTP/HTTPS:** HTTP, *Hypertext Transfer Protocol* o Protocolo de Transferencia de Hipertexto, es un protocolo de la capa de aplicación creado para transferir información entre dispositivos conectados en red. Este se ejecuta sobre otros protocolos de las capas inferiores que forman los protocolos de red [3].

- **API:** API, *Application Programming Interfaces* o Interfaz de Programación de Aplicaciones, son un conjunto de protocolos y funciones creadas para que dos aplicaciones o servicios web se comuniquen entre ellos sin necesidad de saber cómo están implementados, esto hace que se simplifique el trabajo entre estos [4].

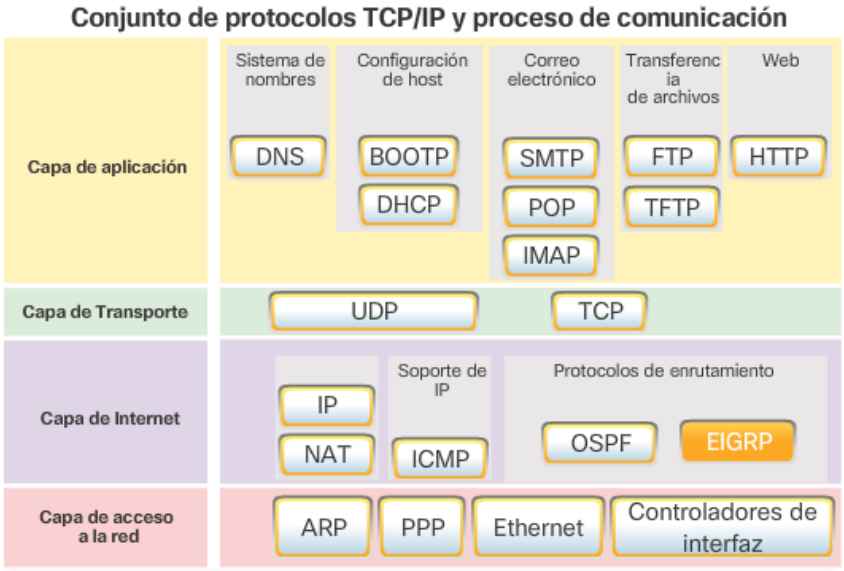

FIGURA 1: PILA DE PROTOCOLOS TCP/IP

Fuente: https://platzi.com/

1.4 Identificación del problema

Hoy en día, el desarrollo de aplicaciones web se ha convertido en algo tan importante para la informática, que las estamos continuamente utilizando para casi cualquier tarea de nuestro día a día. Por ello, la creación de aplicaciones y servicios web eficientes y funcionales es crucial para satisfacer las demandas del público y de las empresas que cada vez hacen más uso de estas.

El problema que surge de esta necesidad, es decidir qué tipo de lenguaje es el adecuado para cada aplicación. Por ello, es necesario realizar un estudio previo al desarrollo de estas, y no solo tener en cuenta la preferencia de los programadores, ya que puede desembocar en una aplicación con rendimiento o características incorrectas.

Por este motivo, este TFG se centra en hacer una comparativa entre los distintos lenguajes principales que existen en el mercado hoy en día y poder determinar para qué es válido cada uno de estos y así aprovechar estos conocimientos para poder aplicarlos en aquellas asignaturas del grado dedicadas al desarrollo de aplicaciones y servicios web.

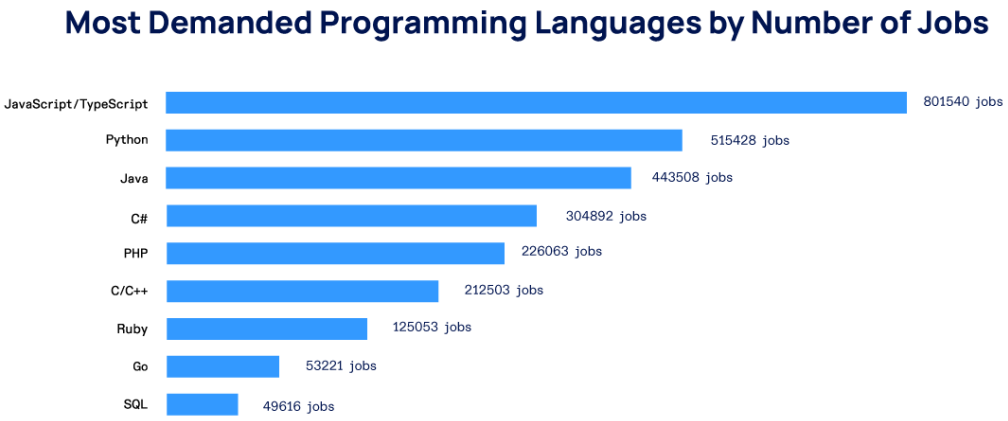

FIGURA 2: LENGUAJES DE PROGRAMACIÓN MÁS USADOS 2023

Fuente: Los lenguajes de programación más demandados en 2023 | HACK A BOSS

2. Justificación

La utilidad de este proyecto es permitir a los profesores de asignaturas de la especialidad de Tecnologías de la Información, específicamente a la asignatura de Aplicaciones Distribuidas, conocer en profundidad las tendencias de los lenguajes de programación de desarrollo web y sus marcos de trabajo actuales. Esto permitirá poder tomar decisiones para el futuro de la asignatura y así actualizarse con las últimas novedades.

2.1 Soluciones existentes

Existen lenguajes de todo tipo y con características específicas para realizar diferentes tareas, como pueden ser lenguajes que permitan gestionar directamente memoria, lenguajes específicos para sistemas operativos, lenguajes de propósito general, etc. Por lo que esto será la principal característica.

2.2 Decisión final

Dado que existe un gran número de lenguajes en el mercado actual, además de unos pocos de Frameworks por cada lenguaje, existen cientos de estos que podríamos tomar como ejemplo. En este proyecto nos basaremos en aquellos lenguajes más conocidos, y no por ejemplo en algún tipo de lenguaje o framework experimental desarrollado por algún miembro de la comunidad o estudiante. Para ello, utilizaremos una encuesta anual realizada a la comunidad de programadores [5].

Es importante remarcar que no todos los lenguajes de programación que existen hoy en día pueden estar dedicados al desarrollo web, por lo que nos tendremos que quedar con aquellos que sí dispongan.

3. Alcance del proyecto

En esta sección del proyecto se establece el alcance del proyecto, es decir, se determinan los objetivos principales, así como los objetivos derivados de estos o subobjetivos que se tienen que cumplir para que el estudio sea correcto. Finalmente, se realiza un análisis de los posibles riesgos o dificultades que pueden surgir a lo largo del desarrollo del proyecto, principalmente para tenerlo en cuenta y así poder minimizar su impacto en el resultado final.

3.1 Objetivo principal

El objetivo principal de este proyecto es hacer una investigación entre los diferentes lenguajes de programación que existen en el mercado actual relacionados con el desarrollo de aplicaciones y servicios web. Además de tener en cuenta los diferentes lenguajes, se conocerán los frameworks que son más usados y que facilitan el trabajo al programador.

En segundo lugar, se realizará un estudio de cada uno de los lenguajes encontrados para conocerlos en más profundidad y determinar cuáles son los cuatro que más nos interesarán. De todos los encontrados, nos quedaremos con cuatro para continuar con el estudio del proyecto.

Una vez escogidos los lenguajes, se desarrollará una aplicación Web y una API REST para cada uno de estos lenguajes analizados. Esto servirá para comprobar y realizar las pruebas necesarias. Estas aplicaciones tendrán las mismas características en cuanto a funcionalidades de cara al usuario. Una vez estén desarrolladas, se usarán contenedores para facilitar el despliegue de manera local o en un servidor. Las aplicaciones consistirán en un manejo de imágenes y vídeos, en las que se guardarán ciertos parámetros, que se podrán realizar consultas y modificaciones. En el caso de las imágenes, estas se podrán subir y se quedarán guardadas. Además, existirá una opción para poder cifrarlas en el servidor. Por otro lado, los vídeos no se subirán al servidor si no que se guardarán los enlaces de vídeos de la aplicación YouTube.

Por último, una vez desarrolladas las aplicaciones, se realizará la comparativa final con todos los datos obtenidos y con la experiencia de estudio y desarrollo de las aplicaciones. Para ello, se tendrá en cuenta el rendimiento de estas, utilización del lenguaje en el sector de TI, características nativas, facilidad en cuanto al aprendizaje y despliegue en un entorno lo más real posible y características de seguridad nativas.

3.2 Competencias técnicas

Considerando los objetivos del TFG, las competencias técnicas que se consideran adecuadas y que se han escogido para este proyecto son:

- CTI1.1: Demostrar comprensión del entorno de una organización y sus necesidades en el ámbito de las tecnologías de la información y las comunicaciones
- CTI2.3: Demostrar comprensión, aplicar y gestionar la garantía y la seguridad de los sistemas informáticos (CEIC6).
- CTI3.1: Concebir sistemas, aplicaciones y servicios basados en tecnologías de red, incluyendo Internet, web, comercio electrónico, multimedia, servicios interactivos y computación ubicua.

3.3 Requisitos

3.3.1 Requisitos funcionales

- **Permitir a los usuarios controlar la sesión:**
    - Tener un registro, inicio y cierre de sesión para poder acceder a las funcionalidades de las aplicaciones.
- **Permitir a los usuarios el manejo de imágenes:**
    - Poder registrar imágenes con los parámetros requeridos y con el archivo de la imagen.
    - Modificar los parámetros y el archivo.
    - Eliminar la imagen del sistema.
    - Consultar la imagen a través de una búsqueda con algunos de los parámetros de la imagen.
    - Listar todas las imágenes.
    - Visualizar la imagen.
    - Cifrar y descifrar el archivo del servidor.
- **Permitir a los usuarios el manejo de los vídeos:**
    - Poder registrar vídeos con los parámetros requeridos.
    - Modificar los parámetros.
    - Eliminar el vídeo del sistema.
    - Consultar vídeos a través de una búsqueda con algunos de los parámetros introducidos
    - Listar todos los vídeos.
    - Visualizar el vídeo con un reproductor.

3.3.2 Requisitos no funcionales

Estos requisitos no funcionales, son características que tienen que tener las aplicaciones finales para garantizar el correcto funcionamiento de estas:

- **Usabilidad:** Las aplicaciones tienen que ser usables para todo tipo de usuarios que la puedan usar.
- **Disponibilidad y fiabilidad:** Las aplicaciones tienen que estar activas siempre que los usuarios lo necesiten.
- **Escalabilidad:** El nombre de usuarios, imágenes y vídeos puede tener un gran incremento, por lo que la aplicación tiene que soportar este crecimiento.
- **Integridad y privacidad:** Las aplicaciones tienen que garantizar que los datos no son modificados por otro actor que no sea el propio usuario.

3.4 Riesgos y posibles obstáculos

En esta última sección del alcance del proyecto, hay que plantear y tener en cuenta las dificultades y problemas que pueden surgir durante el proyecto, especialmente en el desarrollo de las aplicaciones y servicios web, ya que es software y pueden surgir bugs que retrasen la planificación inicial. Para minimizar el impacto sobre el trabajo, he tenido en cuenta los siguientes riesgos:

- **Limitación y gestión del tiempo:** Como el tiempo de entrega del proyecto está definido desde un primer momento y no es posible que haya ningún tipo de retraso, es imprescindible tener una buena planificación inicial de las tareas para poder tenerlo a tiempo. El no cumplir con los plazos establecidos puede hacer que la calidad del mismo sea menor o el incumplimiento de los objetivos y requisitos previstos.
- **Desviación de la planificación inicial:** Durante el desarrollo del proyecto pueden surgir nuevos objetivos o modificaciones de estos que hagan que compliquen la planificación inicial del proyector, ya que estás modificaciones se tendrían que añadir o cambiar.
- **Nuevas tecnologías:** Al realizar una investigación sobre diferentes lenguajes de programación, es muy probable que tenga que trabajar con tecnologías no vistas durante el grado, hecho que implica una curva de aprendizaje, sobre todo tiempo para poder tener los conocimientos necesarios para poder desarrollar con estas.
- **Errores y bugs:** Como se trata de desarrollar con código, implica que surjan problemas como son lo bugs de las aplicaciones. La gravedad de los bugs hará retrasar en mayor o menor medida la planificación inicial, por esto hay que tener en cuenta algo de tiempo a la hora de planificar el desarrollo de las aplicaciones a solucionar estos bugs, aunque no se puede saber con exactitud.

4. Metodología y rigor

4.1 Metodología

Escoger una buena de metodología de trabajo y herramientas para realizar el trabajo de la forma más correcta y fácil es algo básico, especialmente si se trata de realizar una investigación y desarrollo de software. En este caso, se usará una metodología de proyecto iterativo. Se dividirá el proyecto en pequeños módulos independientes. Cuando acabe la implementación del primer módulo, se integrará con los anteriores y se pasará al siguiente y así hasta completar todos.

El método iterativo se divide en las siguientes fases [6]:

- **Fase de preparación y requisitos:** Durante esta fase se crea una hoja de ruta del proyecto que esté alineada con los objetivos definidos previamente. En esta etapa se definen los requisitos esenciales que se tienen que cumplir para que el proyecto finalice con éxito.
- **Fase de análisis y diseño:** Una vez definidos los objetivos, se empezarán a crear los primeros diseños para cumplir las ideas planificadas. Se establece la lógica a seguir durante el proyecto.
- **Fase de implementación:** En esta tercera fase, es donde realmente se creará la primera iteración del proyecto. Se siguen las especificaciones y el diseño previamente establecidos con el objetivo de cumplir los objetivos de desarrollo.
- **Fase de pruebas:** Una vez se ha completado la primera iteración, se realizan las pruebas necesarias para comprobar que todo funciona correctamente y recopilar los datos necesarios. Esta fase ayuda a identificar posibles errores o partes en las que hay que realizar mejoras.
- **Fase de evaluación y revisión:** Con todas las pruebas finalizadas, se evalúa el éxito o no de la iteración y se centra en aquellas debilidades encontradas en la fase previa de evaluación. En caso de haber algún tipo de problema mayor, se puede volver a iniciar la iteración de nuevo para poner solución. Es importante destacar, que, en el proceso iterativo, se aprenden los errores cometidos tanto en el diseño como en la ejecución de la iteración para que estos no se vuelvan a producir en las siguientes.

Además de esta metodología de trabajo, se programarán reuniones periódicas con la tutora del TFG. En estas reuniones se tratará el progreso que se ha producido desde la reunión anterior, los problemas que han ido surgiendo durante el estudio y desarrollo del proyecto y la planificación hasta la siguiente reunión, sobre todo para controlar que el avance del TFG va cumpliendo con los plazos y la calidad necesaria.

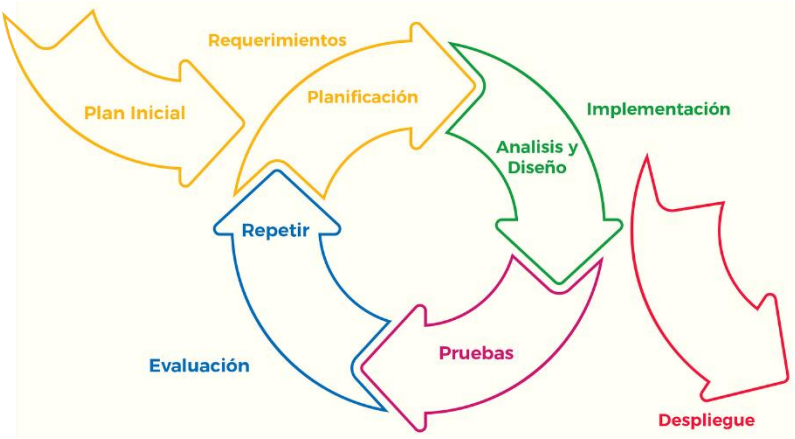

FIGURA 3: FASES DEL MÉTODO ITERATIVO

4.2 Herramientas de seguimiento

Para finalizar este apartado, es importante nombrar las herramientas que se utilizarán para el desarrollo correcto del proyecto para gestionar y organizar de la mejor forma posible el progreso de este. Las herramientas usadas son:

- **Git y GitHub [7]:** Para gestionar el código de las aplicaciones se usará Git, una herramienta de control de versiones que facilita el desarrollo más organizado mediante el uso de "ramas". Además de Git, se usar GitHub, que es una plataforma que permite tener repositorios de Git y gestionarlos mediante una interfaz gráfica.
- **TIckTick [8]:** Esta es una aplicación de tipo ToDoList y Kanban que permite añadir tareas y organizarlas a través de distintos filtros como pueden ser por prioridades, fecha de finalización y etiquetas. Además, permite tomar notas mediante el lenguaje de escritura *Markdown* [*9*].
- **Visual Studio Code [10]:** Es un editor de texto propietario de Microsoft en el que se puede desarrollar con cualquier tipo de lenguaje. Es ampliamente usado por la comunidad. La gran ventaja de este, es que dispone de muchas extensiones que permiten añadir características que no vienen por defecto.
- **Syncthing [11]:** Esta es una aplicación que permite la sincronización de carpetas entre diferentes dispositivos, de manera que los cambios en uno se apliquen en cualquiera de los otros sincronizados. Es una plataforma Open Source disponible para la mayoría de sistemas operativos. Esta correrá en los dispositivos usados a lo largo del trabajo, además de en un servidor propio que será donde se almacene y sincronice los cambios.
- **Suite Google:** Estas son las últimas herramientas que se usarán. Principalmente serán Google Meet y Gmail para la comunicación y reuniones con la directora del proyecto. Gmail se usará como comunicación escrita a través del mail. En Google Meet se programarán y se harán las reuniones a través de videollamada.

5. Planificación temporal

La planificación temporal de un proyecto es uno de los elementos principales de la gestión de proyectos, ya que permite tener clara una organización, paralelización y optimización de todas las tareas y recursos que forman el TFG, todo ello para tener una mejor calidad final.

5.1 Identificación de recursos

En este apartado se explicarán todos los tipos de recursos necesarios que se utilizarán para que el proyecto pueda ser realizado correctamente. Identificar correctamente los recursos utilizados durante el proyecto, ayuda a que las tareas que se tienen que realizar durante el transcurso de este, estén más organizadas. Los recursos los dividiremos en tres tipos:

5.1.1 Recursos humanos

El principal recurso humano del proyecto es el responsable de llevarlo a cabo, en este caso yo como autor del propio TFG. Otros recursos humanos importantes para el proyecto será la directora del TFG y la tutora del curso GEP, ambos participando en la supervisión de este.

5.1.2 Recursos materiales

En cuanto a recursos materiales, el primero y obligatorio, es disponer de un ordenador con conexión a Internet y de conexión a la red eléctrica. En este caso se usará un portátil con Windows 11 y Linux con distribución Manjaro (Arch Linux) para poder realizar todas las tareas y objetivos previstos del trabajo. A este, se le añadirán diversos periféricos externos como un segundo monitor, teclado y ratón. Finalmente, algo importante a destacar es disponer de un buen espacio en el que poder trabajar cómodo y tranquilamente.

5.1.3 Recursos de software

Respecto a los recursos software, podemos encontrar diferentes subtipos:

- **Recursos necesarios para documentación y gestión:** Para realizar documentación se usará la suite de Microsoft Office, más concretamente Word para redactar todo lo necesario y Excel como hoja de cálculos del proyecto. Para la gestión del proyecto se usará TickTick como tablero Kanban y listado de tareas.
- **Recursos de comunicación:** Para poder comunicarme con la directora del TFG, así como con la tutora del GEP, usaremos Gmail para comunicación escrita y Google Meet para comunicación mediante videoconferencia y poder llevar a término todas las reuniones de seguimiento y consulta de dudas. También se usará Atenea para entregas de GEP correspondientes a cada bloque.
- **Recursos para el desarrollo o implementación:** Para mantener un control de versiones sobre el código desarrollad, se usará Git y GitHub como repositorio, todo desde línea

de comando. Para desarrollar el código, se usará Visual Studio Code como editor de texto. Para poder desplegar las aplicaciones, ya sea de manera local o en un servidor, se usará la plataforma de contenedores Docker. Para guardar los datos durante el desarrollo, se usará MySQL de forma local, aunque en las implementaciones finales, la base de datos también estará en contenedores Docker. En cada lenguaje se usará el correspondiente servidor en un contenedor Docker.

5.2 Estimación y descripción de las tareas

El proyecto está planificado para durar como máximo 540 horas. La fecha de inicio se determina desde el 18 de septiembre de 2023 hasta la fecha de lectura del mismo, aproximadamente la semana del 22 al 26 de enero de 2024. El proyecto tendrá una duración estimada en 520 horas, teniendo en cuenta que está destinado a durar 540 horas, habrá un margen de 20 horas para posibles imprevistos o mejoras.

A continuación, se proporcionan todas las tareas que se van a realizar. Se muestran de forma ordenada y en bloques según correspondan, además de una descripción y aproximación de las horas que se van a dedicar.

5.2.1 Gestión del proyecto

En este bloque se recogen todas las actividades relativas a la gestión del proyecto, estas son la documentación, preparación previa y estilización del proyecto. Cuenta con una dedicación total de 65 horas.

- **[GP1] - Contextualización y alcance**: Consiste en la elaboración de la primera parte del proyecto. En este se especifica la justificación del porqué se está realizando este, los objetivos que se tienen, así como la metodología a seguir a lo largo del desarrollo. Se calcula una dedicación de 25 horas.
- **[GP2] – Planificación temporal:** Consiste en la preparación temporal para la ejecución del TFG, donde se definen las fases, tareas y la estimación total de horas. Consta de 20 horas.
- **[GP3] – Gestión económica y sostenibilidad:** En esta parte se define el presupuesto inicial para poder realizar el proyecto y la sostenibilidad de este. Consta de 20 horas.

5.2.2 Investigación del proyecto

En este segundo bloque, se recogerán todas las tareas que tengan que ver con la investigación, es decir, lectura de artículos/reportes/papers, visualización de vídeos y consulta general en cualquier tipo de fuente que esté relacionada con el tema. Se calcula una duración total de 50 horas para este bloque.

- **[IP1] - Definición del Estado del Arte**: El Estado del Arte es un término que se refiere al conocimiento que se tiene sobre el tema estudiado, como introducción al trabajo que se está realizando. Esto servirá para poder entender el desarrollo web en la actualidad y cuáles son las limitaciones actuales. Consta de 25 horas.
- **[IP2] - Estudio del mercado actual**: Esta tarea consiste en realizar la investigación necesaria del mercado actual de los lenguajes de programación más usados actualmente, junto con un framework adecuado. Una vez encontrados los lenguajes más adecuados, se escogerán cuatro. Sobre estos, se realizará un estudio más profundo de cómo funciona, características principales y sintaxis del lenguaje para poder desarrollar con él. Consta de 25 horas.

