uc3m | Universidad Carlos III de Madrid

University Degree in...

2020-2021

Bachelor Thesis

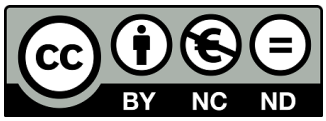
"Thesis title"

Author's complete name

1st Tutor complete name

2nd Tutor complete name

Place and date



SUMMARY

Keywords:

DEDICATION

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Hoy en día, la implementación de la tecnología juega un papel esencial en el ámbito educativo y social, hasta el punto en que se ha convertido en una de las herramientas más populares en numerosos campos, entre ellos el aprendizaje de idiomas y la comunicación entre personas de diferentes países. En este contexto, la creación de una aplicación web de intercambio de idiomas se presenta como una solución innovadora y efectiva para practicar la expresión mediante la corrección colaborativa entre usuarios.

Sin embargo, las aplicaciones de intercambio de idiomas existentes se centran, en muchos casos, en el chat instantáneo, en videollamadas o en ejercicios preparados. Sin embargo, la práctica de la expresión escrita suele quedar en un segundo plano, a pesar de ser una habilidad fundamental tanto en contextos académicos como profesionales. La corrección detallada de textos requiere tiempo y estructura, lo cual construye una base sólida a la hora de aprender un nuevo idioma.

El objetivo de este Trabajo de Fin de Grado es el de diseñar y desarrollar una aplicación web que permita la gestión integral de textos escritos por usuarios que desean practicar idiomas, así como el envío de dichos textos a otros usuarios para su corrección y devolución; y que sirva como herramienta para conectar usuarios de todas partes del mundo. Se trata de una plataforma que busca ser eficiente, práctica y agradable de usar; que optimize el aprendizaje y sirva como una herramienta personal, cercana y social. Al basarse en la redacción, este sistema se presta para ser un lugar donde expresarse, compartir (o simplemente articular) ideas, y abrir paso a la imaginación y a la colección de entradas escritas por uno mismo que no solo podrán servir para la práctica de un idioma sino para conectar y colaborar con otros usuarios.

En definitiva, se busca ofrecer una solución tecnológica que fomente el aprendizaje de idiomas mediante la escritura y la corrección entre pares, mejorando la calidad de la práctica escrita y facilitando la conexión entre personas con intereses lingüísticos comunes.

1.1. Motivación

Desarrollar una aplicación de intercambio de idiomas centrada en la redacción y corrección de textos resulta especialmente interesante por varios motivos.

En primer lugar, responde a una necesidad real de muchos estudiantes de idiomas que buscan practicar la escritura y recibir correcciones de hablantes más avanzados o nativos, sin depender exclusivamente de profesores o academias. Presentar esta práctica en formato de red social lo hace considerablemente más atractivo para las nuevas generaciones, ya que ésta se torna más amena, interactiva y personal.

En segundo lugar, el proyecto permite aplicar y consolidar conocimientos técnicos

adquiridos durante el grado, utilizando tecnologías actuales del ecosistema JavaScript como React, Next.js, Node.js, Express, Mongoose y MongoDB. Dichas tecnologías serán presentadas y estudiadas en más profundidad más adelante en este documento, y su uso e implementación supone una oportunidad para trabajar con una arquitectura moderna, orientada a servicios y preparada para escalar.

Por último, la motivación también es personal: crear una herramienta que no solo tenga valor académico, sino que pueda ser utilizada por usuarios reales, generando comunidad y facilitando el aprendizaje colaborativo de idiomas, que es uno de mis principales intereses personales.

1.2. Objetivos

Los objetivos principales de este proyecto son los de desarrollar una aplicación web que resulte útil para usuarios que desean practicar idiomas mediante la escritura y la corrección colaborativa. De esta manera se pretende mejorar el proceso de aprendizaje, proporcionando una forma estructurada de enviar textos, recibir correcciones y gestionar las relaciones con otros usuarios.