5.2.3 Desarrollo del proyecto

Es el bloque más importante del proyecto. Recoge toda la parte relacionada con la preparación, programación de las aplicaciones y servicios web y despliegue de estas, además de las pruebas necesarias para que funcionen correctamente. Se calcula un total de 360 horas.

- **[DP1] - Lenguaje escogido 1:**
    - **[DP1.1] - Instalación de requisitos para ejecución**: Consiste en descargar el intérprete o compilador, según el lenguaje, configuración de librerías y framework para poder programar correctamente. Tendrá una duración de 5 horas.
    - **[DP1.2] - Desarrollo de objetivos**: Primera implementación de los requisitos que tienen que tener la aplicación y el servicio web de este lenguaje. Se calcula una duración de 60 horas.
    - **[DP1.3] - Evaluación del desarrollo**: En esta parte se testearán todas las funciones desarrolladas anteriormente para comprobar el correcto funcionamiento y encontrar todos los posibles errores cometidos durante la programación de las aplicaciones. Tendrá una duración de 10 horas.
    - **[DP1.4] - Corrección de errores y mejoras**: Consiste en implementar las mejoras y correcciones encontradas en el apartado anterior. Tendrá una duración de 15 horas.
- **[DP2] - Lenguaje escogido 2:**
    - **[DP2.1] - Instalación de requisitos para ejecución**: Consiste en descargar el intérprete o compilador, según el lenguaje, configuración de librerías y framework para poder programar correctamente. Tendrá una duración de 5 horas.
    - **[DP2.2] - Desarrollo de objetivos**: Primera implementación de los requisitos que tienen que tener la aplicación y el servicio web de este lenguaje. Se calcula una duración de 60 horas.

- **[DP2.3] - Evaluación del desarrollo**: En esta parte se testearán todas las funciones desarrolladas anteriormente para comprobar el correcto funcionamiento y encontrar todos los posibles errores cometidos durante la programación de las aplicaciones. Tendrá una duración de 10 horas.
- **[DP2.4] - Corrección de errores y mejoras**: Consiste en implementar las mejoras y correcciones encontradas en el apartado anterior. Tendrá una duración de 15 horas.

- **[DP3] - Lenguaje escogido 3:**
  - **[DP3.1] - Instalación de requisitos para ejecución**: Consiste en descargar el intérprete o compilador, según el lenguaje, configuración de librerías y framework para poder programar correctamente. Tendrá una duración de 5 horas.
  - **[DP3.2] - Desarrollo de objetivos**: Primera implementación de los requisitos que tienen que tener la aplicación y el servicio web de este lenguaje. Se calcula una duración de 60 horas.
  - **[DP3.3] - Evaluación del desarrollo**: En esta parte se testearán todas las funciones desarrolladas anteriormente para comprobar el correcto funcionamiento y encontrar todos los posibles errores cometidos durante la programación de las aplicaciones. Tendrá una duración de 10 horas.
  - **[DP3.4] - Corrección de errores y mejoras**: Consiste en implementar las mejoras y correcciones encontradas en el apartado anterior. Tendrá una duración de 15 horas.
- **[DP4] - Lenguaje escogido 4:**
  - **[DP4.1] - Instalación de requisitos para ejecución**: Consiste en descargar el intérprete o compilador, según el lenguaje, configuración de librerías y framework para poder programar correctamente. Tendrá una duración de 5 horas.
  - **[DP4.2] - Desarrollo de objetivos**: Primera implementación de los requisitos que tienen que tener la aplicación y el servicio web de este lenguaje. Se calcula una duración de 60 horas.
  - **[DP4.3] - Evaluación del desarrollo**: En esta parte se testearán todas las funciones desarrolladas anteriormente para comprobar el correcto funcionamiento y encontrar todos los posibles errores cometidos durante la programación de las aplicaciones. Tendrá una duración de 10 horas.
  - **[DP4.4] - Corrección de errores y mejoras**: Consiste en implementar las mejoras y correcciones encontradas en el apartado anterior. Tendrá una duración de 15 horas.

5.2.4 Finalización del proyecto

La tercera y última parte del desarrollo del TFG es la relacionada con toda la documentación final y la preparación de la defensa en modo exposición oral delante de un tribunal. Cuenta con una dedicación de 45 horas.

- **[FP1] - Integración de la documentación en la memoria:** Incluye toda la preparación de la memoria final, integrando todos los documentos de las entregas parciales de GEP, así como la documentación relacionada con la investigación y justificación del desarrollo del TFG. Cuenta con una dedicación aproximada de 20 horas.
- **[FP2] - Preparación de la defensa:** Consta de la preparación de los recursos necesarios para la presentación oral, esto incluye una pequeña demostración del funcionamiento de las aplicaciones, así como todos los recursos visuales para la defensa del TFG. Tendrá una duración de 25 horas.

5.3 Planificación y diagrama de Gantt

Una vez determinadas todas las tareas que se van a realizar a lo largo del proyecto y toda la estimación de tiempos, se describirá la realización de estas de una manera más sencilla. Para ello, se usará un diagrama de Gantt, el cual se puede encontrar a continuación. Sirve para mostrar la planificación de una forma más visual y secuencial del desarrollo de todas las tareas que se tienen que llevar a término.

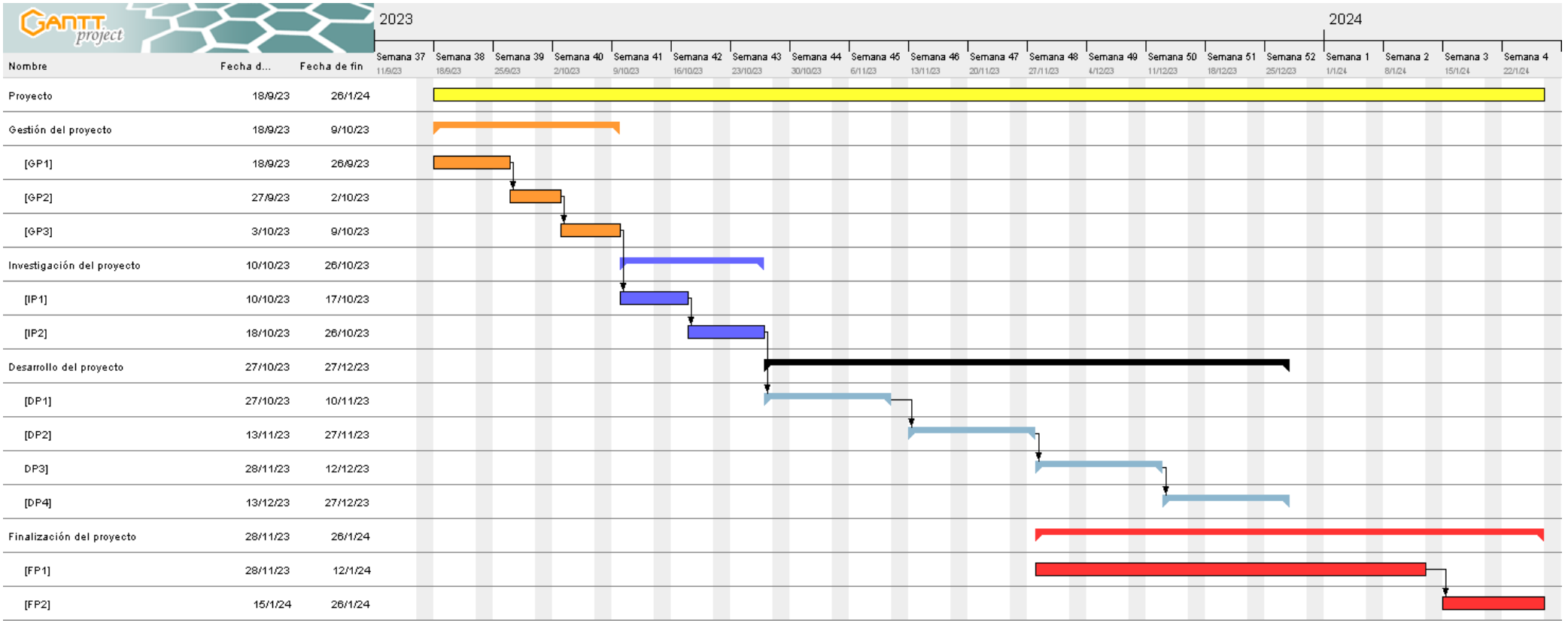

FIGURA 4: DIAGRAMA DE GANTT

FUENTE: ELABORACIÓN PROPIA

5.4 Resumen de la planificación

Una vez están listadas y explicadas todas las tareas, en la siguiente tabla se presenta un resumen de estas, indicado su ID, que tarea es, la dedicación y los recursos necesarios para llevarlas a cabo.

ID	Tarea	Dedicación	Recursos
Gestion del Proyecto		65 horas	
GP1	Contextualización y alcance	25 horas	Word
GP2	Planificación temporal	20 horas	Word, Excel, TickTick
GP3	Gestión económica y sostenibilidad	20 horas	Word, Excel
Investigación del Proyecto		50 horas	
IP1	Definición del Estado del Arte	25 horas	Word
IP2	Estudio del mercado actual	25 horas	Word
Desarrollo del Proyecto		360 horas	
DP1	Lenguaje escogido 1	90 horas	
DP1.1	Instalación de requisitos	5 horas	-
DP1.2	Desarrollo de objetivos	60 horas	Visual Studio Code, Git, GitHub
DP1.3	Evaluación del desarrollo	10 horas	Visual Studio Code, Git, GitHub
DP1.4	Corrección de errores y mejoras	15 horas	Visual Studio Code, Git, GitHub
DP2	Lenguaje escogido 2	90 horas	
DP2.1	Instalación de requisitos	5 horas	-
DP2.2	Desarrollo de objetivos	60 horas	Visual Studio Code, Git, GitHub
DP2.3	Evaluación del desarrollo	10 horas	Visual Studio Code, Git, GitHub
DP2.4	Corrección de errores y mejoras	15 horas	Visual Studio Code, Git, GitHub
DP3	Lenguaje escogido 3	90 horas	
DP3.1	Instalación de requisitos	5 horas	-
DP3.2	Desarrollo de objetivos	60 horas	Visual Studio Code, Git, GitHub
DP3.3	Evaluación del desarrollo	10 horas	Visual Studio Code, Git, GitHub
DP3.4	Corrección de errores y mejoras	15 horas	Visual Studio Code, Git, GitHub
DP4	Lenguaje escogido 4	90 horas	
DP4.1	Instalación de requisitos	5 horas	-
DP4.2	Desarrollo de objetivos	60 horas	Visual Studio Code, Git, GitHub
DP4.3	Evaluación del desarrollo	10 horas	Visual Studio Code, Git, GitHub
DP4.4	Corrección de errores y mejoras	15 horas	Visual Studio Code, Git, GitHub
Finalización del proyecto		45 horas	
FP1	Integración de la documentación	20 horas	Word
FP2	Preparación de la defensa	25 horas	Word, PowePoint

TABLA 1: TABLA DE ESTIMACIONES DE LAS TAREAS

FUENTE: ELABORACIÓN PROPIA.

5.5 Gestión de riesgos

Una buena planificación tiene que tener en cuenta siempre todos los posibles riesgos e imprevistos que puedan hacer desviar el proyecto del programa inicial. Por este motivo se tiene que pensar y planificar un plan de respuesta a estos posibles riesgos para mitigar el impacto en el proyecto. A continuación, para cada uno de los riesgos identificados en el apartado de "Riesgos y posibles obstáculos", se analiza y se crea un plan de contingencia a este. En la siguiente tabla, se muestra para cada riesgo, la probabilidad de que suceda, el impacto que tendría y el plan de mitigación.

Riesgo	Probabilidad	Impacto	Plan de mitigación
Limitación y gestión del tiempo	Baja	Imposibilidad de finalizar el proyecto en el tiempo previsto o antes de la finalización de la fecha, que no se implementen todos los requisitos o que no sea de la calidad necesaria.	Priorizar las tareas pendientes y más importantes del proyecto.
Desviación de la planificación inicial	Media	Retraso en el desarrollo del proyecto y posible no implementación de todas las funciones.	Al usar una metodología iterativa, hasta no haber completado una de las tareas, la siguiente no empezará. La solución será hacer un recálculo de las horas dedicadas.
Nuevas tecnologías	Media	Retraso en el desarrollo de las aplicaciones o incoherencias entre el tiempo planificado y el tiempo realmente dedicado.	Recalcular las horas de las tareas siguientes.
Errores y bugs	Alta	Retraso en el desarrollo general de las aplicaciones.	Buscar una solución y replanificar las siguientes tareas.

TABLA 2: GESTIÓN DE LOS RIESGOS

FUENTE: ELABORACIÓN PROPIA

5.6 Desviaciones de la planificación inicial

Como se comenta más adelante en el análisis de rendimiento de las aplicaciones y servicios web, el tiempo de desarrollo de las aplicaciones ha variado algo con respecto a la planificación inicial. El tiempo de tres de los lenguajes ha sido menor, siendo PHP el que más se ha desviado de la planificación inicial. En la tabla 1, se puede ver como todos los lenguajes tenían un desarrollo estimado de 90 horas pero en la siguiente tabla se muestran los tiempos reales medidos. En total, la desviación ha sido de 17,2 horas.

Desarrollo del Proyecto		342,8
DP1	Python y Flask	84,7
DP1.1	Instalación de requisitos	3,6
DP1.2	Desarrollo de objetivos	61,2
DP1.3	Evaluación del desarrollo	7,8
DP1.4	Corrección de errores y mejoras	12,1
DP2	JavaScript y Express	95,1
DP2.1	Instalación de requisitos	4,5
DP2.2	Desarrollo de objetivos	66,1
DP2.3	Evaluación del desarrollo	8,9
DP2.4	Corrección de errores y mejoras	15,6
DP3	PHP y Laravel	75,3
DP3.1	Instalación de requisitos	1,2
DP3.2	Desarrollo de objetivos	52,3
DP3.3	Evaluación del desarrollo	10,2
DP3.4	Corrección de errores y mejoras	11,6
DP4	Java y Spring	87,7
DP4.1	Instalación de requisitos	3,2
DP4.2	Desarrollo de objetivos	59,1
DP4.3	Evaluación del desarrollo	11,1
DP4.4	Corrección de errores y mejoras	14,3

TABLA 3: TABLA DE TIEMPOS DE DESARROLLO

FUENTE: ELABORACIÓN PROPIA

6. Gestión económica del proyecto

Una vez realizada la planificación temporal del proyecto, junto con todas las tareas y objetivos del proyecto que garantizarán el buen desarrollo del proyecto, se puede elaborar un presupuesto inicial. En este apartado se va a llevar a cabo una determinación y estimación de los costes, así como un control de gestión de las posibles desviaciones del presupuesto inicial que pueden ir apareciendo durante el desarrollo del proyecto.

6.1 Presupuesto

El primer paso para poder hacer el presupuesto, es listar todos los tipos de recursos que suponen algún tipo de gasto. Estos pueden ser de tres tipos, para este proyecto, como son los recursos humanos, materiales o generales.

6.1.1 Recursos humanos

El proyecto será únicamente desarrollado por una persona. Es por esto, que tendré que asumir diferentes roles a lo largo del TFG, según la tarea que haya que desarrollar y los objetivos. Los roles que se identifican con los siguientes:

- **Jefe de Proyecto (PM)**: Es el encargado de la organización, administrar los recursos, controlar el buen funcionamiento del proyecto y de la entrega.
- **Desarrollador (D)**: Realiza las tareas de programador, implementando los objetivos definidos de las aplicaciones y servicios web.
- **Tester de Código (T)**: Asegura que los requisitos definidos en el proyecto se cumplan en la implementación. Además, desarrolla y crea los test sobre el software creado para comprobar el correcto funcionamiento de este.

Una vez identificado los roles de las diferentes personas, el siguiente paso es el de los costes por hora que tendrá cada rol del proyecto. Para encontrar el salario de cada uno de estos, se ha optado por buscarlos a través de la web *Glassdoor* [12]. Esta página permite hacer comparaciones de los sueldos introducidos en el buscador, así como mostrar el sueldo medio de la posición deseada. En este caso utilizaremos el sueldo medio de cada uno de los roles necesarios para el proyecto en la región de Barcelona. Para hacer el cálculo por hora de cada uno, asumiremos un total de 1.800 horas de jornada laboral anual.

En la siguiente tabla los encontraremos el rol de cada uno, el salario anual (medio) sin Seguridad Social, el salario anual con Seguridad Social y el salario por hora con seguridad social. La Seguridad Social se calcula incrementando el precio del salario en un 30%.

Rol	Salario Anual	Salario/Hora	Salario/Hora + SS
Jefe de Proyecto (PM)	38.000	21,11	27,44
Desarrollador (D)	32.500	18,06	23,47
Tester de Código (T)	28.000	15,56	20,22

TABLA 4: SALARIOS POR ROL

FUENTE: ELABORACIÓN PROPIA

Una vez tenemos los precios por hora de cada rol, los aplicaremos al total de actividades. En la siguiente tabla se muestra la estimación del coste total respecto a los recursos humanos.

ID	Tarea	Dedicación	Rol	Coste
Gestion del Proyecto		65 horas		1.783,89
GP1	Contextualización y alcance	25	PM	686,11
GP2	Planificación temporal	20	PM	548,89
GP3	Gestión económica y sostenibilidad	20	PM	548,89
Investigación del Proyecto		50 horas		1.372,22
IP1	Definición del Estado del Arte	25	PM	686,11
IP2	Estudio del mercado actual	25	PM	686,11
Desarrollo del Proyecto		360 horas		8.287,50
DP1	Lenguaje escogido 1	90 horas		2.071,88
DP1.1	Instalación de requisitos	5	D	117,36
DP1.2	Desarrollo de objetivos	60	D	1.408,33
DP1.3	Evaluación del desarrollo	10	D/T	218,47
DP1.4	Corrección de errores y mejoras	15	D/T	327,71
DP2	Lenguaje escogido 2	90 horas		2.071,88
DP2.1	Instalación de requisitos	5	D	117,36
DP2.2	Desarrollo de objetivos	60	D	1.408,33
DP2.3	Evaluación del desarrollo	10	D/T	218,47
DP2.4	Corrección de errores y mejoras	15	D/T	327,71
DP3	Lenguaje escogido 3	90 horas		2.071,88
DP3.1	Instalación de requisitos	5	D	117,36
DP3.2	Desarrollo de objetivos	60	D	1.408,33
DP3.3	Evaluación del desarrollo	10	D/T	218,47
DP3.4	Corrección de errores y mejoras	15	D/T	327,71
DP4	Lenguaje escogido 4	90 horas		2.071,88
DP4.1	Instalación de requisitos	5	D	117,36
DP4.2	Desarrollo de objetivos	60	D	1.408,33
DP4.3	Evaluación del desarrollo	10	D/T	218,47
DP4.4	Corrección de errores y mejoras	15	D/T	327,71
Finalización del proyecto		45 horas		1.235,00
FP1	Integración de la documentación	20	PM	548,89
FP2	Preparación de la defensa	25	PM	686,11
Total		**520**		**12.678,61**

TABLA 5: TABLA DE COSTES HUMANOS

FUENTE: ELABORACIÓN PROPIA

6.1.2 Recursos materiales

Durante el desarrollo, se requiere de un conjunto de maquinaria para poder realizar el proyecto. En este caso, como solo hay un miembro, los costes serán aplicados únicamente a una persona, pero en caso de que fueran realmente las tres personas, asumiendo cada una de los roles, tendríamos que multiplicar por tres los costes. La siguiente tabla hace referencia al coste de los productos y de su amortización durante el proyecto. Asumiremos que cada año se trabaja unas 1.800 horas, la amortización se calcularía de la siguiente manera:

$$\frac{Coste(€) * TiempodelTFG}{Vidaútil * Díaslaborablesalaño * dedicaciónaldía(horas)}$$

Tendremos 220 días laborables al año, 6 horas de dedicación diaria y 520 horas de tiempo de duración del TFG

Dispositivo	Coste	Vida útil	Amortización
Asus ROG GX502GW	2.200€	5 años	173,33€
Monitor LG	200€	4 años	19,69€
Teclado Asus	180€	5 años	14,18€
Ratón SteelSeries	60€	3 años	7,87€
Total			215,07€

TABLA 6: TABLA DE COSTES DE RECURSOS MATERIALES

FUENTE: ELABORACIÓN PROPIA

6.1.3 Recursos de software

Todos los programas usados durante el desarrollo del proyecto son gratuitos, por lo que todos ellos tienen un coste de 0€.

6.1.4 Recursos generales

Los recursos generales son todos aquellos que no tienen relación directa con el personal ni con los recursos que forman las tares de desarrollo del proyecto. Aun así, estos recursos tienen un impacto en la gestión económica del proyecto ya que hay que tenerlos en cuenta.

Para calcular el coste de la luz, se ha tenido en cuenta un precio de 0,23479€/kWh, con un consumo medio del portátil de 200Wh, tiene un coste de 0,23479 * ((200*520)/1000) = 24,42€ en total del proyecto.

El precio de la conexión a Internet es de 4 meses, ya que es aproximadamente lo que dura y no se puede dividir el coste porque es mensual. En cuanto al espacio de trabajo, este incluye una silla, un escritorio y una lámpara.

Recurso	Precio	Coste
Internet	70€ al mes	280€
Luz	24,42€	24,42€
Espacio de trabajo	300€	300€
Total		604,42€

TABLA 7: TABLA DE COSTES GENERALES

FUENTE: ELABORACIÓN PROPIA

6.1.5 Recursos de contingencia

Estos costes de contingencia son aquellos que incluyen la previsión de costes por imprevistos que no se han tenido en cuenta en un inicio. Al ser un proyecto informático, se calcula que estos costes están entre el 10% y el 20% del total del proyecto. Tomaremos la media para calcular el total. El coste de contingencia será la suma de todos los costes multiplicado por 0,15, esto hace un total de 2.024,72€**.**

Tipo de coste	Coste total	Ratio	Coste de contingencia
Costes humanos	12.678,61€	15%	1.901,79€
Costes genéricos	604,42€		90,67€
Costes materiales	215,07€		32,26€
Total			2.024,72€

TABLA 8: TABLA DE COSTES DE CONTINGENCIA

FUENTE: ELABORACIÓN PROPIA

6.1.6 Costes por imprevistos

En este apartado se hace una previsión de costes que puedan surgir durante el desarrollo del proyecto debido a imprevistos identificados en apartados anteriores. En este caso se ha hecho una estimación en el impacto económico que repercute en el presupuesto final.

En la sección de gestión de los riesgos se explica los riesgos que pueden amenazar el desarrollo del proyecto y las acciones a tomar para solucionarlos. Todas estas acciones conllevan un coste que se tiene que añadir al presupuesto. Para realizar el cálculo, hemos tenido en cuenta una probabilidad de que suceda, el sueldo de la persona que tendría que solucionar el problema y las horas extras. La fórmula es la siguiente: Probabilidad * Sueldo * Tiempo extra

Riesgo	Probabilidad	Rol	Tiempo	Coste
Limitación y gestión del tiempo	30%	PM	5 horas	41,16€
Desviación de la planificación inicial	10%	PM	5 horas	13,72€
Nuevas tecnologías	50%	D	15 horas	58,68€
Errores y bugs	60%	T	15 horas	50,55€
Total				164,11€

TABLA 9: TABLA DE COSTES DE IMPREVISTO

FUENTE: ELABORACIÓN PROPIA

6.1.7 Estimación de costes final

A continuación, se muestra un resumen de los gastos y el total de la estimación del proyecto.

Riesgo	Probabilidad
Costes humanos	12.678,61€
Costes genéricos	604,42€
Costes materiales	215,07€
Costes software	0€
Costes generales	604,42€
Costes de contingencia	2.024,72€
Costes por imprevistos	164,11€
Total	16.291,35€

TABLA 10: TABLA DE COSTES FINALES DEL PROYECTO

FUENTE: ELABORACIÓN PROPIA

6.2 Control de gestión

Es importante definir unos mecanismos de control que nos ayuden a determinar desviaciones que puedan aparecer en relación con el presupuesto y así asegurar el cumplimiento máximo del presupuesto inicial. Para esto es importante detectar estas desviaciones, pero sobre todo el motivo por el que se producen y el coste económico que pueden tener.

- **Desviación del tiempo dedicado a una tarea**: *Horas estimadas – horas reales*
- **Desviación de costes por actividad**: *(Horas estimadas – horas reales) * coste por hora estimado*
- **Desviación de horas totales del proyecto:** *horas estimadas totales del proyecto – horas reales dedicadas al proyecto*
- **Desviación de los costes genéricos**: *coste estimado genéricos – coste real genéricos*
- **Desviación del coste de imprevistos:** *coste estimado imprevistos – coste real imprevistos*
- **Desviación total del presupuesto:** *coste estimado total del proyecto – coste real del proyecto*

7. Informe de sostenibilidad

Uno de los aspectos que se ha tratado de forma transversal a lo largo del grado de Ingeniería Informática es la sostenibilidad. Casi siempre tendemos a pensar en que está relacionado con la informática a través del hardware, ya que este es el consume los recursos de forma directa. Además, los recursos que se usan en computación y cálculo no se suelen reciclar de forma correcta ya que se tiran a la basura en vez de realizar un reciclaje o un posible reúso. En cambio, no se explica como analizar cuanto impacto tiene la utilización de estos recursos.

Otro aspecto a destacar es el ámbito social en el cual impacta. Esta sí es una parte que se enseña más, ya que la mayoría sabemos que el software que se desarrolla y se utiliza no se puede usar contra fines malignos y tiene que ser lo máximo posible de accesible. Pero no aprendemos a medir cuanto puede ser.

Por último, existe el impacto económico. En algunas asignaturas nos han enseñado a hacer cálculo de costes y mantenimiento sobre software que desarrollamos y analizar el impacto que este tendrá en el mercado. Por tanto, este es el punto de vista que más he podido ver a lo largo del grado.

7.1 Dimensión económica

Este es el aspecto más trabajado hasta ahora. He tenido que crear todo un presupuesto donde se incluyen tanto los propios recursos que se necesitan tantos costes de imprevistos y contingencia. Este es un proyecto en el que primero se realiza una investigación y un posterior desarrollo, por lo que el coste económico de encontrar la información y el posterior desarrollo de las aplicaciones es nulo. No he tenido que utilizar ninguna librería ni ningún tipo de sistema de pago.

7.2 Dimensión medioambiental

En cuanto a la investigación del proyecto, esta no provocará un gran impacto ambiental de forma directa, solo consumirá recursos por parte de donde esté alojada la información en Internet, pero no por mi parte. Por la parte de desarrollo, el consumo de estas aplicaciones será bajo ya que no estarán en continuo funcionamiento y son aplicaciones sencillas que no llegan a requerir de mucha potencia de cálculo, y, por tanto, de energía. Si lo miramos desde el punto de vista de un proyecto en general, la parte que más recursos usará es el portátil. Este ha tenido un proceso de fabricación que ha gastado bastante recursos, pero está siendo reutilizado para el proyecto, es decir, que no se ha comprado específicamente para esta tarea. Todos los recursos de hardware están siendo reutilizados.

7.3 Dimensión social

Respecto a la dimensión social del proyecto, este me aportará nuevos conocimientos y experiencia que es necesaria para mi futuro profesional, sobre todo en cuanto a desarrollo y gestión de un proyecto tan grande como puede ser este. Este será un reto que hay que asumir y que dará buenos conocimientos orientados en la gestión de proyectos, aunque se centre en una investigación y posterior desarrollo de aplicaciones web.

El hecho de realizar este estudio puede servir para que futuros alumnos del grado puedan encontrar de forma más sencilla un lenguaje de desarrollo que le resulte más cómodo. Esto puede hacer que la curva de aprendizaje se reduzca y así poder dedicar las horas necesarias para desarrollar con más rapidez y facilidad.

8. Integración de conocimientos

La finalidad principal de este proyecto es aplicar los conocimientos adquiridos durante algunas de las asignaturas del grado, especialmente asignaturas de la especialidad de Tecnologías de la Información. A continuación, se destacan las principales que más conocimientos han aportado:

- Asignaturas obligatorias:
    - Bases de Datos (BD): Esta asignatura ha aportado los conocimientos básicos y técnicos necesarios para poder trabajar con bases de datos de tipo relacional, más concretamente con lenguaje SQL para la capa de persistencia de las distintas aplicaciones desarrolladas. Se han aplicado los conocimientos adquiridos para poder crear y estructurar correctamente la base de datos de cada aplicación, teniendo en cuenta los requisitos necesarios de estas.
    - Interacción y Diseño de Interfaces (IDI): El haber cursado esta asignatura, ha ayudado a poder diseñar una interfaz práctica, usable y agradable en cuanto al punto de vista del usuario, cumpliendo con las necesidades del proyecto desarrollado.
    - Introducción a la Ingenierías del Software (IES): La asignatura me ha dado los conocimientos necesarios para poder definir aplicaciones con una arquitectura correcta para que el posterior desarrollo sea más sencillo y correcto.
- Especialidad *Tecnologías de la Información*:
    - Administración de Sistemas Operativos (ASO): Esta asignatura ha servido principalmente para tener conocimientos del sistema operativo de Linux como configuraciones avanzadas, seguridad y protección, así como gestionar los recursos necesarios de un servidor para realizar despliegues sobre estos.
    - Seguridad Informática (SI): Esta asignatura está centrada en tener conocimientos de seguridad aplicados a la informática. Al tener que aplicar patrones de seguridad sobre las aplicaciones desarrolladas, ha tenido un importante impacto el haberla cursado. Lo más importante ha sido entender las diferentes amenazas y riesgos que existen hoy en día en estos sistemas para poder evitarlos a la hora de programar o usar los distintos sistemas operativos. Las prácticas realizadas durante la asignatura han servido para aplicar técnicas de seguridad, y, sobre todo, realizar una pequeña auditoría de seguridad sobre las aplicaciones para evitar problemas con los datos necesarios para el correcto funcionamiento.
    - Aplicaciones Distribuidas: (AD): Es la principal de las asignaturas relacionadas con este proyecto. Trata sobre el conocimiento de aplicaciones y servicios distribuidos basadas en el protocolo *HTTP* y otros protocolos a nivel de aplicación. Me ha permitido tener una buena base de conocimiento sobre este protocolo para poder desarrollar aplicaciones y servicios web básicos y avanzados en lenguaje de desarrollo *Java*.

- Proyecto de Tecnologías de la Información (PTI): Junto con *AD*, ha sido la asignatura en la que he aprendido más para poder realizar el proyecto de la mejor forma posible, ya que en ella se explican bastantes tecnologías, como *Docker*, que posteriormente he tenido que usar. Además, al ser una asignatura práctica y de realizar un trabajo con un tema específico de la especialidad, he aprendido a gestionar y organizar un proyecto de gran alcance.

9. Estudio de lenguajes más usados en el mercado

9.1 Introducción

Un lenguaje de programación es un conjunto de reglas que definen cómo comunicar instrucciones a una computadora. Estas instrucciones se denominan código y son interpretadas por la computadora para realizar tareas específicas. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina para hacer algoritmos precisos o por ejemplo para permitir comunicarse entre personas. Estos lenguajes están formados por unos símbolos y reglas sintácticas y semánticas específicas de cada uno que definen una estructura. Cabe destacar que hay un error común entre las personas que es el no saber diferenciar qué es un lenguaje de programación a un lenguaje informático. El segundo es un término más general, que engloba los lenguajes de programación, como puede ser HTML. Este no es un lenguaje de programación, sino que es un lenguaje para el marcado de páginas web que está formado por un conjunto de instrucciones que permiten estructurar el contenido de las webs.

9.1.1 Paradigmas de programación

Un paradigma de programación es un conjunto de técnicas y principios que se utilizan para diseñar y desarrollar software. Cada paradigma tiene sus propias fortalezas y debilidades, y se adapta mejor a ciertos tipos de problemas.

Los paradigmas de programación más comunes son:

- **Programación imperativa o por procedimientos:** es el más usado en general, se basa en dar instrucciones al ordenador de cómo hacer las cosas en forma de algoritmos. La programación imperativa es la más usada y la más antigua, el ejemplo principal es el lenguaje assembler.
- **Programación estructurada:** Este paradigma se basa en la idea de dividir un programa en módulos o funciones que se pueden reutilizar. Las funciones se conectan entre sí mediante estructuras de control, como bucles y sentencias condicionales. se utiliza a menudo para crear programas sencillos y fáciles de entender.
- **Programación orientada a objetos:** Este paradigma se basa en la idea de que los programas se pueden construir a partir de objetos que interactúan entre sí. Los objetos tienen atributos y métodos que definen su comportamiento. Se suele utilizar para crear programas complejos y reutilizables
- **Programación funcional:** Este paradigma se basa en la idea de que los programas se pueden construir a partir de funciones que se pueden combinar para crear nuevos resultados. Las funciones no tienen efectos secundarios, es decir, no cambian el estado del programa. Se utiliza para crear programas eficientes y seguros.
- **Programación lógica:** Este paradigma se basa en la idea de que los programas se pueden construir a partir de reglas lógicas que se utilizan para resolver problemas. Su uso puede ser para crear programas que resuelvan problemas complejos.
- **Programación concurrente:** Este paradigma se basa en la idea de que los programas pueden ejecutarse simultáneamente en diferentes procesos o hilos. Se utiliza a menudo para crear programas que se ejecutan en paralelo.

- **Programación reactiva:** Este paradigma se basa en la idea de que los programas responden a eventos en tiempo real. Se utiliza a menudo para crear programas que responden a eventos en tiempo real.

Realmente, muchos lenguajes de programación no siguen solo un paradigma, si no que admiten varios. Un ejemplo es Python: es un lenguaje de programación multiparadigma que admite la programación estructurada, la programación orientada a objetos y la programación funcional.

9.1.2 Sistema de tipos de los lenguajes de programación

Los tipos de datos son una parte fundamental de los lenguajes de programación. Definen el tipo de información que puede almacenar una variable o expresión. Los tipos de datos pueden ser simples, como números o cadenas, o complejos, como estructuras de datos u objetos. El tipado es el proceso de asociar un tipo de dato a una variable o expresión. Los lenguajes de programación pueden tener diferentes tipos de tipado, que se dividen en dos categorías principales:

- **Tipado estático:** El tipo de dato se verifica en tiempo de compilación y durante la ejecución del programa no puede cambiar. Algunas de las ventajas de este tipado son que ayudan a prevenir errores del tipo de datos y los hace más seguros, ya que siempre es el mismo, pueden mejorar el rendimiento del código y facilitan la depuración del código. Sin embargo, también tienen desventajas como son la dificultad de aprendizaje y uso y la poca flexibilidad de código que ofrecen.
- **Tipado dinámico:** El tipo de variable se determina en tiempo de ejecución. Esto significa que el código se puede ejecutar sin que el compilador o interprete tenga que comprobar el tipo de dato que es. Las principales ventajas y desventajas de este tipo de tipado son justamente las contrarias al tipado estático.

Estos dos tipos son los más comunes a la hora de nombrar los diferentes tipos de lenguaje, pero a parte de estos existe otra clasificación, según la severidad de las restricciones de los tipos de datos y expresiones, que es la siguiente:

- **Tipado fuerte:** Este tipo de lenguajes tienen restricciones a la hora de usar los distintos tipos de datos. Estos no se pueden convertir entre sí si no es con una conversión explicita, es decir, en caso de mezclar dos tipos de datos, el programa mostrará un error ya que no se pueden mezclar directamente. Un ejemplo es el siguiente (Java):
  ```
  int x = 5;
  // String y = x; // Esto generará un error, se requiere una conversión explícita (siguiente línea)
  String y = Integer.toString(x); // Conversión explícita
  ```

- **Tipado débil:** Este tipo de tipado es justo lo contrario al fuerte, admite operaciones entre distintos tipos de datos. Un ejemplo es el siguiente (JavaScript):
  var x = 5;
  var y = "Hola";
  var z = x + y; // En JavaScript, la concatenación de cadenas y números puede ocurrir sin una conversión explícita
  console.log(x + y) // El resultado de esta operación será de “5Hola”

9.2 Diferentes lenguajes orientados a programación web backend

En esta sección vamos a analizar cuáles son los principales lenguajes de programación web backend que existen y, sobre todo, cuáles son los más usados por los desarrolladores a fecha de redacción del proyecto.

Los lenguajes de backend se utilizan para crear toda la lógica de páginas web que no son estáticas. La principal diferencia entre una página web estática y una que no lo es, es que una estática siempre va a mostrar el mismo contenido a todo el mundo, a menos que se modifique el código fuente, y una página web no estática mostrará contenido en función de las acciones que realice el usuario en tiempo real, datos almacenados en una base de datos u otros eventos, es decir, que existe una lógica en el lado del servidor que procesa los datos del usuario y los almacena.

9.2.1 Compilados

Un lenguaje compilado es un tipo de lenguaje de programación cuyo código fuente no se ejecuta directamente por la máquina, sino que pasa por un proceso de compilación antes de ser ejecutado. Este proceso de compilación consiste en que un programa llamado compilador, analice el código fuente del programa escrito en un lenguaje concreto, que el compilador entienda, y lo traduzca a código máquina. Una vez se ha compilado, se puede ejecutar en el ordenador y este lo entenderá perfectamente. Existen una serie de procesos que se realizan para que el ejecutable llegue a funcionar:

1. **Generación del código fuente**: El programador desarrolla el código que quiere en un lenguaje de alto nivel. Estos son lenguajes son aquellos que se caracterizan por tener una estructura semántica muy similar a la escritura de las personas, lo que permite crear algoritmos de manera más natura y en lugar de tenerlos que codificar en el lenguaje máquina directamente.
2. **Compilación**: El código se pasa a un programa llamado compilador. Este es específico del lenguaje en el que se ha desarrollado el código. A su vez, este compilador sigue una serie de pasos, divididos en dos fases:
    a. **Fase de análisis**: Durante esta fase, el compilador examina el código fuente para comprender su estructura y semántica, y luego crea una representación intermedia que se utilizará en las etapas posteriores del proceso de compilación. Esta fase consta generalmente de cuatro partes principales:

i. **Análisis del léxico (Tokenización)**: En esta etapa el código se divide en unidades más pequeñas llamadas *tokens*. Estos son pequeños fragmentos atómicos que pueden corresponder a palabras claves, identificadores, operadores y constantes del código. Esta parte se realiza para facilitar la manipulación y procesamiento del código.
ii. **Análisis sintáctico (Parser)**: Los tokens generados anteriormente se utilizan para crear una estructura jerarquizada, árbol de análisis sintáctico, para representar la estructura gramatical del código y verificar la conformidad del código y proporcionar una estructura que facilite la interpretación y generación del código intermedio (Cuarta parte).
iii. **Análisis semántico**: En esta etapa se verifica la coherencia semántica del código, esto implica analizar y asegurarse de que cumplan con las reglas semánticas. Se utiliza para detectar errores semánticos y establecer la información necesaria para las posteriores fases del proceso de compilación.
iv. **Generación de código intermedio**: Con todo lo obtenido durante las etapas anteriores, se genera un código que sirve como una representación simple y abstracta del código fuente original. Esta se utilizará para crear el código máquina y facilitar la optimización del código antes de crear el ejecutable final.

b. **Fase de síntesis**: Después de que el compilador haya analizado y comprendido el código, en esta fase se generará el código ejecutable. Consta de dos partes principales:
i. **Optimización**: Es una etapa fundamental. El compilador aplica diversas técnicas para optimizar el código intermedio, pero sin llegar a alterar el comportamiento de este. Algunas pueden ser como la reorganización del código para ejecutar menos instrucciones, propagación de constantes y otro tipo de transformaciones. Todo esto se aplica para poder mejorar el rendimiento de la ejecución del código final y disminuir el consumo de recursos.
ii. **Generación de código**: En esta etapa final, se utiliza todo lo anterior para generar código ejecutable. Puede ser un código final ya ejecutable por la propia máquina o puede ser código intermedio que más tarde se traducirá a código máquina.

3. **Enlace**: El código objeto no contiene toda la información final para ser ejecutado de manera independiente. Puede depender de bibliotecas o módulos. En esta etapa, el enlazador combina el código objeto con otras bibliotecas y genera un archivo ejecutable.
4. **Ejecución**: El usuario final ejecuta el archivo ejecutable. Este archivo contiene las instrucciones de bajo nivel que la CPU puede entender directamente.

El proceso de compilación suele ser más rápido que el proceso de interpretación, ya que el compilador solo necesita realizar el proceso de compilación una vez. Sin embargo, el proceso de compilación puede ser más complejo que el proceso de interpretación, ya que el compilador debe tener en cuenta la arquitectura del procesador y las características del lenguaje de programación.

Las principales características de los lenguajes compilados son:

- Rendimiento: Los programas compilados suelen ser más rápidos que los programas interpretados, ya que el código máquina está optimizado para el procesador específico de la computadora en la que se ejecuta.
- Eficiencia de memoria: Los programas compilados suelen ocupar menos memoria que los programas interpretados, ya que el código máquina no requiere un intérprete para ejecutarse.
- Portabilidad: Los programas compilados suelen ser menos portables que los programas interpretados, ya que el código máquina no está vinculado a una plataforma específica, pero sí tiene que tener la misma arquitectura en la que ha estado compilado. Ejemplo: un programa compilado para la arquitectura Intel x86 no es compatible con una arquitectura ARM. Se tendría que compilar de nuevo el código en una máquina de esta arquitectura para que sea compatible.

A continuación, se enseñarán los principales lenguajes compilados que pueden ser usados para desarrollo web backend y sus principales características.

Java

Java es un lenguaje de programación de propósito general, orientado a objetos y diseñado para ser independiente de la plataforma en la que se ejecute. Fue creado por *Sun Microsystems* y lanzado en 1995, en la actualidad es propiedad de Oracle Corporation y se ha convertido en uno de los lenguajes de programación más utilizados para el desarrollo de software. La filosofía central de Java es "Write Once, Run Anywhere", esto significa que el código puede ejecutarse en cualquier dispositivo que disponga de la máquina virtual de Java (JVM).

CARACTERÍSTICAS GENERALES

- **Orientación a Objetos**: Java sigue el paradigma de programación orientada a objetos, lo que implica la organización del código en clases y objetos, hecho que fomenta la reutilización y modularidad del código.
- **Portabilidad**: La capacidad de ejecutar el mismo código en diferentes plataformas sin modificaciones es una de las características distintivas de Java. Esto se logra mediante la compilación del código fuente en un formato intermedio llamado *bytecode*, que se ejecutará en la máquina virtual de Java.
- **Seguridad**: Java incorpora mecanismos de seguridad como el sistema de gestión de memoria, la revisión de límites de arrays y la verificación de *bytecode*, lo que ayuda a prevenir vulnerabilidades comunes en la programación actual.
- **Multitarea**: Java facilita la programación concurrente y paralela a través de la inclusión de conceptos como los threads, permitiendo que varias tareas se ejecuten a la vez.
- **Dinámico**: La capacidad de cargar clases de forma dinámica durante la ejecución es una característica esencial de Java, lo que permite una mayor flexibilidad y extensibilidad.

Java se diseñó para ser independiente de la plataforma, lo que significa que el código Java puede ejecutarse en cualquier sistema que tenga una JVM disponible. Esto incluye sistemas operativos como Windows, Linux, macOS y otros. Para ello es necesario instalar *Java Runtime Environment* [13].

Después de compilar el código fuente Java a *bytecode*, la JVM interpreta y ejecuta este *bytecode* en tiempo de ejecución. La existencia de la JVM facilita la portabilidad, ya que proporciona una capa de abstracción entre el código Java y el hardware que se encuentra por debajo. Esto significa que el mismo programa Java puede ejecutarse en diferentes plataformas sin necesidad de recompilación, siempre que haya una JVM disponible para esa plataforma.