Los objetivos de este Trabajo de Fin de Grado son los siguientes:

- Diseñar e implementar un sistema de autenticación de usuarios que incluya registro, inicio de sesión y recuperación de contraseña.
- Desarrollar una bandeja de cartas enviadas y recibidas, con posibilidad de ver el estado de corrección y filtrar por diferentes campos para mejor accesibilidad.
- Implementar un sistema de relaciones entre usuarios que permita enviar y aceptar solicitudes de amistad, así como gestionar la lista de contactos y explorar usarios nuevos que agregar.
- Crear una página de perfil de usuario con datos editables y opciones de cambio de contraseña y borrado de cuenta.
- Diseñar una interfaz para crear y editar cartas, y enviarlas a amigos para su corrección.
- Desarrollar una página específica para corregir cartas recibidas y reenviarlas al autor con las modificaciones pertinentes.
- Garantizar una arquitectura limpia, mantenible y escalable, utilizando buenas prácticas y un diseño óptimo.
- Ofrecer una experiencia de usuario agradable e intuitiva.

1.3. Marco regulador

Al tratarse de una aplicación que gestiona datos personales de usuarios (correo electrónico, nombre, idioma, datos de perfil, etc.), el proyecto debe tener en cuenta el marco regulador vigente en España y en la Unión Europea:

- Reglamento General de Protección de Datos (RGPD): normativa europea que establece las reglas para el tratamiento de datos personales de los ciudadanos de la UE. La aplicación debe garantizar la confidencialidad, integridad y disponibilidad de los datos, así como ofrecer mecanismos para el ejercicio de derechos (acceso, rectificación, supresión, etc.). Por tanto, es necesario el consentimiento explícito del usuario para la recopilación y uso de sus datos por el software. [1].
- Ley Orgánica de Protección de Datos Personales y garantía de los derechos digitales (LOPDGDD): adapta el RGPD al ordenamiento jurídico español desarrollando en mayor profundidad varios de sus puntos y concretando obligaciones adicionales. Profundiza, por ejemplo, en el establecimiento de reglas sobre las autoridades competentes en esta área o el tratamiento de datos de personas fallecidas. [2].
- Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico (LSSICE): regula la prestación de servicios por vía electrónica, incluyendo aspectos como el acceso por parte del usuario a los datos de identificación del responsable del software, condiciones de uso y posibles comunicaciones comerciales. [3].
- Ley de Propiedad Intelectual: relevante en caso de reutilización de contenidos, textos de ejemplo o recursos de terceros dentro de la aplicación. En este caso es necesario el consentimiento del autor o autores para su utilización. [4].

El diseño y despliegue de la aplicación debe tener en cuenta estos marcos legales, especialmente en lo relativo al tratamiento de datos personales, almacenamiento en la base de datos, opciones de borrado de cuenta, y uso de recursos de terceros.

1.4. Medios empleados

Los elementos de **hardware** son las partes físicas de un sistema informático, que se utiliza para el desarrollo de proyectos de amplia variedad. En el caso de este Trabajo de Fin de Grado, se ha empleado el siguiente dispositivo:

- Ordenador portátil personal: equipo utilizado para el desarrollo de la aplicación y sus servicios, además de pruebas locales y documentación. Estas son sus características:
  - Modelo: LG Gram 16Z90P

  - Sistema operativo: Microsoft Windows 10 Home
  - Procesador: 11th Intel Core i7-1165G7 2.80GHz
  - Memoria RAM: 16 GB
  - Tarjeta gráfica: Intel Iris Xe Graphics

Los elementos de **software** son los programas informáticos que otorgan carácter funcional al hardware, en este caso para hacer posible el desarrollo del proyecto. Los elementos de software utilizados son los siguientes:

- Visual Studio Code [5]: entorno de desarrollo integrado para la programación del front-end y back-end, así como para la redacción de la memoria utilizando el sistema de composición de textos LaTeX.
- Firefox Developer Edition [6]: navegador web utilizado como motor de búsqueda para la investigación de la información aquí presentada (legislación, tecnologías, metodologías, etc.), documentación y resolución de dudas para el desarrollo, y para la ejecución y prueba de la aplicación.
- MongoDB Compass [7]: programa que ofrece una interfaz que facilita la gestión y consulta de los datos de la base de datos.
- Postman [8]: herramienta para verificar el correcto funcionamiento de las rutas de la API.
- GitHub [9]: herramienta de control de versiones y guardado del proyecto que permite mantener el historial y copias de seguridad del código, además de compartirlo de forma práctica.