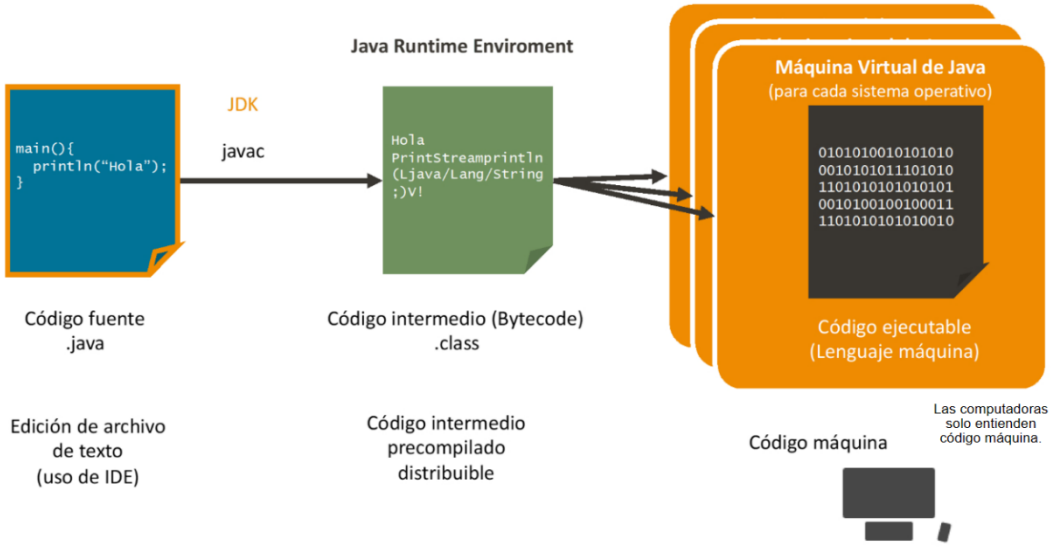

FIGURA 5: COMPILACIÓN EN JAVA

FUENTE: HISTORIA DEL LENGUAJE DE PROGRAMACIÓN JAVA (UNAM.MX)

C#

C# es un lenguaje de programación desarrollado en el año 2000 por Microsoft como parte de la plataforma .NET. Este se ha convertido en uno de los lenguajes más usados para desarrollar aplicaciones web, de escritorio y aplicaciones móviles. La sintaxis de C# es similar a la de otros lenguajes populares como C++ y Java, lo que facilita a los desarrolladores familiarizarse con la sintaxis y semántica del lenguaje.

CARACTERÍSTICAS

- **Orientado a Objetos**: C# es un lenguaje de programación orientado a objetos que permite a los desarrolladores organizar el código en clases y objetos, promoviendo la reutilización y la modularidad del código.
- **Tipado Fuerte y Estático**: C# utiliza un sistema de tipos fuerte y estático, lo que significa que las variables deben ser declaradas con un tipo específico y los errores de tipo se detectan en tiempo de compilación.
- **Gestión Automática de Memoria**: C# incorpora un recolector de basura (*garbage collector*) que gestiona automáticamente la liberación de memoria, simplificando la gestión de recursos y reduciendo el riesgo de fugas de memoria.
- **Control de excepciones**: Proporciona un enfoque estructurado y extensible para la detección y recuperación de los errores producidos durante la ejecución del código.

C# CLR

En C#, el código se escribe y se compila generando un archivo un archivo con código intermedio llamado Common Intermediate Language (CIL). Este es el lenguaje de programación de más bajo nivel pero que las personas pueden entender. Es un lenguaje ensamblador orientado a objetos y está basado en pilas. Una vez se ha creado este lenguaje intermedio, se genera el código ensamblador para que pueda ser ejecutado, se crea el archivo .exe que incluye el código y los metadatos que describen los tipos, métodos y demás propiedades. A partir de aquí entra la Common Language Runtime (CLR), es un componente esencial en el entorno de ejecución de C#. A diferencia de Java, no es una máquina virtual si no que es un tipo de compilador intermedio encargado de la ejecución del código. Este es responsable de gestionar la ejecución del código y de realizar una compilación *Just-In-Time*, el código resultante se guarda en memoria y estará listo para poder ejecutarse.

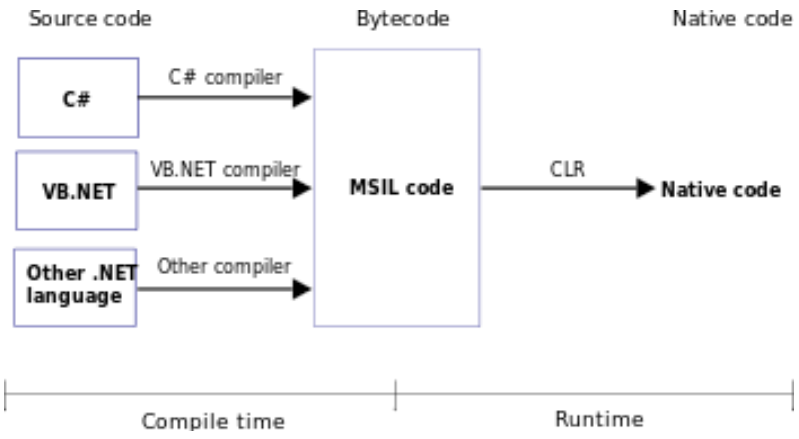

FIGURA 6: COMPILACIÓN Y EJECUCIÓN EN C#

FUENTE: WIKIMEDIA COMMONS

Esta CLR le da algunas características como las siguientes:

- Portabilidad: La CLR permite la ejecución de código C# en diferentes plataformas, proporcionando a los desarrolladores la capacidad de crear aplicaciones que pueden funcionar en entornos diversos.
- Gestión de Memoria Automática: La CLR simplifica la gestión de memoria al ofrecer un recolector de basura eficiente, liberando a los desarrolladores de la tarea manual de asignación y liberación de memoria.
- Seguridad: La CLR contribuye a la seguridad del código C# mediante la verificación y la imposición de restricciones en tiempo de ejecución, reduciendo así las vulnerabilidades asociadas con la ejecución de código no seguro.

Go

Go, también conocido como Golang, es un lenguaje de programación de código abierto lanzado por Google en noviembre de 2009. Es un lenguaje de programación compilado, concurrente, imperativo, estructurado y orientado a objetos que está disponible para diferentes tipos de sistemas y diferentes arquitecturas de procesadores. A diferencia de los dos anteriores, este no necesita de ninguna máquina virtual o software intermedio para ejecutar, el compilador directamente crea el código máquina.

CARACTERÍSTICAS

- **Simplicidad y Legibilidad**: Minimiza la necesidad de paréntesis y caracteres especiales, priorizando la comprensión intuitiva.
- **Concurrencia**: Ofrece soporte nativo para la concurrencia a través de goroutines, que son unidades de ejecución ligeras que permiten la concurrencia de manera eficiente.
- **Tipado Estático y Compilación Rápida**: Es un lenguaje de tipado estático que ayuda a prevenir errores comunes durante la compilación.
- **Garbage Collection**: Utiliza un recolector de basura eficiente para gestionar la memoria automáticamente, liberando recursos no utilizados.
- **Canalización (Pipeline)**: Go favorece el diseño de programas mediante la composición de pequeñas funciones que se ejecutan concurrentemente y se comunican a través de canales. Esto promueve la construcción de programas concurrentes de manera limpia y eficiente.
- **Compilación cruzada**: La compilación cruzada es la capacidad de compilar un programa en una plataforma y ejecutarlo en otra plataforma sin necesidad de realizar cambios en el código fuente. Go facilita la compilación cruzada gracias a su diseño y a algunas herramientas integradas. Es importante tener en cuenta que la compilación cruzada puede tener limitaciones si el programa depende de bibliotecas o dependencias específicas del sistema operativo o arquitectura. Esto facilita mucho el desarrollo multiplataforma, ya que permite compilar y probar los programas en diversas plataformas sin tener que tener un tipo de máquina con cada arquitectura o sistema operativo, sobre todo cuando se compila automáticamente con CI/CD (*Continuous Integration/Continuous Delivery*). Esta es una práctica de desarrollo de software que tiene como objetivo mejorar la eficiencia y calidad del proceso de entrega del software. Está formado por dos partes:
    - **CI**: Añadir el código desarrollado por cada developer a un repositorio, el cual ejecuta tests de forma automática para detectar errores antes de poner el software en producción y garantizar la integración del código desarrollado con el ya existente.
    - **CD**: Después de realizarse la integración, se realiza el despliegue del software de forma automatizada en producción sin tener que intervenir en el proceso.
- **Manejo de errores**: Go no tiene excepciones, ya que se tiende a usar las excepciones como estructuras de control, y esto no es realmente para lo que están pensadas. Es la principal razón por la que no se han implementado. Esto favorece a que el control de errores sea más sencillo.

Rust

Rust es un lenguaje de programación desarrollado por Mozilla y lanzado en 2015. Es un lenguaje compilado, de propósito general y multiparadigma que soporta programación funcional pura, imperativa y orientada a objetos. Diseñado para ofrecer un rendimiento similar al de C y C++, pero con un enfoque en la seguridad y la concurrencia sin sacrificar la velocidad. Como Go, no necesita de máquina virtual o software intermedio para ejecutar el programa, si no que compila y ya se puede ejecutar el programa en cualquier máquina de la misma arquitectura.

CARACTERÍSTICAS

- **Ownership y Borrowing:** Rust tiene el concepto de *ownership,* donde cada parte de memoria tiene un "dueño." Los préstamos de memoria se gestionan mediante las reglas de *borrowing,* permitiendo transferir temporalmente la propiedad sin necesidad de un recolector de basura.
- **Lifetimes:** Los *lifetimes* son anotaciones que indican la duración de las referencias. Ayudan al compilador a garantizar que las referencias no superen la vida de los datos a los que apuntan, evitando problemas de acceso a memoria inválida.
- **Eliminación de Null y Referencias Nulas:** Rust no tiene el concepto de nulos. El sistema de tipos asegura que las referencias siempre apunten a datos válidos.
- **Prevención de Race Conditions**: El sistema de préstamos y el enfoque en la concurrencia sin riesgos evitan condiciones de carrera y garantizan la seguridad en entornos concurrentes.
- **Sin recolector de basura**: La ausencia de un recolector de basura elimina la sobrecarga asociada y permite un control más preciso sobre el uso de la memoria.
- **Interacción con C**: Rust se integra fácilmente con código escrito en C, permitiendo utilizar bibliotecas existentes y garantizando la compatibilidad con sistemas existentes.

9.2.2 Interpretados

Un lenguaje de programación interpretado es aquel que no necesita ser procesado por un compilador antes de ser ejecutado. Esto implica que el ordenador tiene que ser capaz de ejecutar el código línea a línea a través de un programa llamado intérprete (Parser), un programa encargado de traducir cada instrucción escrita con una semántica propia del lenguaje a código máquina. Este enfoque permite que el mismo código fuente pueda ejecutarse en diferentes sistemas, siempre y cuando dispongan del intérprete correspondiente instalado.

La fase de ejecución de un programa interpretado es la siguiente:

1. **Análisis Léxico**: En esta primera etapa, el intérprete analiza el código fuente línea por línea, identificando los tokens que lo componen. Los tokens son las unidades básicas de significado del código fuente, como palabras clave, identificadores, operadores, paréntesis, etc. El analizador léxico también puede eliminar comentarios y espacios en blanco, ya que estos son innecesarios para le ejecución del código.
2. **Análisis Sintáctico**: El Parser analizará la estructura gramatical del código fuente, comprobando que lo anterior sea correcto y no haya ningún error. El análisis sintáctico se realiza mediante un algoritmo que genera un árbol sintáctico que representa la estructura del código fuente. El árbol de sintaxis abstracta (AST) es una estructura de datos que muestra cómo están relacionadas las diferentes partes del código fuente.
3. **Generación de Código Intermedio**: Esta es una etapa que es opcional. Algunos intérpretes generan un código intermedio después del análisis sintáctico el cual es una representación del programa que puede ser más fácil de manejar durante la ejecución del propio código.
4. **Análisis semántico**: En esta cuarta parte, el intérprete analiza el significado del código fuente, comprobando que las variables tienen los tipos de datos correctos y que las operaciones se realizan de forma correcta. Se realiza mediante un algoritmo que comprueba la verificación de tipos, la resolución de nombres y otras reglas semánticas específicas del lenguaje.
5. **Ejecución**: En esta última etapa, es donde se puede ejecutar el programa desarrollado. Existen diferentes enfoques, dependiendo de cómo esté diseñado el intérprete y el lenguaje:
    a. **Evaluación Directa**: En este enfoque, el intérprete evalúa cada instrucción directamente y realiza las operaciones correspondientes.
    b. **Interpretación Just-in-Time (JIT)**: Algunos intérpretes convierten el código fuente en código máquina en tiempo real antes de su ejecución. Esto combina elementos de la compilación y la interpretación.
    c. **Interpretación Pura**: En este enfoque, el intérprete ejecuta las instrucciones del código fuente directamente sin una fase de compilación.

    Además, durante la ejecución el intérprete debe manejar los errores que puedan surgir y en algunos lenguajes, el intérprete puede encargarse de la gestión de la memoria, incluyendo la asignación y liberación de memoria para las variables y los objetos.

Como características generales de los lenguajes interpretados, se podría decir que son casi que las contrarías a la de los lenguajes compilados. Son las siguientes:

- **Portabilidad**: El código desarrollado es independiente de la arquitectura del procesador en el que se va a ejecutar. La única característica a tener en cuenta es que el intérprete esté disponible, que, como norma general, suelen estar disponibles para las arquitecturas más usadas y conocidas. Pero esta característica genera la desventaja de tener que depender del intérprete, incluso en algunos casos, de una versión específica de este.
- **Facilidad de depuración de código**: La depuración del código desarrollado suele ser más sencillo debido a que se pueden realizar cambios en el código de forma más sencilla, ahorrándonos el paso de compilar el código, y ejecutarlo de manera inmediata. Además, existe la facilidad de que el intérprete muestra directamente donde está el error directamente ya que ejecuta el código fuente directamente.
- **Eficiencia**: Los programas desarrollados con lenguajes interpretados son menos eficientes que los desarrollados con lenguajes compilados por la razón de que las instrucciones se traducen en tiempo real mientras se ejecutan y que el compilador puede aplicar técnicas de optimización.
- **Gestión de la memoria**: Los programas interpretados suelen utilizar más memoria que los programas compilados, ya que el intérprete debe almacenar el código fuente original y la traducción a lenguaje máquina.

A continuación, se enseñarán los principales lenguajes interpretados que pueden ser usados para desarrollo web backend y sus principales características.

JavaScript

JavaScript es un lenguaje [14] desarrollado por Brendan Eich de Netscape con el nombre de Mocha, el cual fue renombrado posteriormente a LiveScript, para acabar llamándose JavaScript. Apareció por primera vez en diciembre de 1995. Debido a que muchas organizaciones fueron creando sus propias versiones del lenguaje, como Microsoft con *JScript*, fue estandarizado por ECMA en 1997, creando así ECMAScript para especificar las bases del lenguaje. Actualmente sigue estando bajo este estándar el cual se actualiza regularmente.

CARACTERÍSTICAS

Es un lenguaje interpretado del tipo Just-in-Time, de propósito general, orientado a objetos, imperativo, débilmente tipado y dinámico.

- **Creación de Frameworks y bibliotecas**: A lo largo de los años, JavaScript ha ido ganando popularidad debido a que es un lenguaje de propósito general, y se ha ido usando para crear bibliotecas y frameworks de desarrollo no solo para su uso como scripting web, sino también para crear aplicaciones web completas. También fomentó el uso de JavaScript la aparición de *AJAX* (Asynchronous JavaScript and XML), una técnica que permitía la actualización de elementos o las propias páginas web sin tener

que recargarlas, hecho que hacía que la experiencia del usuario mejorara significativamente.

- **Ejecución en el lado del servidor**: JavaScript fue diseñado principalmente para ser usado en el lado del cliente, debido a lo explicado anteriormente, pero con los años fueron apareciendo nuevas implementaciones que permitieron usarlo como lenguaje para el lado del servidor. Node.js [15] es un entorno de ejecución diseñado para cualquier plataforma para la capa del servidor, principalmente.
- **Event-drive**: JavaScript es especialmente eficaz para la programación basada en eventos. Esto quiere decir que puede responder a eventos del usuario, como clics de ratón o pulsaciones de las teclas.
- **Asincronía**: Las operaciones asíncronas permiten que ciertas tareas se realicen en segundo plano sin bloquear la ejecución normal del programa. Un ejemplo de esto son las solicitudes de red que se pueden realizar a otras aplicaciones web como APIs. Esto se logra mediante *callbacks*, *promises* o *async*/*await*.
- **Acceso y Modificación del DOM**: El DOM es el Modelo de Objetos del Documento, es decir, una interfaz que representa el documento HTML o XML de forma estructurada. Que JavaScript tenga acceso al DOM, quiere decir que puede manipular en memoria la estructura de una página web y permitir cargar contenido dinámicamente, así como modificar el aspecto.

Python

Python es un lenguaje creado por Guido van Rossum y lanzado en febrero de 1991. El principal objetivo del desarrollador fue hacer un lenguaje de programación de alto nivel, con una sintaxis lo más clara posible y que fuera fácil de leer, además de que fuera eficiente en cuanto a funcionamiento y a la hora de escribir código.

CARACTERÍSTICAS

Es un lenguaje interpretado multiparadigma, está orientado a objetos, tiene programación imperativa y programación funcional, aunque no sea su principal paradigma, fuertemente tipado y dinámico. A continuación, se presentan las principales características del lenguaje:

- **Gestión de memoria**: La memoria se gestiona de forma automática a través de un recolector de basura, liberando así los objetos y variables de la memoria para no tener que hacerlo de forma manual.
- **Diferentes implementaciones**: Al ser un lenguaje tan común y usado, se han ido creando distintas versiones basadas en Python para usos concretos o para realizar optimizaciones sobre la versión original. Por ejemplo, PyPy es una implementación de Python creada con Python y que está optimizada con el tipo de compilación Just-in-Time.
- **Manejo de excepciones**: Python cuenta con un sistema robusto de manejo de excepciones que facilita la detección y gestión de errores.
- **Bibliotecas**: Python cuenta con una gran cantidad de bibliotecas estándar que van desde tipos de datos simples hasta bibliotecas especializadas en campos como

desarrollo web, protocolos de red o *data science*. Pero además de contar con estas, también existen muchas más implementadas por la comunidad, ya que es un lenguaje Open Source.

- **Metaprogramación**: Python permite la manipulación de su propia estructura y comportamiento del programa o de otros durante su ejecución. Esto quiere decir que la metaprogramación consiste en crear o modificar otros programas.
- **Lambda**: Python implementa funciones Lambda o funciones anónimas. Estas permiten realizar operaciones rápidas.

PHP

PHP (antiguamente Personal Home Page Tools y actualmente Hypertext Preprocessor) es un lenguaje creado por Rasmus Lerdfor en 1994 como un conjunto de scripts CGI (Common Gateway Interface) desarrollados en Perl para rastrear visitantes en su propia página web y guardar ciertos datos de los visitantes. Con el paso del tiempo, dos programadores implementaron de nuevo estos scripts a una base de un lenguaje de programación completo, PHP3, la primera versión de lo que actualmente usamos.

CARACTERÍSTICAS

PHP es un lenguaje multiparadigma, imperativo, funcional y orientado a objetos. A continuación, se muestran las principales características de PHP:

- **Sintaxis**: Es similar a la del lenguaje C, ya que está inspirada en este lenguaje y en Java. Es por esto por lo que aprender PHP es más fácil si ya conoces el lenguaje de programación C.
- **Integración con HTML**: Se integra fácilmente con este lenguaje de marcado, además de poder incrustar código PHP directamente en los documentos HTML.
- **Interacción con el Servidor Web**: PHP puede ejecutarse sobre un servidor, como por ejemplo Apache, o puede ejecutarse sobre línea de comandos con un intérprete independiente.
- **Manejo de sesiones y cookies**: PHP cuenta con esta característica y la hace útil en la autenticación y seguimiento de los datos de los usuarios.
- **Seguridad**: El lenguaje incluye funciones de seguridad para evitar inyecciones de código SQL y ataques XSS [16] (Cross-Site Scripting), así como otros tipos de ataques comunes encontrados en el OWASP Top 10 [17].

Ruby

El lenguaje de programación Ruby fue creado por Yukihiro "Matz" Matsumoto en Japón. Matz comenzó a trabajar en Ruby en 1993 y lanzando la primera versión al público en 1995. El objetivo principal de este lenguaje es tener una sintaxis clara y fácil de entender, así como combinar la simplicidad del lenguaje de programación Perl y Smalltalk, junto a las grandes funcionalidades de Python. A medida que fue actualizándose, también fue ganando popularidad fuera de Japón. Esto también estuvo relacionado con la aparición del gestor de paquetes *RubyGems*.

CARACTERÍSTICAS

Ruby es un lenguaje interpretado, orientado a objetos, dinámico y funcional. A continuación, se muestran algunas de las principales características del lenguaje:

- **Sintaxis legible**: Ruby destaca por su sintaxis clara y expresiva, diseñada para ser fácil de leer y escribir de cara a que los programadores no tengan graves problemas al leerla, todo pensado desde el diseño inicial del lenguaje.
- **Recolector de basura**: el recolector de basura se encarga de identificar y eliminar los objetos y variables que no están siendo referenciados en ninguna parte del programa, para así liberar la parte de memoria que ocupan.
- **Hilos de ejecución**: Además del tradicional uso de hilos que proporciona el sistema operativo, Ruby tiene una implementación de hilos propia. Permite la ejecución concurrente de varias tareas de un programa a través del uso de una clase *Thread* que facilita la creación y gestión de estos. Cada hilo tiene su propio flujo de control independiente, pero comparte el mismo espacio de direcciones y recursos con otros hilos del mismo proceso.
- **Metaprogramación**: Ruby permite que los programas escritos con Ruby pueden manipular su propia estructura y comportamiento en tiempo de ejecución.
- **Interactive Ruby Shell**: Este es un intérprete interactivo de línea de comandos que permite ejecutar código en comandos de manera rápida, principalmente para probar líneas de código sueltas y asegurarse de que funcionan correctamente.

9.3 Uso de frameworks web

Un framework web [18] es un conjunto de herramientas, estilos propios y librerías configuradas para el desarrollo de aplicaciones y servicios web de manera más ágil y para mantener el código de forma más sencilla. Estos frameworks se han convertido en básicos para cualquier programador de hoy en día ya que permiten optimizar tiempos, prestaciones y, lo más importante para las empresas, optimizar costes de desarrollo. A modo resumen, "*Los framework hacen que el desarrollador no esté continuamente "reinventado la rueda" y se centre en el problema que quiere resolver y no en la implementación de funcionalidades que normalmente son de uso común y que ya están resueltas por otros.* " [19]

A partir de conocer qué es y qué hace un framework, es necesario conocer un poco las ventajas y desventajas del uso de estos, ya que igual no es necesario usarlos en todos los proyectos. A continuación, se mostrarán.

9.3.1 Ventajas

Como he comentado antes, el principal motivo por el que se suelen usar frameworks es la optimización del tiempo que requiere desarrollar un proyecto o la parte en la que se aplique el framework. Obviamente no todas las partes de un proyecto necesitarán de un framework o habrá partes que no estén implementadas y habrá que desarrollarlas desde cero completamente, pero proporcionan una estructura predefinida que acelera el desarrollo al ofrecer soluciones para tareas comunes. Con esto, estaríamos reutilizando código, ya que se aprovecha la modularidad del código implementado en los frameworks.

Además, si se usan frameworks populares, estos seguirán las mejores prácticas y estandarizaciones de la industria, por lo que estarán más que probados y facilita la colaboración entre las diferentes personas. Esto también es importante de cara a tener documentación pública y foros donde los usuarios hacen sus preguntas a la comunidad para resolver las dudas.

Pero sin duda la más importante podría ser tener un código más limpio. Esta es la ventaja más apreciada por los desarrolladores, ya que usando frameworks es más fácil obtener un código totalmente limpio y sin código duplicado, además de que el código del framework ha sido testeado, actualizado con más frecuencia y optimizado en cuanto a rendimiento, aunque a continuación veremos que esto no siempre es así.

9.3.2 Desventajas

La curva de aprendizaje del framework es una de las desventajas encontradas. El adoptar un nuevo framework hace que los desarrolladores pierdan tiempo en formación y en resolución de dudas de sintaxis, estructura del código o en mejores prácticas, sobre todo al inicio del desarrollo.

En cuanto al diseño de la propia aplicación, el desarrollar con un framework, puede hacer que tengamos que tener alguna estructura predefinida sobre el proyecto y a lo mejor esto no se adecúa sobre él. Esto limitaría el diseño y la toma de decisiones inicial en las que se necesitaría flexibilidad, como puede ser en proyectos de investigación.

La principal desventaja encontrada sería la dependencia sobre las versiones de framework usadas en un momento determinado, ya que al ser un código que no mantienen los propios desarrolladores, podría ocurrir que en cualquier momento este quedara “deprecated” y se tuviera que implementar de nuevo la funcionalidad en la que se basa o, como se hace en algunas ocasiones, aunque no es lo recomendado, no actualizar para poder seguir usándolo con normalidad. Este hecho puede comportar problemas de seguridad ya que se mantendría en una versión antigua sin parches de seguridad.

Por último, una desventaja a tener en cuenta es el overhead producido por la capa de abstracción que implementa el framework para simplificar el desarrollo. Al final de las ventajas comentaba que al usarse un framework podría mejorar el rendimiento debido a que el código es más probado y optimizado al usarse tanto, pero también se introduce este overhead computacional que puede afectar al rendimiento. Es por eso que, si se busca una eficiencia máxima, no se debería usar un framework.

9.3.3 Conclusiones

Después de explicar las ventajas y desventajas del uso de framework para desarrollar, hay que extraer alguna conclusión. Al final, la elección entre desarrollar con o sin frameworks depende de la naturaleza y los objetivos del proyecto, así como de las preferencias y las habilidades del equipo de desarrollo encargado. En algunos casos, una combinación de ambos enfoques, aprovechando frameworks para tareas específicas y desarrollando para otras sería el mejor enfoque para un proyecto con unas características generales.

10. Contexto de las aplicaciones

El objetivo principal de este proyecto es desarrollar un conjunto de aplicaciones y servicios web con diferentes lenguajes de programación. Todas las aplicaciones web serán la misma y realizará las mismas funciones, pero cada una con una tecnología distinta.

10.1 Arquitectura de las aplicaciones

Por cada lenguaje se realizarán distintas aplicaciones web, en este caso la primera de las dos partes que conformarán cada lenguaje es una aplicación web básica formada por un frontend en HTML, usando motores de plantillas [20]. Estos leen de un fichero que está formado por código HTML normal y corriente, pero en algunas partes se inserta código en algún tipo de lenguaje, según el que se esté usando en el backend, para generar información de forma dinámica. Esta información está insertada desde la parte de la lógica del backend. Esto puede facilitar la creación de por ejemplo listas, tablas, elementos condicionales… Esta parte del frontend será común para todas las aplicaciones, exceptuando las partes de código de las plantillas en las que cada lenguaje usa la suya. Todos los archivos están implementados con un Bootstrap llamado **W3 CSS** [21], esto facilita la implementación de una mejora visual de la página web, dado que se reutiliza el código creado por alguien y que está alojado en Internet. Esto funciona añadiendo los atributos que se indican en la página web para obtener botones de distintas formas y colores, tablas con colores intercalados, ventana para mostrar opciones…

Además de tener un front en común, la base de datos será la misma para todas los lenguajes y aplicaciones ya que todas guardan la misma información. La base de datos será MySQL. Durante el desarrollo se usará una en local, pero al finalizar el desarrollo, se adaptará a una alojada en un contenedor Docker. En las siguientes figuras se puede ver el esquema interno de las tablas de la base de datos.

La tabla usuarios es la más sencilla de las tres, está formada por un campo que corresponde al nombre de usuario y otro campo que corresponde a la contraseña. El nombre de usuario tiene que ser único en todo el sistema, por eso se añade como *Primary key*.

```
create table usuarios (
        id_usuario varchar (256) primary key,
        password varchar (256)
);
```

FIGURA 7: TABLA DE USUARIOS DE LA BASE DE DATOS

FUENTE: ELABORACIÓN PROPIA

```
create table image (
        id int NOT NULL AUTO_INCREMENT,
        title varchar (256) NOT NULL,
        description varchar (1024) NOT NULL,
        keywords varchar (256) NOT NULL,
        author varchar (256) NOT NULL,
        creator varchar (256) NOT NULL,
        capture_date varchar (10) NOT NULL,
        storage_date varchar (10) NOT NULL,
        filename varchar (2048) NOT NULL,
        primary key (id),
        foreign key (creator) references usuarios(id_usuario)
);
```

FIGURA 8: TABLA DE IMÁGENES DE LA BASE DE DATOS

FUENTE: ELABORACIÓN PROPIA

```
create table video (
        id              int NOT NULL AUTO_INCREMENT,
        title           varchar (256) NOT NULL,
        description     varchar (1024) NOT NULL,
        keywords        varchar (256) NOT NULL,
        creator          varchar (256) NOT NULL,
        author          varchar (256) NOT NULL,
        link            varchar (2048) NOT NULL,
        primary key (id),
        foreign key (creator) references usuarios(id_usuario)
);
```

FIGURA 9: TABLA DE VÍDEOS DE LA BASE DE DATOS

FUENTE: ELABORACIÓN PROPIA

10.1.1 Aplicación web

Una aplicación web es un software pensado para funcionar a través del navegador web. Estas siguen una arquitectura cliente-servidor en la que los usuarios interactúan con la web y el navegador hace de cliente y se comunica con un servidor remoto que se encarga de procesar las solicitudes, realizar cálculos o consultar datos y las devuelve al navegador. La comunicación entre el cliente y el servidor se realiza a través del protocolo HTTP. Este realiza solicitudes desde el navegador web con diferentes llamadas, en este caso GET o POST, y el servidor responde con los datos necesarios o redirige a la página.

Todas las vistas implementan un control de sesiones, es decir, solo el usuario que ha hecho login correctamente puede acceder a todas las páginas de la aplicación, en algunas incluso no puede hacer eliminar o modificar si no es el propietario de la imagen o vídeo. Este control de sesiones se implementa guardado el nombre de usuario en el *Session Storage* del navegador o en una cookie para acceder a él cada vez que se llama a representar cada una de las vistas. Este permite a las aplicaciones web almacenar datos en el lado del cliente, pero con una duración de vida más corta, mientras el usuario no cierre la pestaña. Otros componentes o pestañas del navegador no pueden acceder a los datos de *Session Storage* de una pestaña diferente ni a los datos de otra ventana del mismo navegador.

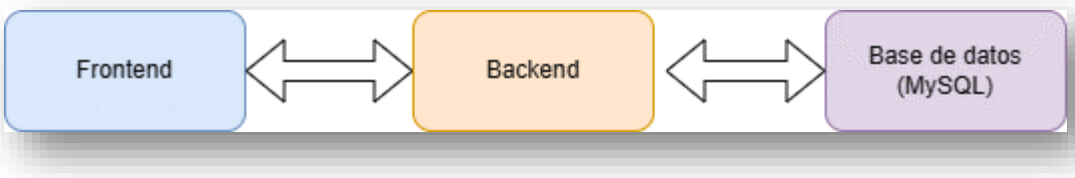

FIGURA 10: ESQUEMA DE LA APLICACIÓN WEB

FUENTE: ELABORACIÓN PROPIA

10.1.2 Aplicación web + API REST

Antes de explicar cómo será la arquitectura de esta segunda parte del desarrollo para cada lenguaje, hay que explicar que es una API [22], y en específico una API Rest [23]. Una API (Application Programming Interface) son mecanismos que permiten a dos o más componentes comunicarse entre ellos a través de un conjunto de definiciones y protocolos. Ahora bien, una API REST (Representational State Transfer) es un tipo de API con dos características principales:

- **Representational:** Se refiere a la representación de los recursos. Estos pueden ser datos, servicios o cualquier cosa que las aplicaciones necesiten representados en uno o varios formatos, como pueden ser XML o JSON.
- **State Transfer:** El principio de una API REST está basado en que es una API sin estado, es decir, toda la información tiene que estar contenida en la petición que se realiza. Esto significa que no hay ningún tipo de control de sesión por parte del lado del servidor y que estos no pueden almacenar datos relacionados con la solicitud del cliente.

Una vez explicado estos términos, lo siguiente es explicar la arquitectura y como se ha transforma la aplicación web a una arquitectura con una API REST. Esta API realizará todas las funciones que impliquen añadir, modificar, eliminar o consultar los datos guardados en disco y en la base de datos. La aplicación web simplemente recogerá los datos que el usuario introduce en el navegador web y preparará una petición HTTP llamando a la función de la API que corresponda para realizar la acción necesaria. Una vez realiza la petición, en caso de que los datos introducidos sean los correctos y que la petición se haya procesado bien, recibirá un 200 y si es el caso, los datos que devuelva la API. En caso de que esto no sea así, se recibe un código de error determinado y relacionado con el error y la aplicación web mostrará en el navegador el mensaje de error relacionado.

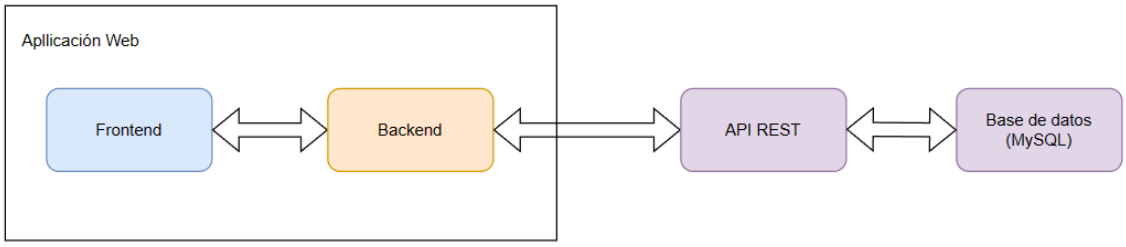

FIGURA 11: ESQUEMA DE LA APLICACIÓN WEB + API REST

FUENTE: ELABORACIÓN PROPIA

10.1.3 Despliegue sobre contenedores Docker

Docker [24] es una plataforma de contenedores que permite desarrollar, implementar y ejecutar aplicaciones. Estos contenedores son entornos aislados y ligeros, con lo mínimo necesario para que funcione, que permiten una portabilidad entre entornos, debido contiene en su interior todo lo necesario. Para hacerlo funcionar entre distintos entornos, solo hace falta tener instalado Docker en la máquina y ya se podrá hacer despliegues de los contenedores sobre esta. Existen contenedores ya creados en *Docker Hub*, pero a partir de estos podemos hacer modificaciones e integrar nuestra aplicación o servicio en su interior y así facilitar el desarrollo. En Docker tenemos que diferenciar dos conceptos que pueden llegar a ser similares si no tenemos un conocimiento básico. Las imágenes de Docker son plantillas inmutables que actúan como puntos de partida para la creación de contenedores. Los contenedores son las instancias en tiempo de ejecución de esas imágenes, que se pueden ejecutar, detener, reiniciar y eliminar según sea necesario. Las imágenes son estáticas y se utilizan como base para los contenedores, mientras que los contenedores son dinámicos y se utilizan para ejecutar aplicaciones en un entorno aislado y efímero. La figura 12 describe el proceso anterior de una manera más visual.

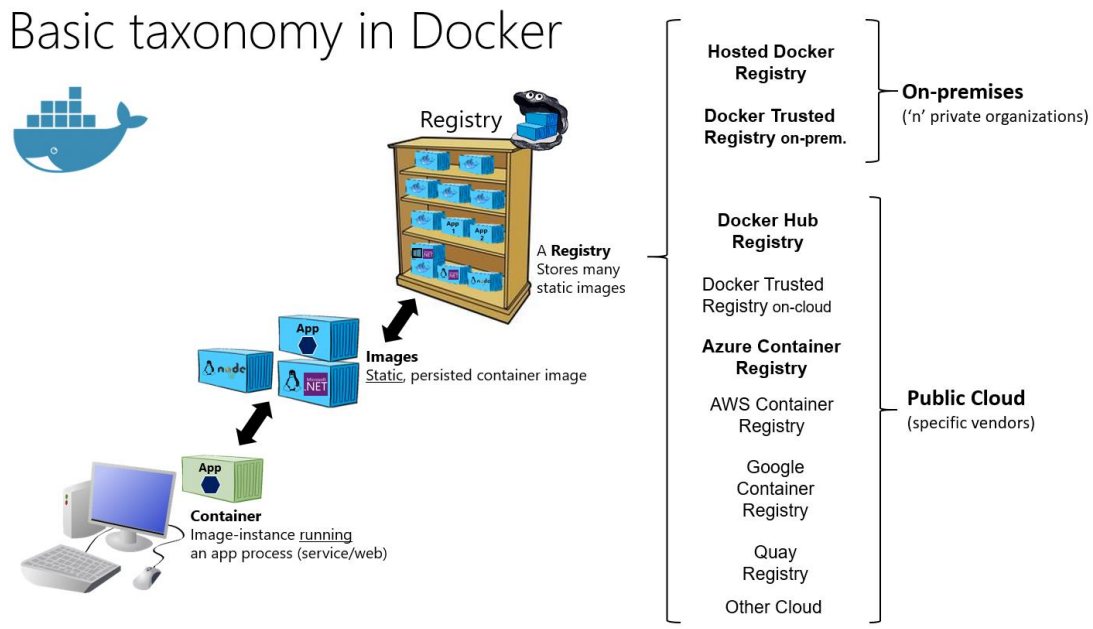

FIGURA 12: PROCESO DE CREACIÓN DE CONTENEDOR

FUENTE: CONTENEDORES, IMÁGENES Y REGISTROS DE DOCKER - .NET | MICROSOFT LEARN

En el caso de una de las aplicaciones web, tendríamos que crear un archivo Dockerfile. En este archivo se definen una serie de comandos que crean la imagen del contenedor. Para facilitar el trabajo y no tener que crearla desde cero, consultamos si existe en *Docker Hub* y a partir de aquí lo adaptaríamos a nuestra aplicación. Pongamos como ejemplo un Dockerfile de una aplicación web de Node.JS. Cada línea de la figura es un comando, siempre empezará con un **FROM** ya que es un método de creación recursivo que llamará a imágenes anteriores hasta que se encuentre con el más básico, que será la que cree una imagen de Linux u otro sistema operativo. Todos los siguientes comandos serán para instalar dependencias, librerías y copiar los archivos de nuestro proyecto. Al tratarse de una aplicación web a la que debemos tener acceso, tenemos que habilitar algún puerto para poder acceder, con el comando **EXPOSE**, se abren los puertos 8080 para HTTP y 4443 para HTTPS. Por último, **CMD**, ejecutará el comando que inicia nuestra aplicación en Node.JS.

La Figura 13 es un ejemplo de un Dockerfile para Node.JS, pero en otros lenguajes será similar, teniendo en cuenta que cada uno tiene unas librerías, llamada a ejecución distinta… pero la base es la misma.

```
FROM node:16
WORKDIR /usr/src/app

COPY package*.json ./

RUN npm install express path

COPY . .

EXPOSE 8080
EXPOSE 4443
CMD [ "node", "app.js" ]
```

FIGURA 13: EJEMPLO DE ARCHIVO DOCKERFILE

FUENTE: ELABORACIÓN PROPIA

Para poder usar estos contenedores, lo primero sería hacer un *build* y después ejecutar el contenedor. Existen dos maneras de hacer esto, una a través de un comando en el que se indica todo lo necesario o a través de un fichero “docker-compose.yml” en el que se define una estructura de todo lo necesario para poder hacer el *build* y puesta en marcha de los contenedores. Esta es una manera más sencilla ya que no tenemos que ejecutar cada contenedor uno a uno con su comando, sino que simplemente con hacer “docker compose up” en el terminal, ya se ejecutaría todo lo definido en el archivo y no habría que hacer nada más.

A continuación, en la figura 14, se muestra un ejemplo de archivo “docker-compose.yml” de la aplicación Web en Node.JS, para continuar con el ejemplo anterior. Como se puede ver, se definen dos contenedores, uno llamada **db** y otro llamado **práctica1**.

El primero corresponde a la base de datos MySQL, que descargará automáticamente de Docker Hub. Este, además de indicar el nombre y la imagen que va a usar, indica que va a

mapear un volumen. Un volumen, en Docker, es una forma de unir una carpeta que se encuentra en la máquina host con una carpeta en el interior del contenedor Docker, de manera que, si se añade, elimina o modifica algo del interior de la carpeta en el host, se modificará también en la carpeta del contenedor y viceversa. En este caso, servirá para mapear un archivo con las tablas de la base de datos a una carpeta del contenedor, la cual ejecuta cada vez que el contenedor se inicia, de esta manera tendremos la base de datos y las tablas necesarias creadas de forma automática. El comando **ports** nos permitirá mapear el puerto de la base de datos de Docker a un puerto de la máquina, en este caso la sintaxis es host:contenedor. Realmente no es necesario mapear este puerto, pero se ha hecho para poder acceder desde la consola del host de manera más sencilla. e**nv_file** se utiliza para indicar un archivo con las variables de entorno necesarias, en este caso, en Docker Hub se indicar cuáles son las mínimas necesarias para poder usar el contenedor, así como otras. En este caso, solo es necesario definir **MYSQL_ROOT_PASSWORD**, que será la contraseña del usuario root de MySQL, necesaria para poder hacer funcionar el contenedor. Por último, se añade el comando **networks** para indicar que se añadirá a la red que se define al final del archivo.

Ahora nos encontramos el contenedor que contienen la aplicación web, llamado “practica1”. Los dos comandos importantes para este contenedor son **build** y **depends_on**. Como en el contenedor de MySQL no lo creábamos nosotros, sino que lo cogíamos de Docker Hub, no teníamos que crear una imagen. En cambio, en el de la práctica1 hemos creado una imagen a partir de una anterior, por tanto, tendremos que hacer un build para crearla a partir del archivo Dockerfile. Por tanto, con el comando **build** indicaremos donde está el Dockerfile y creará la imagen personalizada. El comando **depends_on** indicará que el contenedor tendrá que iniciarse una vez se haya iniciado el indicado, en este caso el de la base de datos. Esto sirve para asegurarse que la aplicación se ha iniciado y que, si el usuario intenta hacer un login, pueda conectarse correctamente sin que suceda ningún error relacionado con la base de datos. En este contenedor se van a exponer los puertos HTTP y HTTPS para que se pueda acceder correctamente a la aplicación web desde el navegador.

```
version: "3.9"
services:
  db:
    container_name: db
    image: mysql
    volumes:
      - "./scripts/:/docker-entrypoint-initdb.d/"
    ports:
      - "3307:3306"
    env_file:
      - .env
    networks:
      my-network:

  practica1:
    container_name: practica1
    build: .
    ports:
```

```
      - "80:8080"
      - "443:4443"
    depends_on:
      - db
    networks:
      - my-network


networks:
  my-network:
    driver: bridge
```

FIGURA 14: EJEMPLO DE ARCHIVO DOCKER-COMPOSE.YML

FUENTE: ELABORACIÓN PROPIA

De esta forma queda automatizado todo el proceso para descargar la imagen de MySQL, compilar la imagen de la aplicación web, crear la red donde van a estar funcionando los contenedores, crear la dependencia entre el arranque de los contenedores y exponer los puertos para que se pueda acceder a estos.

10.2 Casos de uso

En esta sección se van a explicar los diferentes casos de uso. Estos se dividen principalmente en tres partes: casos de uso de usuarios, casos de uso de imágenes y casos de uso de vídeos.