1.5. Estructura del documento

Esta memoria se estructura en siete capítulos además del índice de contenidos, figuras y tablas:

- **Capítulo 1 – Introducción**: en esta sección se introduce el contexto de este documento y del proyecto, se expone la motivación para su realización, los objetivos que ésta persigue, el marco regulador que encuadra el contexto legislativo en que se desarrolla este software y los medios tanto físicos como digitales empleados, además del apartado actual que incluye la explicación de los apartados que componen el documento.
- **Capítulo 2 – Estado del arte**: esta capítulo cuenta con tres secciones. En la primera y la más extensa, se hace un recorrido por las tecnologías presentes y más utilizadas

actualmente para el desarrollo de aplicaciones web, pasando por los campos de frontend, entornos de trabajo, backend, creación y consumo de APIs, bases de datos, computación en la nube, e inteligencia artificial generativa para la programación. En la segunda sección, se exponen algunas de las metodologías principales para el desarrollo de un proyecto de software. Por último, la tercera sección presenta varios ejemplos de negocio que podrían ser similares a la aplicación aquí desarrollada y por tanto actuar como competencia.

- **Capítulo 3 – Análisis del caso práctico**: requisitos funcionales y no funcionales, casos de uso, historias de usuario (login, cartas, amigos, perfil, correcciones, etc.).
- **Capítulo 4 – Diseño e implementación**: arquitectura de la aplicación, diseño de la base de datos, estructura de carpetas, principales componentes de front-end y endpoints de back-end.
- **Capítulo 5 – Gestión del proyecto**: planificación temporal (sprints), herramientas de organización (Trello, GitHub Projects, etc.), estimación de esfuerzo y posibles costes.
- **Capítulo 6 – Pruebas y validación**: pruebas de aceptación, pruebas funcionales de las principales características (registro, envío de cartas, corrección, gestión de amigos, etc.).
- **Capítulo 7 – Conclusiones y trabajos futuros**: valoración personal, logros alcanzados, limitaciones encontradas y posibles mejoras o ampliaciones de la aplicación.

2. ESTADO DEL ARTE

2.1. Tecnologías para el Desarrollo de Aplicaciones Web

2.1.1. Frontend

El frontend es la parte visible de una aplicación web, con la que el usuario interactúa de forma directa. Se compone de tres tecnologías principales: lenguaje de marcado de hipertexto (HTML), lenguaje de diseño gráfico y lenguaje de programación; que trabajan juntas para definir la estructura, el diseño y la interactividad de la interfaz. Estas tecnologías son esenciales para cualquier desarrollo web, desde sitios estáticos hasta aplicaciones dinámicas y complejas como la que se plantea en este proyecto.

HTML: la estructura de la web

HTML (HyperText Markup Language) es el lenguaje de marcado utilizado para definir la estructura de una página web. Funciona mediante etiquetas que organizan el contenido en elementos como encabezados, párrafos, listas, imágenes y enlaces. Como ejemplo, las etiquetas correspondientes a estos elementos son <h>, <p>, <ol>, <img> y <link>. Existe una gran cantidad de etiquetas para construir los diferentes elementos que se pueden encontrar en una página web. Otros ejemplos de estos elementos son los botones, áreas de texto, formularios, menús, secciones, títulos, y muchos otros. La estructura de una etiqueta o "tag" HTML se compone de una etiqueta de apertura, el contenido, y una etiqueta de cierre:

- La etiqueta de apertura puede contener atributos que especifican información sobre ese elemento, como puede ser su clase, un link a otra página, funciones que se ejecutan cuando el usuario interactúa con ese elemento, y más. Existen atributos opcionales y obligatorios para según qué etiqueta; cada atributo tendrá un valor, y es posible tener tags sin atributos.

- Un tag puede tener contenido, que se especifica entre la etiqueta de apertura y la de cierre. El contenido es el texto, imagen u otro elemento dentro de la etiqueta, lo que se mostrará en la pantalla. Así como existen tags sin atributos, también podemos crear tags sin contenido.

- La etiqueta de cierre marca el fin de la etiqueta. Si la etiqueta carece de contenido, se puede prescindir de la etiqueta de cierra marcando la etiqueta de apertura con un *slash* ("/"), es decir, se puede utilizar *<tag/>* en vez de *<tag></tag>*. A continuación vemos un ejemplo de etiqueta de link a otra página.