10.2.1 Diagrama

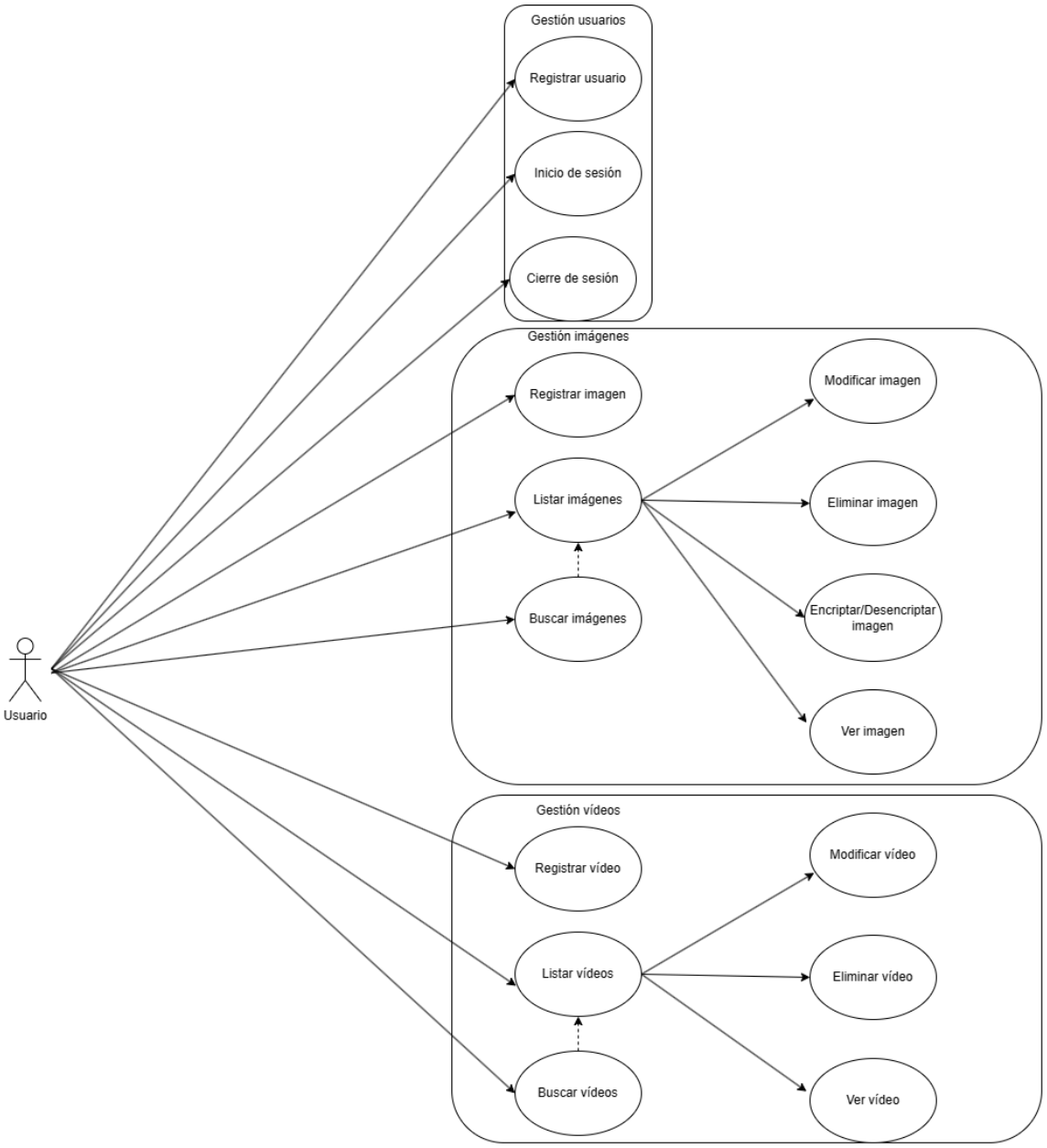

FIGURA 15: DIAGRAMA DE CASOS DE USO

FUENTE: ELABORACIÓN PROPIA

10.2.2 Descripción

Nombre del caso	Registro de usuario
Descripción	Una persona se registra en la aplicación para poder iniciar sesión en la aplicación.
Flujo básico	La persona selecciona el botón *Sign Up!* y se muestra una página en la que introducirá el nombre de usuario, una contraseña y la confirmación de esta.
Condiciones previas	
Posibles errores	Las contraseñas introducidas no coinciden, el usuario ya existe o ha sucedido algún error con la conexión de la base de datos.
Flujo alternativo	Se muestra un mensaje con el error correspondiente con un botón de *return back* que devuelve a la página anterior.
Condiciones posteriores	Se ha registrado el usuario con el usuario y la contraseña introducidos. Se redirige automáticamente a la página de inicio de sesión.

Nombre del caso	Inicio de sesión
Descripción	El usuario inicia sesión para acceder a las funcionalidades de la aplicación.
Flujo básico	El usuario inicia sesión correctamente.
Condiciones previas	El usuario tiene que estar registrado en el sistema.
Posibles errores	El usuario no existe, es incorrecto o la contraseña es introducida incorrectamente.
Flujo alternativo	Se muestra un mensaje con el error correspondiente con un botón de *return back* que devuelve a la página anterior.
Condiciones posteriores	Se redirige automáticamente al menú inicial donde se muestran las opciones principales.

Nombre del caso	Cerrar sesión
Descripción	El usuario cierra la sesión de la aplicación.
Flujo básico	El usuario cierra sesión correctamente.
Condiciones previas	El usuario tiene que haber iniciado sesión previamente.
Posibles errores	No cumple la condición previa.
Flujo alternativo	Se redirige automáticamente a la página de inicio de sesión.
Condiciones posteriores	Se redirige automáticamente a la página de inicio de sesión.

Nombre del caso	Registrar una imagen
Descripción	El usuario puede registrar una imagen en el sistema introduciendo un título, descripción, palabras clave, autor, fecha de creación y adjuntar un archivo perteneciente a la imagen del tipo .jpg, .png, .jpeg y .gif.
Flujo básico	El usuario introduce los campos necesarios, adjunta la imagen y la registra en el sistema.
Condiciones previas	El usuario tiene que haber iniciado sesión correctamente.
Posibles errores	No se han introducido todos los campos, no se ha adjuntado la imagen o la imagen introducida no corresponde a un formato aceptado.
Flujo alternativo	Se muestra un mensaje con el error correspondiente con un botón de *return back* que devuelve a la página anterior. En caso de no haber iniciado sesión, se redirige automáticamente a la página de inicio de sesión.
Condiciones posteriores	Se ha registrado la imagen correctamente y se redirige de nuevo al menú.

Nombre del caso	Listar imágenes
Descripción	Un usuario puede ver el listado de imágenes en la página relacionada.
Flujo básico	El usuario ve el listado de imágenes en la que se muestran los campos introducidos previamente, un campo en la que se indica si la imagen está cifrada en disco o no, la imagen en un tamaño reducido, y en caso de ser la persona que ha registrado la imagen, verá unos botones de acción para modificar, eliminar o encriptar la imagen.
Condiciones previas	El usuario tiene que haber iniciado sesión correctamente.
Posibles errores	En caso de no haber registrado ninguna imagen, simplemente no se mostrará nada en la lista.
Flujo alternativo	En caso de no haber iniciado sesión, se redirige automáticamente a la página de inicio de sesión.
Condiciones posteriores	Se muestran las imágenes registradas en el sistema.

Nombre del caso	Modificar una imagen
Descripción	El usuario puede modificar una imagen en el sistema cambiando el título, descripción, palabras clave, autor, fecha de creación y adjuntar un archivo perteneciente a la imagen del tipo .jpg, .png, .jpeg y .gif. No es necesario adjuntar un nuevo archivo de imagen.
Flujo básico	El usuario accede a la imagen que se puede eliminar desde el listado de imágenes y modifica los campos necesarios, adjunta la imagen si lo requiere y se modifica la imagen del sistema.
Condiciones previas	El usuario tiene que haber iniciado sesión, la imagen tiene que existir en el sistema y el usuario tiene que ser el propietario de la imagen.
Posibles errores	Si alguna de las condiciones previas no se cumple, los campos no están completamente rellenados o si la imagen adjuntada no corresponde a uno de los formatos admitidos.
Flujo alternativo	Se muestra un mensaje con el error correspondiente con un botón de *return back* que devuelve a la página anterior. En caso de no haber iniciado sesión, se redirige automáticamente a la página de inicio de sesión.
Condiciones posteriores	Se ha modificado la imagen correctamente y se redirige de nuevo al menú.

Nombre del caso	Eliminar una imagen
Descripción	El usuario puede eliminar una imagen del sistema.
Flujo básico	El usuario accede a la imagen que se puede eliminar desde el listado de imágenes o búsqueda de imágenes y la elimina si lo requiere.
Condiciones previas	El usuario tiene que haber iniciado sesión, la imagen tiene que existir en el sistema y el usuario tiene que ser el propietario de la imagen.
Posibles errores	Si alguna de las condiciones previas no se cumple.
Flujo alternativo	Se muestra un mensaje con el error correspondiente con un botón de *return back* que devuelve a la página anterior. En caso de no haber iniciado sesión, se redirige automáticamente a la página de inicio de sesión.
Condiciones posteriores	Se ha eliminado la imagen correctamente y se redirige de nuevo al menú.

Nombre del caso	Encriptar/Desencriptar una imagen
Descripción	El usuario puede encriptar una imagen del sistema.
Flujo básico	El usuario accede a la imagen que se puede encriptar/desencriptar desde el listado de imágenes o la búsqueda y la encripta/desencripta si lo requiere.
Condiciones previas	El usuario tiene que haber iniciado sesión, la imagen tiene que existir en el sistema y el usuario tiene que ser el propietario de la imagen.
Posibles errores	Si alguna de las condiciones previas no se cumple.
Flujo alternativo	Se muestra un mensaje con el error correspondiente con un botón de *return back* que devuelve a la página anterior. En caso de no haber iniciado sesión, se redirige automáticamente a la página de inicio de sesión.
Condiciones posteriores	Se ha encriptado/desencriptado la imagen correctamente y se redirige de nuevo al menú.

Nombre del caso	Buscar imágenes
Descripción	El usuario puede buscar una imagen del sistema por similitud a los campos introducidos. Sobre estos campos que se introducen en la búsqueda, se aplicará una búsqueda que se haga sobre todos ellos a la vez, es decir, que, si se introduce un título y una descripción, se buscan todas las imágenes que tengan un título y descripción similares a este.
Flujo básico	El usuario introduce los campos necesarios y se realiza la búsqueda sobre todas las imágenes del sistema.
Condiciones previas	El usuario tiene que haber iniciado sesión correctamente.
Posibles errores	En caso de no haber registrado ninguna imagen con los parámetros introducidos, simplemente no se mostrará nada en la lista de imágenes buscadas.
Flujo alternativo	En caso de no haber iniciado sesión, se redirige automáticamente a la página de inicio de sesión.
Condiciones posteriores	Se muestran las imágenes registradas en el sistema correspondientes a la búsqueda realizada.

Nombre del caso	Ver una imagen
Descripción	El usuario puede ver una imagen en tamaño grande.
Flujo básico	El usuario puede ver una imagen cuando se está en el listado de imágenes, cuando se ha realizado una búsqueda, cuando va a modificar, eliminar o cifrar/descifrar una imagen.
Condiciones previas	El usuario tiene que haber iniciado sesión y la imagen tiene que existir en el sistema.
Posibles errores	Si alguna de las condiciones previas no se cumple.
Flujo alternativo	Se muestra un mensaje con el error correspondiente con un botón de *return back* que devuelve a la página anterior. En caso de no haber iniciado sesión, se redirige automáticamente a la página de inicio de sesión.
Condiciones posteriores	Se ha mostrado en pantalla una página en la que se puede ver la imagen correctamente.

Nombre del caso	Registrar un vídeo
Descripción	El usuario puede registrar una imagen en el sistema introduciendo un título, descripción, palabras clave, autor, y enlace de YouTube.
Flujo básico	El usuario introduce los campos necesarios.
Condiciones previas	El usuario tiene que haber iniciado sesión correctamente.
Posibles errores	No se han introducido todos los campos.
Flujo alternativo	Se muestra un mensaje con el error correspondiente con un botón de *return back* que devuelve a la página anterior. En caso de no haber iniciado sesión, se redirige automáticamente a la página de inicio de sesión.
Condiciones posteriores	Se ha registrado el vídeo.

Nombre del caso	Listar vídeos
Descripción	Un usuario puede ver el listado de vídeos en la página relacionada.
Flujo básico	El usuario ve el listado de vídeos en el que se muestran los campos introducidos previamente, un campo para poder ver el vídeo en tamaño completo, y en caso de ser la persona que ha registrado el vídeo, verá unos botones de acción para modificar o eliminar.
Condiciones previas	El usuario tiene que haber iniciado sesión correctamente.
Posibles errores	En caso de no haber registrado ningún vídeo, simplemente no se mostrará nada en la lista.
Flujo alternativo	En caso de no haber iniciado sesión, se redirige automáticamente a la página de inicio de sesión.
Condiciones posteriores	Se muestran los vídeos registrados en el sistema.

Nombre del caso	Modificar un vídeo
Descripción	El usuario puede modificar un vídeo en el sistema cambiando el título, descripción, palabras clave, autor y enlace de YouTube. No es necesario adjuntar un nuevo archivo de imagen.
Flujo básico	El usuario accede al vídeo que se puede eliminar desde el listado de vídeos y modifica los campos necesarios y se modifica el vídeo del sistema.
Condiciones previas	El usuario tiene que haber iniciado sesión, el vídeo tiene que existir en el sistema y el usuario tiene que ser el propietario de la imagen.
Posibles errores	Si alguna de las condiciones previas no se cumple o los campos no están completamente rellenados.
Flujo alternativo	Se muestra un mensaje con el error correspondiente con un botón de *return back* que devuelve a la página anterior. En caso de no haber iniciado sesión, se redirige automáticamente a la página de inicio de sesión.
Condiciones posteriores	Se ha modificado el vídeo correctamente y se redirige de nuevo al menú.

Nombre del caso	Eliminar un vídeo
Descripción	El usuario puede eliminar un vídeo del sistema.
Flujo básico	El usuario accede al vídeo que se puede eliminar desde el listado de vídeos o búsqueda de vídeos y la elimina si lo requiere.
Condiciones previas	El usuario tiene que haber iniciado sesión el vídeo tiene que existir en el sistema y el usuario tiene que ser el propietario del vídeo.
Posibles errores	Si alguna de las condiciones previas no se cumple.
Flujo alternativo	Se muestra un mensaje con el error correspondiente con un botón de *return back* que devuelve a la página anterior. En caso de no haber iniciado sesión, se redirige automáticamente a la página de inicio de sesión.
Condiciones posteriores	Se ha eliminado el vídeo correctamente y se redirige de nuevo al menú.

Nombre del caso	Buscar vídeos
Descripción	El usuario puede buscar un vídeo del sistema por similitud a los campos introducidos. Sobre estos campos que se introducen en la búsqueda, se aplicará una búsqueda que se haga sobre todos ellos a la vez, es decir, que, si se introduce un título y una descripción, se buscan todos los vídeos que tengan un título y descripción similares a este.
Flujo básico	El usuario introduce los campos necesarios y se realiza la búsqueda sobre todos los vídeos del sistema.
Condiciones previas	El usuario tiene que haber iniciado sesión correctamente.
Posibles errores	En caso de no haber registrado ningún vídeo con los parámetros introducidos, simplemente no se mostrará nada en la lista de vídeos buscados.
Flujo alternativo	En caso de no haber iniciado sesión, se redirige automáticamente a la página de inicio de sesión.
Condiciones posteriores	Se muestran los vídeos registrados en el sistema correspondientes a la búsqueda realizada.

Nombre del caso	Ver un vídeo
Descripción	El usuario puede ver un vídeo en un reproductor a través de la aplicación.
Flujo básico	El usuario puede ver un vídeo cuando se está en el listado de vídeos, cuando se ha realizado una búsqueda y cuando va a modificar o eliminar un vídeo.
Condiciones previas	El usuario tiene que haber iniciado sesión y el vídeo tiene que existir en el sistema.
Posibles errores	Si alguna de las condiciones previas no se cumple.
Flujo alternativo	Se muestra un mensaje con el error correspondiente con un botón de *return back* que devuelve a la página anterior. En caso de no haber iniciado sesión, se redirige automáticamente a la página de inicio de sesión.
Condiciones posteriores	Se ha mostrado en pantalla una página en la que se puede ver el vídeo en el reproductor de la aplicación.

11. Justificación de los lenguajes escogidos

En este apartado se explicarán los lenguajes escogidos para desarrollar las aplicaciones y servicios web, explicando las características del framework, lenguaje y librerías necesarias para el correcto funcionamiento.

11.1 Cuáles son (Lenguaje + framework)

11.1.1 JavaScript y Express

Por qué este lenguaje

JavaScript es el lenguaje más usado actualmente para desarrollo de aplicaciones web en general. Existe cantidad infinita de frameworks para este lenguaje ya que es fácil de aprender y bastante querido por la comunidad al ofrecer librerías para todo tipo de proyectos. La principal razón para escoger y aprender este lenguaje, es que se puede aplicar hoy en día tanto para desarrollar aplicaciones web como para escritorio sin tener grandes cambios e incluso tanto para el lado del cliente como del servidor, es por ello que es una gran opción como lenguaje.

Características orientadas al desarrollo web

Para poder ejecutar JavaScript en el lado del servidor, es necesario disponer de Node.js. Hasta entonces, JavaScript solo podía ejecutarse en el lado del cliente en los navegadores, pero con la aparición de este entorno con gran rendimiento y capacidad de construir aplicaciones web escalables, JavaScript dio un salto más grande en cuanto a desarrolladores que lo usaban.

Entorno de desarrollo

Como en todos los lenguajes interpretados escogidos, no es necesario disponer de un entorno preparado para desarrollar ya que la ejecución del programa es sencilla y está organizada desde el propio código y no genera ningún archivo, por lo que se usará Visual Studio Code y un terminal desde el que ejecutar y debugar el código.

Infraestructura necesaria para el despliegue

Para el despliegue de las aplicaciones y la base de datos, se necesitará tener instalado docker en el servidor y en el ordenador donde se desarrolla. En caso de que se quiera desarrollar de manera local en el ordenador, sin usar docker, tendremos que tener instalado MySQL y los paquetes de Express. Para usarlo, se necesita usar *npm* para instalar todos los requisitos y para ejecutar solo será un comando simple que ejecutará el servidor.

Identificación de leyes y regulaciones

JavaScript no tiene una licencia específica propia. JavaScript es estandarizado por la organización Ecma International a través de su especificación ECMA-262. La versión más común y utilizada de JavaScript es ECMAScript 6 (ES6 o ES2015) y sus versiones posteriores.

La licencia que se aplica a JavaScript depende del entorno en el que se esté utilizando. Por ejemplo:

- Navegadores web: Los motores de JavaScript en los navegadores web, como V8 en Chrome o SpiderMonkey en Firefox, tienen sus propias licencias. Por lo general, estos navegadores son de código abierto, y sus motores JavaScript incorporados también lo son. En el caso de Chrome, por ejemplo, utiliza el proyecto Chromium, que tiene una licencia de código abierto (BSD) para el código fuente del navegador.
- Node.js: Node.js, que permite ejecutar JavaScript en el lado del servidor, utiliza la Licencia MIT, una licencia de código abierto muy permisiva.
- Express: Es un framework de código abierto y también con licencia MIT.

11.1.2 Python y Flask

Por qué este lenguaje

Python es conocido por la facilidad de entender la sintaxis del lenguaje y la facilidad de aprendizaje del lenguaje en general, es la razón por la que me ha llevado a escogerlo principalmente. Además, es el segundo lenguaje más usado para desarrollar, por detrás de JavaScript, por lo que es un lenguaje que muchos desarrolladores usan. Además, como los tres lenguajes interpretados escogidos, es muy sencillo de ejecutar en local y con un contenedor Docker por lo que facilitará el despliegue sobre estos.

Características orientadas al desarrollo web

Entre los frameworks de desarrollo web backend que existen para Python, destacan dos: Django y Flask. Ambos tienen características similares, pero hay una que las diferencia claramente, la simplicidad de Flask y la complejidad de Django. Al tratarse de una aplicación y servicio web que no necesita de grandes características. Como Express, no incluye ningún ORM (Object-Relational Mapping) por defecto, por lo que habrá que implementar las funciones de consulta de la base de datos desde cero. La característica más importante de Flask es que es muy ligero y compatible como servidor, por lo que será capaz de correr en cualquier tipo de ordenador/servidor. Como última característica, no tenemos que seguir una estructura predefinida de carpetas o archivos, así que podemos organizar el código como nos sea más fácil. Como motor de plantillas, se ha usado Jinja2, que es el más común y usado para Python y Flask.

Entorno de desarrollo

Como en todos los lenguajes interpretados escogidos, no es necesario disponer de un entorno preparado para desarrollar ya que la ejecución del programa es sencilla y está organizada desde el propio código y no genera ningún archivo, por lo que se usará Visual Studio Code y un terminal desde el que ejecutar y debugar el código.

Infraestructura necesaria para el despliegue

Para el despliegue de las aplicaciones y la base de datos, se necesitará tener instalado docker en el servidor y en el ordenador donde se desarrolla. En caso de que se quiera desarrollar de manera local en el ordenador, sin usar docker, tendremos que tener instalado MySQL y los paquetes de Flask. Para usar Flask, se necesita usar *Pip* para instalar todos los requisitos y para ejecutar solo será un comando simple que ejecutará el servidor.

Identificación de leyes y regulaciones

Python está distribuido bajo la Licencia Python (también conocida como Python Software Foundation License). Esta licencia es una licencia de código abierto que es compatible con la Open Source Initiative (OSI) y la Free Software Foundation (FSF).

En cuanto al framework de desarrollo web Flask, utiliza BSD, que es otra licencia de software libre y de código abierto. La Licencia BSD es bastante permisiva y flexible en cuanto a uso y redistribución.

11.1.3 PHP y Laravel

Por qué este lenguaje

PHP ha sido históricamente uno de los lenguajes que más se ha usado para desarrollar aplicaciones y servicios web, como, por ejemplo, usando WordPress o actualmente el framework más conocido, Laravel. La mayoría de las aplicaciones de hoy en día, siguen desarrolladas bajo este lenguaje, aunque muchos ya no quieran usarlo y renieguen de este, sigue ocupando una cuota de mercado del 79,2% [25], esto teniendo en cuenta que las aplicaciones tienen alguna dependencia de este lenguaje. Es por la razón de haber sido y seguir siendo tan popular a la hora de crear aplicaciones del lado del servidor por la que lo he escogido.

Características orientadas al desarrollo web

A diferencia de los otros lenguajes interpretados, Laravel crea un paquete en el que se incluye todo tipo de archivos y carpetas para poder iniciar el desarrollo sin tener que preocupar por la estructura del proyecto, incluso creando archivos de ejemplo sobre los que poder editar. Esto es de gran ayuda para alguien que nunca ha usado el framework, ya que no tiene que empezar completamente desde cero. Laravel implementa por defecto una arquitectura con patrón de Modelo Vista Controlador, además de implementar también el motor de plantillas Blade, el cual usaremos para representar los datos sobre HTML. Además, la conexión con la base de datos MySQL viene por defecto, solo tendremos que ajustar los parámetros del usuario, contraseña, contraseña de *root* y la URL del contenedor docker o local.

Entorno de desarrollo

Como en todos los lenguajes interpretados escogidos, no es necesario disponer de un entorno preparado para desarrollar ya que la ejecución del programa es sencilla y está organizada desde el propio código y no genera ningún archivo, por lo que se usará Visual Studio Code y un terminal desde el que ejecutar y debugar el código.

Infraestructura necesaria para el despliegue

Para el despliegue de las aplicaciones y la base de datos, se necesitará tener instalado docker en el servidor y en el ordenador donde se desarrolla. En caso de que se quiera desarrollar de manera local en el ordenador, sin usar docker, deberemos tener instalado MySQL, ya que los paquetes necesarios para la conexión ya vienen instalados al crear el proyecto y por tanto no hace falta instalarlos.

Identificación de leyes y regulaciones

PHP cuenta con una licencia de código abierto llamada PHP License. Es una licencia de estilo BSD que es compatible con la Open Source Initiative (OSI). Permite la libre redistribución y modificación del software, siempre que se cumplan ciertas condiciones.

En cuanto al framework de PHP Laravel, también es de código abierto y utiliza la Licencia MIT. La Licencia MIT es una licencia de software libre y de código abierto que permite la libre redistribución y modificación del software, siempre que se incluyan los avisos de copyright en todas las copias o partes sustanciales del software.

11.1.4 Java y Spring Boot

Por qué este lenguaje

He decidido escoger este lenguaje, principalmente, porque en el grado lo usamos en distintas asignaturas, sobre todo en la asignatura en la que se basa este proyecto (Aplicaciones Distribuidas). En esta desarrollamos una aplicación igual, pero con menos casos de uso con este lenguaje, así que servirá como base para poder tener unos conocimientos, aunque en el proyecto agregaremos el uso del framework Spring y Maven como herramienta de gestión del proyecto, por lo que no será exactamente igual. Además, será el único lenguaje compilado que se use para desarrollar y comparar las aplicaciones.

Características orientadas al desarrollo web

Spring es el framework más usado por los desarrolladores web que trabajan con Java, así como uno de los más preferidos en general por la comunidad. La principal característica de Spring Boot es que está pensado para simplificar el desarrollo y, sobre todo, el inicio y configuración de dependencias para no tener que realizar una configuración extensa. Además, existe la posibilidad de usar Spring Boot Starter, que crea un proyecto con los paquetes y dependencias que seleccionemos orientados a web, datos, seguridad, etc. Facilitando la creación de las aplicaciones con configuraciones específicas.

Como motor de plantillas, se ha usado Thymeleaf, ya que este se integra completamente con Java Spring y se puede añadir con Spring Boot desde que se crea el proyecto.

Entorno de desarrollo

El entorno de desarrollo Spring Tools Suite para este proyecto será uno propio creado por Spring en el que se incluye Spring y Spring Boot para poder crear, ejecutar y configurar todo lo necesario, incluyendo el propio servidor. Este servidor será Apache Tomcat, del que más tarde hablaremos. Existen varias configuraciones, como puede ser una extensión con todo lo necesario para usar en Visual Studio Code o un IDE propio basado en Eclipse. Este será el que usemos ya que es el que necesita de menos configuración.

Infraestructura necesaria para el despliegue

Spring tiene una configuración por la que, si la añadimos al crear el proyecto, automáticamente creará un archivo *docker-compose.yml* para usar la base de datos directamente con Docker y así no tener que tenerla instalada en local. Esto hará que el despliegue de la aplicación sea algo más lenta al tener que ejecutar el contenedor, pero nos será más sencillo de cara a cuando tengamos que añadir la propia aplicación como Docker ya que ya estarán enlazados desde un principio.

En cuanto al servidor en el que se ejecutará nuestra aplicación mientras la desarrollamos, ya viene en el IDE por defecto y es Apache Tomcat. El IDE se encarga de ejecutar la aplicación y de

poner los archivos compilados en el servidor automáticamente, por lo que no tendremos que preocuparnos de nada.

Una vez esté desarrollada la aplicación, sí que deberemos tener más cuidado. Al estar usando Maven, tendremos que crear un archivo ".war", que es donde estará todo lo necesario para desplegarlo sobre un contenedor Docker. Este es un archivo parecido a un archivo compresor pero que contiene todos los archivos para ejecutar las funciones y mostrar las vistas. En este caso, el contenedor Docker será el mismo servidor que usa en local, Apache Tomcat, solo que tendremos que copiar este archivo ".war" en la carpeta que se indica en las instrucciones de despliegue del contenedor para que, al ejecutarse, se despliegue todo de forma automática.

Identificación de leyes y regulaciones

El caso de Java es algo más característico. La Máquina Virtual de Java (JVM) y la Biblioteca de Clases de Java (Java Standard Edition - JSE), son distribuidos por Oracle bajo la Oracle Binary Code License (Oracle BCL). Pero Java también está disponible bajo la Licencia Pública General de GNU (GNU General Public License - GPL) en su implementación OpenJDK, que es una implementación de código abierto de la plataforma Java Standard Edition.

En cuanto a Spring, utiliza principalmente la Licencia Apache 2.0. La Licencia Apache 2.0 es una licencia de código abierto que permite la redistribución y modificación del software bajo algunas condiciones. Es compatible con la Open Source Initiative (OSI).

12. Comparativa entre las aplicaciones desarrolladas

Una vez desarrolladas las dos aplicaciones para cada lenguaje, en este apartado del proyecto se van a realizar distintos análisis y comparaciones sobre los resultados obtenidos.

12.1 Similitud con lenguajes aprendidos durante el grado

Al empezar a cursar el grado de ingeniería informática, existe la asignatura de Programación 1 y programación 2. En estas se enseñan las bases y la lógica de la programación básica y programación orientada a objetos en el lenguaje de programación C++, así que tomaremos este como lenguaje para comparar con los usados en este proyecto.

C++ es una extensión del lenguaje de programación compilado C, en la que se añaden características de programación orientado a objetos. Además de esto, puede actuar como lenguaje orientado a programación orientada a eventos, programación funcional y programación general. Esta última se introdujo junto al uso de plantillas (templates) que permiten escribir algoritmos y estructuras de datos sin perder eficiencia. También hay que tener en cuenta el tipado de los datos, que es fuerte y estático.

A continuación, se compararán las características de los lenguajes usados para el desarrollo con C++.

12.1.1 Java

En cuanto a la sintaxis del lenguaje, es sin duda el lenguaje más similar ya que C/C++ fueron una gran influencia para crear Java. Sin tener conocimientos sobre Java es fácil entender su sintaxis si la tienes sobre C++ y no tiene gran complicación adaptarse entre ellos. Otra gran similitud entre ellos es que ambos son orientados a objetos y con un tipado de datos estático, además que ambos incluyen tipos de datos primitivos (enteros, caracteres, booleanos, etc.). Es verdad que ambos lenguajes son compilados, pero existe una diferencia importante entre ellos. C++ es un lenguaje compilado directamente a lenguaje máquina y se ejecuta sobre la CPU, en Java es necesario tener la máquina virtual de Java (JVM) para poder ejecutar el código compilado, en nuestro caso, el servidor Apache Tomcat será quien disponga de la JVM. Otra diferencia es como se utilizan las excepciones de la ejecución, ya que Java es muy estricto en este sentido, mientras que C++ es más laxo.

12.1.2 JavaScript

La mayor diferencia encontrada es que es un lenguaje interpretado, pero esto no es una característica importante que sea diferencial para comparar entre C++ y JavaScript. La parte más diferente de cómo funciona C++ es el retorno de las funciones, ya que en JavaScript no funciona de la misma manera. En este caso había dos posibles opciones, el uso de *callbacks* o *promises* para hacer uso de la asincronía. Las operaciones asíncronas son aquellas que no bloquean la ejecución del código, permitiendo que otras tareas continúen mientras se espera que la operación asíncrona se complete. Por tanto, una promesa representa un valor que puede no estar disponible de inmediato pero que en algún punto lo estará. Decidí hacer uso de la primera opción al tener un conocimiento previo sobre estos y al semejarse más al funcionamiento de C++ y de Java. Aunque JavaScript también tenga una sintaxis basada en la de C/C++, difiere algo más de esta al tratarse de un lenguaje con tipados dinámico y débil al no tener que declarar un número entero de forma explícita como se hace en C++ y el poder cambiar el tipo de variable durante la ejecución del código, pero aun así la sintaxis de bucles y condicionales es casi igual, así como la creación de funciones para simplificar el código. JavaScript admite también orientación a objetos, aunque no es la principal razón de este lenguaje y algo que se haya usado como característica en el desarrollo de la aplicación y el servicio web.

12.1.3 Python

Python también está basado en C/C++, por lo que comparten una parte de la sintaxis, como controles de flujo, y los operadores. Ambos son lenguajes que admiten funciones y orientados a objetos, aunque esta última característica no la he usado para nada con Python y el desarrollo de las funcionalidades de la aplicación y el servicio web. Sí que existe una diferencia a la hora de declarar las variables, que, si se asigna directamente un valor, no hace falta declarar de que tipo es ya que lo obtiene de manera automática al asignarle un valor o carácter, pero sigue siendo más similar que en PHP. En cuanto al uso de funciones, hace el mismo uso que C++, no como lo explicado anteriormente con JavaScript, aunque también existe una opción de usar *callbacks* con Python.

12.1.4 PHP

PHP es el lenguaje que más se diferencia a C++ en todos los sentidos. No tiene una sintaxis similar en cuanto al uso de variables, ya que en PHP se hace uso del símbolo *$* para remarcar que es una variable. Sí que es verdad que en ambos se hace uso de la programación orientada a objetos y de programación con funciones y esto hace que se asimilen, pero el hecho diferenciador entre este lenguaje y los demás es claramente el uso las variables. PHP es también un lenguaje interpretado y de tipado dinámico, como los otros lenguajes interpretados usados, aunque no es una característica que haya sido necesario usar y que sea diferencial para escoger uno de estos lenguajes con respecto a Java, que no lo admite. A pesar de ser el más diferente entre todos, ha sido sin duda el lenguaje más fácil de aprender, en cuanto a sintaxis y uso del propio lenguaje, junto con su framework Laravel.

12.2 Eficiencia de desarrollo y problemas encontrados

La eficiencia del desarrollo de las aplicaciones y servicios web es algo bastante complicado de medir, ya que es algo subjetivo, pero también va relacionado con cuanto de fácil o difícil es desarrollar en un lenguaje.

Para medir cuanto de eficiente ha sido desarrollar en cada lenguaje, tomaremos el tiempo estimado para cada uno de ellos y el tiempo real medido en horas. Este tiempo es lo más real posible, pero teniendo en cuenta que siempre se pueden cometer errores de precisión de tiempo o que el tiempo pueda ser demasiado alto debido a la solución de problemas y bugs surgidos durante el desarrollo. En la tabla 11 se puede ver el tiempo estimado en un inicio sobre el tiempo que se tardaría en desarrollar todas y cada una de las aplicaciones y servicios web. En cada uno de los lenguajes existen cuatro tareas concretas:

1. **Instalación de requisitos**: Comprende la instalación de librerías necesarias, entorno de desarrollo, servidor local, creación de los archivos para desplegar sobre los contenedores Docker (Dockerfile y docker-compose.yml) y creación de un pequeño "Hello World" para comprobar que todo funciona correctamente, tanto para la aplicación como para el servicio web.
2. **Desarrollo de objetivos**: Incluye la realización de todos los casos de uso para aplicación y servicio web.
3. **Evaluación del desarrollo**: Engloba el testing del funcionamiento de los casos de uso desarrollados, así como el despliegue sobre Docker de las aplicaciones.
4. **Corrección de errores y mejoras**: Comprende el proceso de solución de todos los bugs encontrados durante el desarrollo.

Desarrollo del Proyecto		360 horas
DP1	Python y Flask	90 horas
DP1.1	Instalación de requisitos	5 horas
DP1.2	Desarrollo de objetivos	60 horas
DP1.3	Evaluación del desarrollo	10 horas
DP1.4	Corrección de errores y mejoras	15 horas
DP2	JavaScript y Express	90 horas
DP2.1	Instalación de requisitos	5 horas
DP2.2	Desarrollo de objetivos	60 horas
DP2.3	Evaluación del desarrollo	10 horas
DP2.4	Corrección de errores y mejoras	15 horas
DP3	PHP y Laravel	90 horas
DP3.1	Instalación de requisitos	5 horas
DP3.2	Desarrollo de objetivos	60 horas
DP3.3	Evaluación del desarrollo	10 horas
DP3.4	Corrección de errores y mejoras	15 horas
DP4	Java y Spring	90 horas
DP4.1	Instalación de requisitos	5 horas
DP4.2	Desarrollo de objetivos	60 horas
DP4.3	Evaluación del desarrollo	10 horas
DP4.4	Corrección de errores y mejoras	15 horas

TABLA 11: TABLA DE ESTIMACIONES DE LAS TAREAS DE DESARROLLO DE LAS APLICACIONES

FUENTE: ELABORACIÓN PROPIA.

Durante el periodo que comprende el desarrollo de las 4 aplicaciones y 4 servicios web, he ido apuntando el tiempo que dedicaba a cada una de las cuatro tareas por cada lenguaje, con la que se ha obtenido una tabla igual que la de las estimaciones de tiempo (Tabla 10) para poder comparar realmente el tiempo que se dedica a cada proyecto. Hay que tener en cuenta la integración de conocimiento explicados en el apartado 8 de este proyecto, ya que no se parte totalmente de cero, sino que hay ciertos conocimientos básicos previos adquiridos.

En este apartado nos centraremos en los puntos de instalación de requisitos, desarrollo de objetivos y corrección de errores y mejoras, dejando la evaluación del desarrollo para el siguiente apartado debido a que incluye el despliegue de infraestructura y el testing. Este último es el mismo para todos ya que cuentan con las mismas funcionalidades.

- **Instalación de requisitos**:
    - Sin duda, Laravel fue el framework más fácil de instalar con su gestor de paquetes "composer" ya que este tiene una opción de crear un proyecto con toda la estructura creada, con los paquetes básicos, conexión con la base de datos MySQL por defecto, etc. Todo esto hizo que el desarrollo pudiera comenzar de una manera más sencilla y rápida sin tener preocupación alguna sobre la jerarquía de carpetas del proyecto, así como los distintos archivos de configuración ni nada. Además, venía con una aplicación de "Hello World" creada y en la que solo había que ejecutar el servidor mediante un comando para poder probarla.
    - En segundo lugar, se encuentra Spring y Java. Al descargar un entorno de desarrollo personalizado en el que se puede crear un proyecto desde cero e incluyendo dependencias directamente desde una interfaz gráfica, es más sencillo de usar que por ejemplo Python y JavaScript. El mayor de los problemas que tuve fue configurar el servidor Apache Tomcat como contenedor de Docker para hacer el despliegue final, ya que la carpeta hay que introducirla con un nombre muy específico o se desplegará de manera errónea. Otros problemas fueron configurar la conexión con la base de datos ya que por defecto usa SSL y esto es innecesario para este proyecto y configurar y crear correctamente los archivos que usa Spring para realizar las operaciones con la base de datos a través de los modelos. Todo esto hizo que el tiempo aumentara, aunque fue menor que lo previsto inicialmente.
    - En tercer lugar, se encuentra Python y Flask. El único problema encontrado fue crear y configurar los directorios y archivos de configuración ya que el gestor de paquetes no crea la estructura ni lo mínimo necesario, ya que es un proyecto desde cero, pero existe información sobre cómo hacerlo y es una estructura sencilla. Hay que recordar que la utilidad de Flask es la ligereza del framework para no ocupar demasiado espacio y tener también un buen rendimiento.
    - Por último, se encuentra JavaScript y Express. En esto caso sí que existe un gestor de paquetes (NPM) y que es capaz de iniciar un proyecto para Express, pero configura solo las carpetas más básicas, pero para un proyecto de JavaScript general, por tanto, hay que seguir adaptando la estructura y crear los archivos. Además de esto, hubo algunos problemas con las librerías ya que costaba encontrar los nombre desde las que se instalaban con "npm".

- **Desarrollo de objetivos**:
    - De nuevo PHP y Laravel tienen un tiempo de desarrollo menor. Esto se debe principalmente al uso del ORM integrado *Eloquent*. Este permite no tener que crear y ejecutar las queries de SQL para insertar, modificar y eliminar de la base de datos. Este ORM permite crear, actualizar y eliminar usuarios, imágenes y vídeos ejecutando un solo comando por el que hace persistir el objeto en la base de datos con los cambios. También permite realizar consultas a través del parámetro *key* de la base de datos de manera sencilla. Las consultas más complejas, como puede ser la búsqueda combinada, hace falta crear una query sobre el ORM de forma manual. Con el uso del ORM queda un código más limpio y sencillo de comprender. La creación de los modelos es sencilla y además existe una serie de archivos que son las *migrations* las cuales permiten crear y actualizar las tablas de manera automática, esta funcionalidad se aprovecha al desplegar la aplicación sobre Docker para crear la base de datos correctamente.
    - En segundo lugar, está Java y Spring. No ha habido una complicación concreta para el desarrollo de los objetivos de la aplicación y el servicio web. Lo único a destacar es la tediosa configuración de las clases y paquetes para intentar organizar el código y que no estén los archivos juntos. También la configuración para poder usar el ORM Hibernate es algo difícil de hacer la primera vez debido al uso de una clase servicio para poder controlar los repositorios de los modelos, ambos archivos son los que manejarán las consultar a la base de datos, fue algo que costó entender al no tener una experiencia previa.
    - A continuación, se encuentra Python y Flask. Con este framework partíamos desde cero prácticamente. Por una parte, para que el código quede lo más limpio posible y ante la falta de un ORM configurado por defecto, decidí que las funciones que realizan las consultas a la base de datos estuvieran en un archivo aparte. El problema más importante, aunque no demasiado significativo, fue la configuración del motor de plantillas ya que no encontraba la carpeta donde estos estaban guardados, a pesar de estar creados de la manera que la documentación mostraba. El error se solucionó instalando una versión más reciente de la librería que los implementa.
    - Finalmente se encuentra de nuevo JavaScript. El desarrollo de las aplicaciones ha sido más complicado debido al uso de *callbacks* y entender cómo funcionan correctamente. El uso de estos provoca que el control de los errores cree demasiadas funciones anidadas que hacen que el código sea más complicado de entender y seguir a la hora de revisar errores. Además, también de tener que crear un archivo a parte que cuenta con las funciones de las llamadas a la base de datos para persistir los datos para intentar que el código quede lo más limpio posible. Esto ha sido lo único que ha retrasado el desarrollo más allá de las horas previstas y que haga que sea el lenguaje que más he tardado en desarrollar las aplicaciones, pero el coste de tiempo ha sido bastante alto.

- **Corrección de errores y mejoras**: Como he comentado anteriormente, el único lenguaje que ha dado problemas a la hora de solucionar bugs y errores concretos ha sido JavaScript, debido a la complejidad de las funciones anidadas y el uso de *callbacks*. El resto del tiempo para todos los lenguajes ha sido solución de errores comunes surgidos durante el propio desarrollo.

Es por lo motivos explicados anteriormente por los que el lenguaje más eficiente ha sido PHP junto al framework Laravel.

Desarrollo del Proyecto		342,8
DP1	Python y Flask	84,7
DP1.1	Instalación de requisitos	3,6
DP1.2	Desarrollo de objetivos	61,2
DP1.3	Evaluación del desarrollo	7,8
DP1.4	Corrección de errores y mejoras	12,1
DP2	JavaScript y Express	95,1
DP2.1	Instalación de requisitos	4,5
DP2.2	Desarrollo de objetivos	66,1
DP2.3	Evaluación del desarrollo	8,9
DP2.4	Corrección de errores y mejoras	15,6
DP3	PHP y Laravel	75,3
DP3.1	Instalación de requisitos	1,2
DP3.2	Desarrollo de objetivos	52,3
DP3.3	Evaluación del desarrollo	10,2
DP3.4	Corrección de errores y mejoras	11,6
DP4	Java y Spring	87,7
DP4.1	Instalación de requisitos	3,2
DP4.2	Desarrollo de objetivos	59,1
DP4.3	Evaluación del desarrollo	11,1
DP4.4	Corrección de errores y mejoras	14,3

TABLA 12: TABLA FINAL DE DEDICACIÓN POR APLICACIÓN Y TAREA

FUENTE: ELABORACIÓN PROPIA.

12.3 Facilidad de despliegue

En este apartado explicaré brevemente el proceso de despliegue de las aplicaciones, tanto de forma local para facilitar el desarrollo como sobre contenedores Docker para un despliegue en un servidor.

12.3.1 Local

En este contexto local, asumiremos que la base de datos se ejecuta también de forma local (*localhost)* y no en un contenedor. Para ello es necesario, tanto en Windows como Manjaro (Linux), ejecutar el servicio de MySQL en el puerto 3306.

En Java, al tener el servidor incorporado en el entorno de desarrollo *Spring Tool Suite*, no es necesario descargar ni instalar nada más para poder ejecutar las aplicaciones correctamente. Viene con una configuración por defecto por el que se ejecuta en el puerto 8080, el que usaremos para todas las aplicaciones web sin HTTPS. Para ejecutarlo, únicamente hace falta ejecutar la aplicación como *Spring Boot App* y el IDE se encarga de todo lo demás.

Para los tres lenguajes interpretados, el arranque de cada una de las aplicaciones es igual, salvando las diferencias de que cada uno usa un comando distinto, y el servidor arrancará correctamente y podremos acceder a la aplicación.

12.3.2 Docker

Con base de datos en Docker, si anteriormente hemos usado la base de datos en local, es necesario modificar la URL de conexión de la base de datos en todas las aplicaciones. Para facilitar la comprensión, asumiremos que la base de datos local no se está ejecutando, para evitar conflicto de puertos, y que el contenedor de MySQL se llama *db*. Al crearse la red entre los contenedores, Docker resuelve automáticamente el nombre como si se tratara de un registro DNS (nombre que apunta a una IP), por tanto, en todos los archivos de configuración de los distintos lenguajes, es necesario modificar la dirección *localhost* por *db* para que las aplicaciones y servicios web encuentren la base de datos.

En caso de ser el despliegue de la aplicación web, como comentamos en el apartado del contexto de las aplicaciones, necesitaremos crear un Dockerfile para crear la imagen de la aplicación y el posterior docker-compose.yml para ejecutarlos. La principal razón por la que usar Docker es por el aislamiento y la portabilidad que ofrecen, por ello se usa en entornos de producción real ya que proporciona algo más de seguridad y que no haya errores debido a incompatibilidades de sistemas operativos o plataformas. Es común entre los desarrolladores y las personas de infraestructuras escuchar la frase "*en mi ordenador funciona correctamente, pero en el servidor no"*, esto lo soluciona Docker. En este caso cualquiera de los lenguajes interpretados, menos PHP por tener que utilizar un servidor (Apache) distinto al local (propio de Laravel) y la configuración que conlleva, han sido más fácil de desplegar que Java.