A diferencia de otros lenguajes, HTML no tiene lógica de control (como bucles, variables o condicionales), sino que se enfoca exclusivamente en estructurar la información. Por tanto, HTML es el pilar sobre el cual se construyen las páginas. La evolución de HTML ha llevado a versiones más avanzadas, en particular HTML5, que introduce nuevas etiquetas semánticas como <article>, <section> y <nav>, mejorando la organización del contenido y optimizando su accesibilidad. También incorpora soporte para audio, video y gráficos vectoriales sin necesidad de complementos adicionales, lo que amplía las posibilidades de una aplicación web moderna.

Fig. 2.1. Logo de HTML5

CSS: El diseño y la presentación

Para construir el diseño y el aspecto visual de una página web, se utiliza CSS, que quiere decir "Cascade Style Sheets" u hojas de estilo en cascada. Se basa en reglas que seleccionan elementos HTML y les aplican propiedades de estilo (color, forma, márgenes, posición, animaciones, transiciones, y demás). Existen varias formas de incluir CSS en una página web:

- **CSS en línea:** Los estilos se aplican directamente dentro de una etiqueta HTML mediante el atributo `style`. Esta opción es útil para aplicar estilos rápidos a elementos individuales, pero dificulta la mantenibilidad y reutilización del código.
- **CSS interno:** Los estilos se definen dentro de una etiqueta `<style>` en el documento HTML. Esto permite centralizar los estilos dentro del mismo archivo, pero puede volverse poco eficiente en proyectos grandes.
- **CSS externo:** Los estilos se escriben en un archivo separado (.css), permitiendo una mejor organización y reutilización. Esta es la forma más recomendada en proyectos escalables, dado que facilita la separación entre la estructura (HTML) y el diseño (CSS).
- **CSS basado en utilidades:** se utilizan clases predefinidas aplicadas directamente en el HTML. Esta metodología permite construir interfaces rápidamente sin necesidad de escribir reglas CSS personalizadas. Aunque este enfoque facilita la productividad y la coherencia visual, puede generar un código HTML más cargado de clases, lo que a algunos desarrolladores les resulta menos legible. Tailwind CSS es

el framework de CSS de introdujo esta funcionalidad y permite aplicar estilos de esta manera.

Fig. 2.2. Logo de Tailwind

Con la introducción de CSS3, se han añadido características avanzadas como gradientes, sombras, transiciones y animaciones, así como técnicas de diseño responsivo como flexbox y grid, que permiten adaptar la interfaz a distintos tamaños de pantalla. Esto es especialmente importante para que el diseño una aplicación web sea correcto y agradable en cualquier tipo de dispositivo, tanto portátiles como tabletas, teléfonos móviles y otros..

Fig. 2.3. Logo de CSS

JavaScript: La interactividad y la lógica

JavaScript es el lenguaje de programación estándar para implementar la interactividad de un sitio web. Se trada de un lenguaje de propósito general que se ejecuta en el navegador del usuario y permite añadir interactividad y dinamismo a una aplicación web. Mediante JavaScript, una aplicación web puede responder a eventos como clics, introducción de texto o movimientos del ratón sin necesidad de recargar la página. Con esta herramienta se pueden implementar todo tipo de funcionalidades, como validar formularios, actualizar datos en tiempo real, integrar servicios externos y demás funciones.

Un "framework" o entorno de trabajo es una herramienta que ofrece varias funcionalidades y esquemas para desarrollar un proyecto de forma más organizada, ágil y escalable. Con la introducción de frameworks de Javascript, como React.js, Vue.js y Angular, el desarrollo frontend ha evolucionado hacia la creación de aplicaciones de una sola página (SPA), donde JavaScript gestiona la navegación y actualización de contenido sin necesidad de recargar la página; y hacia la construcción de páginas mediante componentes reutilizables, lo que aumenta la escabilidad y acelera el desarrollo de una aplicación compleja.

Otro aspecto fundamental que Javascript permite implementar es el uso de técnicas como AJAX o Fetch API, que permiten comunicarse con un servidor en segundo plano, enviando y recibiendo datos sin interrumpir la experiencia del usuario.

Fig. 2.4. Logo de Javascript

En 2012, vista la falta de herramientas que facilitaran un desarrollo óptimo, apareció Typescript, un *superset* de Javacript que incorporaba clases, módulos y una estructura más organizada y conveniente para proyectos grandes. Un superset de Javascript es una tecnología capaz de compilar y ejecutar código Javascript pero que tiene su propia sintaxis y diferenciaciones, de forma que podemos incluir código Typescript en un proyecto Javascript y viceversa. La principal diferencia que marca Typrescript es que se trata de un lenguaje de tipado estático, es decir, al declarar una variable se debe especificar el tipo de dato de esta, y este no es modificable. Esto no solo facilita la detección de *bugs* o errores en el código, sino que también permite ofrecer sugerencias a la hora de operar con esas variables o tomar argumentos en funciones.