12.4 Rendimiento

Para realizar un análisis de rendimiento de las aplicaciones, se hará un estudio sobre las cuatro aplicaciones y los cuatro servicios web. Para ello se cogerán un total de 20 imágenes y se utilizarán sobre el caso de uso de registrar imagen y listar imágenes. Se irá midiendo el tiempo que tarda en ejecutarse la función cada vez que se sube una imagen y listar el total de imágenes en cada momento. Para que el estudio sea lo más preciso posible, se escogerán de manera aleatoria las 20 imágenes de un repositorio de 1.000 imágenes. Cada imagen tendrá un número asignado y se generarán 20 números sobre los que escogerlas. También se realizará todas las mediciones en la misma máquina para evitar que el factor de rendimiento interfiera en los resultados. En el Anexo A se puede consultar las características de las imágenes escogidas.

12.4.1 Registrar Imagen

Una vez realizado el proceso de medir el tiempo de las 20 imágenes en las cuatro aplicaciones y servicios web, he calculado la media del tiempo tardado por las aplicaciones en milisegundos junto con la desviación estándar para calcular la variación entre los datos y la media calculada.

El primer lenguaje es JavaScript. En cuanto a la aplicación web, como podemos ver en la siguiente tabla 13, JavaScript y Express tiene unos tiempos de ejecución de la función bastante bajos, con una media de 8,55 milisegundos. Tiene una desviación no demasiado alta, aunque hay algunos valores altos que difieren de la media.

Viendo los datos del servicio web, sigue teniendo un tiempo bastante razonable en comparación con los siguientes lenguajes de programación. Tiene una media de 18,5 milisegundos y una desviación de 9,67. Esa es superior a la de la aplicación ya que el tiempo medio es más superior y tiene más picos entre los distintos valores medidos.

Como se puede ver, existe una diferencia clara entre la aplicación y el servicio web, esto es normal debido a que hay un intermediario al que hacerle la petición con los datos y esto retrasa la subida de la imagen.

Aplicación Web

Imagen	Tiempo	Media	Desviación
1	11	8,55	4,47
2	7		
3	5		
4	8		
5	6		
6	7		
7	22		
8	8		
9	5		
10	4		
11	15		
12	13		
13	4		
14	5		
15	10		
16	5		
17	5		
18	11		
19	11		
20	9		

Servicio Web

Imagen	Tiempo	Media	Desviación
1	34	18,5	9,67
2	14		
3	13		
4	24		
5	20		
6	18		
7	20		
8	15		
9	28		
10	38		
11	13		
12	10		
13	10		
14	12		
15	41		
16	16		
17	12		
18	8		
19	13		
20	11		

TABLA 13: TABLAS DE CÁLCULOS DE EJECUCIÓN DE REGISTRAR IMAGEN EN JAVASCRIPT

FUENTE: ELABORACIÓN PROPIA.

El siguiente lenguaje es Python con el marco de desarrollo Flask. La aplicación web cuenta con un tiempo medio de ejecución de 16,5 milisegundos y una desviación de 13,16. Analizando la tabla 14, destacan algunos picos. Algunos de estos coinciden entre todas las tablas, por lo que, si miramos los archivos, nos daremos cuenta de que son más pesados por lo que es normal que existan estos.

En el uso del servicio web, el tiempo aumenta de nuevo a casi el doble, como en el anterior caso, con una media de 25,9 milésimas de segundo y una desviación de 31,57. La desviación pasa a ser mayor incluso que la media, por lo que existen varios valores demasiado altos. Esto indica que no tiene un tiempo constante a la hora de realizar las operaciones de subida de la imagen.

Aplicación Web			
Imagen	Tiempo	Media	Desviación
1	17	16,5	13,16
2	11		
3	11		
4	12		
5	19		
6	26		
7	13		
8	20		
9	26		
10	11		
11	10		
12	10		
13	9		
14	11		
15	68		
16	12		
17	13		
18	10		
19	10		
20	11		

Servicio Web			
Imagen	Tiempo	Media	Desviación
1	17	25,9	31,57
2	11		
3	14		
4	24		
5	28		
6	29		
7	13		
8	42		
9	46		
10	19		
11	20		
12	11		
13	10		
14	12		
15	153		
16	15		
17	14		
18	14		
19	11		
20	15		

TABLA 14: TABLAS DE CÁLCULOS DE EJECUCIÓN DE REGISTRAR IMAGEN EN PYTHON

FUENTE: ELABORACIÓN PROPIA.

PHP, el último de los tres lenguajes interpretados, parece ser el peor de ellos. La aplicación web tiene un tiempo medio de 20 milisegundos con una desviación de 5,96. Es más lento de media incluso que JavaScript usando el servicio web, por lo que no parece ser bueno en cuanto a rendimiento.

Si pasamos a ver el servicio web, destaca por ser el peor con diferencia de los tres interpretados y el peor de los cuatro lenguajes al obtener una media de 439,1 milisegundos y una desviación exageradamente alta, ya que existen bastantes valores muy altos, incluso superando el segundo de ejecución, por lo que no resulta eficiente de usar. Existe una diferencia muy exagerada entre el uso o no del servicio web para realizar la operación de registro de imagen.

Aplicación Web

Imagen	Tiempo
1	18
2	22
3	21
4	22
5	24
6	20
7	19
8	18
9	19
10	20
11	20
12	22
13	22
14	20
15	35
16	19
17	21
18	0
19	20
20	18

Media	Desviación
20	5,96

Servicio Web

Imagen	Tiempo
1	126
2	133
3	121
4	1159
5	1158
6	1158
7	130
8	1160
9	1158
10	146
11	118
12	117
13	128
14	127
15	1217
16	135
17	120
18	126
19	125
20	120

Media	Desviación
439,1	489,99

TABLA 15: TABLAS DE CÁLCULOS DE EJECUCIÓN DE REGISTRAR IMAGEN EN PHP

FUENTE: ELABORACIÓN PROPIA.

Java es un lenguaje compilado, por lo que la teoría dice que debería de ser el más rápido de los cuatro lenguajes. En la tabla de las aplicaciones web vemos como el tiempo medio es de 22,85 milisegundos y una desviación de 4,03, por lo que no es el lenguaje más rápido de todos los que se han estudiado, siendo incluso el tercero.

En cuanto al servicio web si consultamos la tabla 16, podemos ver como el tiempo es prácticamente el doble que el de la aplicación web. Más o menos sigue los mismos patrones que los demás servicios web, tardando más en aquellas imágenes que son más pesadas.

Aplicación Web

Imagen	Tiempo
1	19
2	22
3	25
4	21
5	25
6	26
7	22
8	33
9	23
10	19
11	24
12	20
13	22
14	19
15	31
16	20
17	21
18	18
19	27
20	20

Media	Desviación
22,85	4,03

Servicio Web

Imagen	Tiempo
1	42
2	46
3	42
4	55
5	50
6	51
7	32
8	52
9	66
10	51
11	55
12	36
13	41
14	49
15	136
16	44
17	35
18	34
19	51
20	49

Media	Desviación
50,85	21,71

TABLA 16: TABLAS DE CÁLCULOS DE EJECUCIÓN DE REGISTRAR IMAGEN EN JAVA

FUENTE: ELABORACIÓN PROPIA.

12.4.2 Listar Imágenes

Cada vez que se registraba una imagen y se anotaba el tiempo de ejecución de la función, después a listar imágenes para medir el tiempo y analizar el tiempo que tardaba en mostrarlas en la tabla.

Referenciando a la tabla 17, en JavaScript y usando la aplicación web, volvemos a ver cómo es la más rápida de los lenguajes con un tiempo medio de 5,45 y una desviación de 1,64 milisegundos. Que sea tan baja la desviación nos hace indicar que no es importante el número de imágenes que tiene que mostrar respecto al rendimiento general de la función.

En cuanto al servicio web, nos volvemos a encontrar con que es casi el doble de lento que la aplicación web, aun así con un tiempo bastante razonable de 9,7 milésimas de media y una desviación de 2,83 milisegundos.

Aplicación Web

Imagen	Tiempo		Media	Desviación
1	4		5,45	1,64
2	5			
3	5			
4	3			
5	5			
6	3			
7	6			
8	3			
9	6			
10	7			
11	6			
12	5			
13	6			
14	5			
15	8			
16	7			
17	5			
18	4			
19	7			
20	9			

Servicio Web

Imagen	Tiempo		Media	Desviación
1	11		9,7	2,83
2	11			
3	13			
4	8			
5	8			
6	10			
7	9			
8	19			
9	10			
10	10			
11	8			
12	9			
13	9			
14	9			
15	10			
16	7			
17	6			
18	9			
19	12			
20	6			

TABLA 17: TABLAS DE CÁLCULOS DE EJECUCIÓN DE LISTAR IMÁGENES EN JAVASCRIPT

FUENTE: ELABORACIÓN PROPIA.

El siguiente lenguaje es Python. Consultando la tabla 18, en la aplicación web, obtiene una media de 6,2 milisegundos y una desviación de 1,44. Es un poco más lenta que JavaScript, pero más rápida que PHP. También tiene una desviación muy baja, lo que hace indicar que tampoco influye la cantidad de imágenes en el tiempo de carga de la lista.

El dato sorprendente viene del uso del servicio web, el cual obtiene un rendimiento muy similar al de la aplicación web, con una media de 7,2 y una desviación de 2,73 milésimas de segundo. Es el servicio web que mejor rendimiento ha obtenido.

Aplicación Web

Imagen	Tiempo		Media	Desviación
1	6		6,2	1,44
2	6			
3	8			
4	7			
5	5			
6	6			
7	6			
8	6			
9	5			
10	4			
11	6			
12	6			
13	7			
14	9			
15	10			
16	6			
17	5			
18	5			
19	6			
20	5			

Servicio Web

Imagen	Tiempo		Media	Desviación
1	8		7,2	2,73
2	5			
3	6			
4	6			
5	6			
6	9			
7	18			
8	7			
9	6			
10	7			
11	7			
12	5			
13	7			
14	7			
15	7			
16	8			
17	7			
18	6			
19	6			
20	6			

TABLA 18: TABLAS DE CÁLCULOS DE EJECUCIÓN DE LISTAR IMÁGENES EN PYTHON

FUENTE: ELABORACIÓN PROPIA

PHP vuelve a ser el lenguaje más lento. En la tabla 19, se muestra el tiempo con una media de 11,7 milésimas y 1, 72 de desviación la diferencia es del doble o casi con respecto a los demás, nos hace indicar que no es la aplicación con mejor rendimiento si no todo lo contrario.

En cuanto al servicio web, este crece de manera exagerada de nuevo, con una media de 108,65 milisegundos y una desviación de 4,26. La desviación sigue siendo muy baja por lo que mantiene los tiempos de ejecución con más o menos imágenes en la base de datos y guardados en disco.

Aplicación Web				
Imagen	Tiempo		Media	Desviación
1	12		11,7	1,72
2	11			
3	10			
4	11			
5	12			
6	10			
7	16			
8	14			
9	15			
10	11			
11	12			
12	11			
13	10			
14	11			
15	10			
16	13			
17	10			
18	11			
19	11			
20	13			

Servicio Web				
Imagen	Tiempo		Media	Desviación
1	106		108,65	4,26
2	107			
3	105			
4	104			
5	119			
6	113			
7	103			
8	103			
9	108			
10	108			
11	113			
12	109			
13	111			
14	108			
15	109			
16	104			
17	107			
18	109			
19	116			
20	111			

TABLA 19: TABLAS DE CÁLCULOS DE EJECUCIÓN DE LISTAR IMÁGENES EN PHP

FUENTE: ELABORACIÓN PROPIA.

En cuanto a las aplicaciones web, Java se encuentra en tercer lugar. En la tabla 20, se puede ver que con una media de listado de imágenes de 8,5 y una desviación de 1,57 milésimas de segundo no tiene uno de los mejores tiempos como se esperaba en un inicio.

El servicio web vuelve de nuevo a duplicar el tiempo de ejecución de la función con un tiempo medio de 16 milésimas de segundo y una desviación 1,95. Esto nos hace indicar que la implementación del servicio web hace perder rendimiento a las aplicaciones, es una de las desventajas principales del uso de esta arquitectura.

Aplicación Web			
Imagen	Tiempo	Media	Desviación
1	7	8,50	1,57
2	7		
3	8		
4	9		
5	11		
6	11		
7	6		
8	9		
9	8		
10	9		
11	7		
12	8		
13	7		
14	12		
15	9		
16	7		
17	10		
18	8		
19	8		
20	9		

Servicio Web			
Imagen	Tiempo	Media	Desviación
1	19	16,00	1,95
2	17		
3	15		
4	14		
5	15		
6	18		
7	17		
8	18		
9	14		
10	17		
11	15		
12	19		
13	14		
14	15		
15	17		
16	14		
17	18		
18	15		
19	12		
20	17		

TABLA 20: TABLAS DE CÁLCULOS DE EJECUCIÓN DE LISTAR IMÁGENES EN JAVA

FUENTE: ELABORACIÓN PROPIA.

12.4.3 Sobrecarga de la base de datos

En este apartado se va a valorar si sobrecargando la base de datos, influye en el rendimiento del lenguaje. Para ello, se llenará la base de datos con 2.000 imágenes, sin registrarlas de forma manual y no tener archivos, creando un bucle que haga estas inserciones. La comparación se ha hecho tanto en la aplicación como en el servicio web. Para introducir esta cantidad de datos, se ha creado una función en cada aplicación que realiza 2.000 conexiones introduciendo las imágenes de forma automática. A continuación veremos los resultados.

En la siguiente tabla se pueden ver los resultados obtenidos. Vemos con JavaScript vuelve a ser el más rápido de los lenguajes. Para ser tantos elementos, no son tiempos demasiado altos. Python es el lenguaje más lento en ejecutar el listado de imágenes, tardando incluso cinco veces más que el mejor de los cuatro.

Aplicación Web			
JavaScript	Python	PHP	Java
21	109	56	45

TABLA 21: TABLAS DE CÁLCULOS DE EJECUCIÓN DE LISTAR IMÁGENES CON 2.000 ELEMENTOS CON APLICACIÓN WEB

FUENTE: ELABORACIÓN PROPIA.

En la tabla 22 se puede ver los resultados obtenidos usando el servicio web, vemos como el tiempo en milisegundos ha aumentado en todos los lenguajes con un incremento de casi el doble como mínimo en todos. El caso más destacable es el de Python, el tiempo el cual se ha multiplicado por 20. Con el lenguaje Python, a la hora de introducir los 2.000 elementos en la base de datos, no permitía realizar el bucle completo, si no que no aceptaba tantas inserciones seguidas. Para ello tuve que modificar el bucle hasta un máximo de 100 iteraciones por ejecución, teniendo que llamar a la función 20 veces para poder realizar la comparativa. Esto es una señal de que no gestiona las conexiones de manera eficiente y no permite un gran volumen. En cuanto a PHP, en el servicio web siempre se ha obtenido un rendimiento demasiado pobre.

Servicio Web			
JavaScr	Python	PHP	Java
35	2186	705	146

TABLA 22: TABLAS DE CÁLCULOS DE EJECUCIÓN DE LISTAR IMÁGENES CON 2.000 ELEMENTOS CON SERVICIO WEB

FUENTE: ELABORACIÓN PROPIA.

13. Conclusiones

Finalmente, para cerrar el estudio y desarrollo de este proyecto, en esta sección se explicarán las conclusiones obtenidas a los largo del trabajo

En cuanto al lenguaje más similar al tomado como referencia, C++, ha sido Java. Al tener una sintaxis muy similar, ser un lenguaje compilado y tener el mismo tipado, hace que la sintaxis y la forma de desarrollo sea lo más similar posible. Si bien es cierto que los demás lenguajes no son tan diferentes como para que sea un factor esencial para escoger uno u otro.

En cuanto a eficiencia de desarrollo, ha sido difícil obtener una conclusión real sobre cuál ha sido el mejor. Si tenemos en cuenta el análisis, PHP ha sido el más sencillo debido al framework Laravel y al gestor de paquetes que tiene incluido. Este framework tiene una sintaxis muy sencilla y funcionalidades incluidas que simplifican la vida del desarrollador. Todo esto ha hecho que surgieran el mínimo posibles de errores y bugs, haciendo que el tiempo dedicado sea menor. Además, al contar con un gestor de queries (ORM) para la base de datos, es más sencillo el hacer persistir los datos y la modificación de estos, solo se tuvo que implementar una query para la búsqueda combinada.

En el apartado de facilidad de despliegue, hice una separación entre el despliegue local en el entorno de desarrollo y el despliegue sobre Docker para un entorno de producción real. El más fácil para desplegar sobre local ha sido Java, ya que en el entorno de desarrollo viene configurado el servidor y todo para desplegarlo lo más rápido posible. En cuanto al despliegue de Docker, los lenguajes interpretados han sido más sencillo de desplegar debido a que no hay un proceso de compilación intermedio para generar un paquete, simplemente se copian los archivos al contenedor correspondiente y no es necesario hacer mucho más. En el caso de PHP podría ser el peor de los tres al tener que configurar algún parámetro más del servidor Apache sobre el que se despliega, por lo que Python y JavaScript son los destacados en este aspecto.

En cuanto al rendimiento, existe un ganador y un perdedor claro. JavaScript ha sido el más rápido en el caso de la aplicación y del servicio web, destacando por tener una velocidad de ejecución rápida en comparación con las demás. El claro perdedor ha sido PHP, con un rendimiento muy pobre en ambos casos. Entre Java y Python existen rendimiento similares, menos cuando hay que manejar grandes cantidades de datos, en la que Python claramente obtiene un peor rendimiento que Java.

En la tabla 23, se resume y se valora del uno al cuatro los diferentes aspectos analizados para tratar de obtener un resultado global a todas las características, siendo uno el peor resultado y cuatro el mejor. Tal y como se puede ver en la tabla, JavaScript resulta el ganador una vez asignadas las puntuaciones. PHP, con un total de 12 puntos, es el peor de los cuatro lenguajes comparados.

Como conclusión del análisis y de este proyecto, escoger cada lenguaje y framework es esencial, pero sobre todo hay que tener en cuenta las características de la aplicación que se quiera desarrollar. Esto es imprescindible analizarlo detenidamente en el proceso de definición de un proyecto para después no sufrir las consecuencias de tomar una mala decisión por no tener en cuenta los requisitos.

	JavaScript	Python	PHP	Java
Similitud a lenguaje C++	3	2	2	4
Eficiencia de desarrollo	1	3	4	2
Facilidad de despliegue (Local)	3	3	2	4
Facilidad de despliegue (Docker)	4	4	2	1
Rendimiento	4	2	1	3
Total	15	14	12	14

TABLA 23: RESUMEN DE LAS CARACTERÍSTICAS ANALIZADAS

FUENTE: ELABORACIÓN PROPIA.

Referencias

1. Contributor, T. (2023). Web application (web app). Software Quality.https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app
2. IBM documentation. (s. f.). Servicios Web. https://www.ibm.com/docs/es/was/9.0.5?topic=services-web
3. ¿Qué es HTTP? (s. f.). Cloudflare. https://www.cloudflare.com/es-es/learning/ddos/glossary/hypertext-transfer-protocol-http/
4. ¿Qué es una API y cómo funciona? (s. f.). https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces
5. Most popular technologies - Programming, scripting, and markup languages. (s. f.). Stack Overflow. https://survey.stackoverflow.co/2023/#section-most-popular-technologies-programming-scripting-and-markup-languages
6. Martins, J. (2022, 6 diciembre). Cómo entender los procesos iterativos (con ejemplos) [2022] • Asana. Asana. https://asana.com/es/resources/iterative-process
7. GitHub: Let's build from here. (s. f.). GitHub. https://github.com/
8. TickTick. (s. f.). TickTick: To-do list, tasks, calendar, reminder. TickTick. https://ticktick.com
9. Markdown - la guía definitiva en español. (2022, 14 febrero). Markdown. https://markdown.es/
10. Visual Studio Code - Code editing. Redefined. (2021, 3 noviembre). https://code.visualstudio.com/
11. Syncthing. (2019, 5 septiembre). 2019 The syncthing Foundation. https://syncthing.net/
12. Glassdorr. (s. f.). Glassdoor. https://www.glassdoor.es/Empleo/index.htm
13. https://www.java.com/es/download/manual.jsp
14. JavaScript | MDN. (2023, 24 julio). MDN Web Docs. https://developer.mozilla.org/es/docs/Web/JavaScript
15. Node.js. (s. f.). https://nodejs.org/en
16. ¿Qué es un ataque de scripting entre sitios? definición y explicación. (2023, 19 abril). www.kaspersky.es. https://www.kaspersky.es/resource-center/definitions/what-is-a-cross-site-scripting-attack
17. Inicio - OWASP Top 10:2021. (s. f.). https://owasp.org/Top10/es/
18. illusion Studio - Desarrollo de software, diseño web a medida, aplicaciones web a medida y APPs. (s. f.). ¿Qué es un framework web y qué ventajas aportan?- Diseño web Valencia. ¿Qué es un framework web y qué ventajas aportan?- Diseño web Valencia. https://www.illusionstudio.es/que-es-un-framework-web
19. Madedios. (2023, 21 junio). Frameworks en el desarrollo web: las mejores prácticas para tu negocio online. https://www.wearemarketing.com/es/blog/frameworks-en-el-desarrollo-web-las-mejores-practicas-para-tu-negocio-online.html
20. JavaHispano - Noticias importadas del antiguo JavaHispano.org - Introducción a los motores de plantillas. (s. f.). http://www.javahispano.org/antiguo_javahispano_org/2002/6/1/introduccrion-a-los-motores-de-plantillas.html
21. W3.CSS home. (s. f.). https://www.w3schools.com/w3css/
22. ¿Qué es una API? - Explicación de interfaz de programación de aplicaciones - AWS. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/api/

23. ¿Qué es una API REST? | IBM. (s. f.). https://www.ibm.com/es-es/topics/rest-apis
24. Kinsta. (2023, 13 enero). Qué es Docker: una guía completa. Kinsta®. https://kinsta.com/es/base-de-conocimiento/que-es-docker/
25. Kinsta. (2023b, diciembre 11). Cuota de mercado de PHP en 2024 - Kinsta®. Kinsta®. https://kinsta.com/es/cuota-mercado-php/

Anexo A. Análisis de las imágenes utilizadas

Nombre	Formato	Tamaño(KB)
1	JPG	358
2	JPEG	212
3	JPG	418
4	JPG	1317
5	JPG	1300
6	PNG	1354
7	PNG	315
8	JPG	2275
9	JPG	2713
10	JPG	117
11	JPG	173
12	PNG	170
13	JPG	166
14	PNG	165
15	GIF	9342
16	JPG	371
17	JPG	360
18	JPG	227
19	PNG	190
20	PNG	350