Fig. 2.5. Logo de TypeScript

2.1.2. Frameworks de Javascript

La utilización de un framework de Javascript para el desarrollo de una aplicación web es muy recomendable dadas las funcionalidades preprogramadas que ofrecen agilizar las tareas de programación; funcionalidades como la modularización del código, la optimización del rendimiento, la integración de servicios, la manipulación de eventos y la creación de interfaces dinámicas; además del manejo del DOM (Document Objetc Model), la interfaz de programación que permite crear, eliminar o modificar elementos de la página. Estudiaremos algunas de las alternativas de frameworks que hay actualmente a disposición del programador.

React

React.js es una de las tecnologías más utilizadas en el desarrollo de interfaces dinámicas. Creada por ingenieros de Meta, esta biblioteca de JavaScript permite construir aplicaciones modulares y eficientes gracias a su sistema basado en componentes reutilizables y la utilización de un DOM virtual, que realiza una evaluación de los cambios que se aplicarán al DOM, y los aplica solamente al elemento o los elementos que cambian o se ven afectados por la modificación. Esto mejora el rendimiento en la actualización de la interfaz. Aplicaciones como Facebook, Instagram y Airbnb han adoptado React debido a las facilidades que ofrece y su flexibilidad y ecosistema robusto, además de su facilidad de instalación y relativamente accesible curva de aprendizaje. Actualmente es uno de los frameworks web más populares y utilizados, aunque existen otra opciones también interesantes. Esto implica que existe una gran cantidad de documentación y guías para aprender y resolver dudas sobre React, además de numerosas librerías compatibles con esta tecnologías y componentes gratuitos que uno puede incorporar en sus propios proyectos y que encontramos en plataformas como 21st.dev, Aceternity ui o MUI.

Fig. 2.6. Logo de React

Next

Sobre React se ha construido Next.js, un framework que optimiza el rendimiento al incorporar renderizado del lado del servidor (la generación del contenido de una página web ocurre en el servidor, en lugar de hacerlo en el navegador del cliente buscando una mejora del rendimiento) y generación de páginas estáticas, lo que lo convierte en una muy buena opción para aplicaciones con necesidades de optimización para motores de búsqueda. Empresas como TikTok, Twitch y Hulu han apostado por Next.js debido a su capacidad para mejorar la velocidad de carga y la experiencia del usuario. A pesar de estas ventajas, su configuración inicial puede ser más compleja en comparación con React puro, y algunas de sus funcionalidades avanzadas requieren un aprendizaje adicional. Otra de las principales ventajas de utilizar Next.js es que cuenta con Vercel, una plataforma que nos permite desplegar y alojar proyectos de forma altamente optimizada, rápida y sencilla.

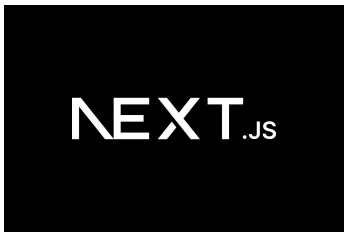


Fig. 2.7. Logo de Next

Vue

Otra opción destacada en frontend es Vue.js, un framework progresivo que ha ganado popularidad por su curva de aprendizaje baja y su sintaxis intuitiva. Facilita la creación de sitios web de una sola página ligeros y con alta velocidad de carga. Empresas como Xiaomi, Alibaba y Nintendo lo utilizan por su facilidad de integración y su capacidad para construir interfaces interactivas con menor esfuerzo en comparación con React o Angular. Además, existe una librería llamada Vuetify que pone a disposición del desarrollador numerosos componentes para utilizar en sus aplicaciones. Una desventaja es que Vue cuenta con una comunidad más pequeña y menos recursos disponibles, lo que puede ser un inconveniente en proyectos grandes.

Fig. 2.8. Logo de Vue

Angular

Angular es un framework completo desarrollado por Google basado en TypeScript que ofrece una arquitectura robusta para el desarrollo de aplicaciones web a gran escala. Gmail y Microsoft Teams lo utilizan debido a su capacidad para estructurar aplicaciones complejas con un fuerte soporte empresarial. No obstante, su curva de aprendizaje es considerablemente más alta que la de React o Vue, y su enfoque basado en clases y módulos puede resultar más rígido.

Fig. 2.9. Logo de Angular

2.1.3. Backend

El backend es la parte de una aplicación que gestiona la lógica de negocio, el procesamiento de datos, la seguridad, y la comunicación con la base de datos y otros servicios. El backend funciona en el servidor y se encarga de recibir peticiones del cliente, procesarlas y devolver una respuesta adecuada. Hay diversas opciones de tecnologías que permiten llevar a cabo estos objetivos. Entre as más significativas a día de hoy se encuentran: Node.js, Django, Spring Boot, Laravel, Ruby, y Gin.

Node.js

Node.js es un entorno de ejecución basado en JavaScript que permite ejecutar código en el servidor. Su principal ventaja es que permite desarrollar tanto el frontend como el backend con el mismo lenguaje, lo que facilita la comunicación entre ambas partes. Su arquitectura está basada en eventos y es no bloqueante, es decir, el sistema no espera a que se termine de ejecutar cada tarea o consulta para empezar con la siguiente, sino que libera el flujo de ejecución para llevar a cabo otras tareas mientras tanto. Esto lo hace altamente eficiente en aplicaciones que requieren muchas conexiones simultáneas. Es posible cargar y utilizar una amplia gama de paquetes con npm (Node Package Manager), que ofrece miles de bibliotecas para ampliar sus funcionalidades y agilizar el desarrollo; y permite dar soporte y servicio en tiempo real a una aplicación. Empresas como Netflix, PayPal y Uber utilizan Node.js dada su capacidad para manejar grandes volúmenes de tráfico y su escalabilidad. Sin embargo, no es la mejor opción para aplicaciones que requieren cálculos intensivos, como procesamiento de imágenes o simulaciones matemáticas complejas. Dentro del ecosistema de Node.js, destaca el Express.js, un framework minimalista que facilita la creación de APIs y servidores web. Es rápido, flexible y ampliamente utilizado en el desarrollo de aplicaciones modernas.

Fig. 2.10. Logo de NodeJS

Python con Django

Django es un framework de Python que se encuentra entre uno de los frameworks más utilizados en el ámbito del backend debido a su enfoque en la seguridad y el desarrollo rápido.

Python es un lenguaje de programación conocido por su simplicidad y legibilidad, con una sintaxis clara y cercana al lenguaje humano. Su flexibilidad y estructura intuitiva, basada en indentación en lugar de símbolos complejos, lo hace fácil de entender y aprender. Por ello, es una buena opción tanto para principiantes como para desarrolladores en áreas diversas, desde inteligencia artificial hasta desarrollo web.

Fig. 2.11. Logo de Python

Django tiene un Enfoque "batteries included", es decir, ofrece muchas herramientas listas para usar: cuenta con ORM (Mapeo Objeto-Relacional), integrado, una herramienta que convierte los objetos de una aplicación en un formato compatible para guardarlos en una base de datos y así permitir al desarrollador interactuar con bases de datos sin escribir código SQL; y herramientas de Seguridad avanzada, protegiendo contra ataques comunes como inyección SQL y cross-site scripting (XSS).

Django es ampliamente utilizado en aplicaciones como Instagram, Spotify y Pinterest. No obstante, es un framework monolítico, es decir, todo su código está integrado en una sola estructura. Por ello, es menos flexible cuando se necesita dividir la aplicación en partes independientes que trabajen de forma autónoma, como ocurre en arquitecturas basadas en microservicios, donde cada parte de la aplicación se maneja de manera independiente y puede ser escalada o actualizada por separado.

Fig. 2.12. Logo de Django

Java con SpringBoot

Java es una de las opciones más utilizadas en el desarrollo backend, especialmente en entornos empresariales, dadas la robustez de sus funciones de seguridad y su sólido rendimiento en tareas que demandan un uso intensivo del procesador como cálculos matemático avanzados o intérpretes de lenguaje. El framework más popular dentro de este ecosistema es Spring Boot, que agiliza la creación de aplicaciones escalables y distribuidas. Entre las ventajas de Spring Boot encontramos su arquitectura modular, que permite crear servicios independientes y fácilmente escalables; su alta seguridad y estabilidad, lo que lo hace ideal para aplicaciones bancarias y corporativas; y su compatibilidad con múltiples bases de datos y herramientas empresariales. Sin embargo, Java tiene una curva de aprendizaje más alta y puede ser más pesado en comparación con otras opciones como Node.js o Python.

Fig. 2.13. Logo de Java

Fig. 2.14. Logo de SpingBoot

PHP y Laravel

PHP ha sido históricamente uno de los lenguajes más utilizados para el desarrollo web, especialmente en sistemas de gestión de contenido como WordPress, que permite crear

páginas como blogs o tiendas en línea sin a penas conocimientos técnicos. PHP nos permite agregar dinamismo a nuestros sitios web conectando el servidor y base de datos con la interfaz de usuario. En la actualidad, Laravel es el framework más moderno dentro del ecosistema PHP, ofreciendo una sintaxis elegante y herramientas avanzadas para el desarrollo de aplicaciones web. Laravel cuenta con un sistema de enrutamiento y autenticación integrado, buena documentación y comunidad activa. Además, existe ORM Eloquent, que es un Mapeador de Objetos Relacionales especialmente diseñado para Laravel, que ofrece una experiencia de conexión con la base de datos más ágil y simple. Se trata de una opción sólida para proyectos que necesitan desarrollo rápido, pero PHP no es tan popular en nuevas aplicaciones escalables como Node.js o Django. Entre sus desventajas encontramos herramientas de debugging no tan potentes, los comunes problemas de seguridad, y el hecho de que es necesario un servidor web para que la app funcione.

Fig. 2.15. Logo de PHP

Fig. 2.16. Logo de Laravel

Ruby on Rails

Ruby es un lenguaje de programación con una sintaxis cercana al lenguaje humano y orientado a objetos. Ruby on Rails es un framework basado en el lenguaje Ruby que se centra en la productividad del desarrollador y la facilidad de uso dado que toma decisiones por el desarrollador basándose en convenciones preestablecidas e incluye una gran cantidad de herramientas integradas para tareas comunes. La generación automática de código que ofrece Ruby On Rails acelera el desarrollo; y además la seguridad está integrada, protegiendo contra ataques comunes. Aplicaciones como GitHub y Airbnb fueron construidas con esta tecnología debido a su facilidad para construir aplicaciones rápidamente. El principal inconveniente de Rails es que su rendimiento puede ser inferior al de otras tecnologías más modernas, como Node.js o Go.

Fig. 2.17. Logo de Ruby

Fig. 2.18. Logo de Ruby On Rails

Golang y Gin

Golang, también conocido como Go, es un lenguaje desarrollado por Google enfocado en el rendimiento, la seguridad y la concurrencia a la hora de atender solicitudes, lo cual es ideal para sistemas que requieren gran escalabilidad.. Plataformas como Dropbox, Netflix o Uber están programadas con Go. Se trata de una alternativa de Alto rendimiento, comparable al de C/C++, ya que utiliza Código compilado, es decir, el código fuente es compilado para traducirse a código máquina, lo que mejora la velocidad de ejecución. Su framework más popular para el backend es Gin, que permite construir APIs rápidas y eficientes. Sin embargo, Go tiene menos bibliotecas y comunidad que otras tecnologías más maduras como Node.js o Python. Además, su funcionamiento puede resultar complejo en ciertos puntos, por ejemplo, en ciertas situaciones se requiere manejar manualmente la memoria para optimizar la aplicación.

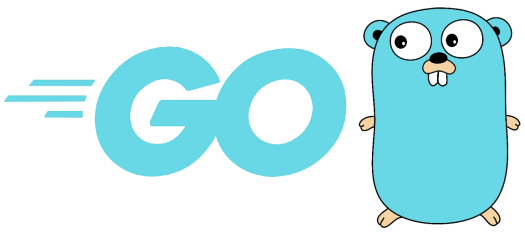


Fig. 2.19. Logo de Go

Fig. 2.20. Logo de Gin

2.1.4. Creación y consumo de APIs

Estos frameworks y tecnologías backend que son útiles entre muchas otras cosas para la creación de APIs. Una API o "Interfaz de Programación de Aplicaciones" es un mecanismo que permite conectar dos piezas de software para que una pueda interactuar con la otra, intercambiar información, invocar funciones o utilizar servicios. Existen diferentes tecnologías que nos permiten consumir o utilizar una API desde una aplicación web, destacando Axios, Fetch API y cURL; además de herramientas que ayudan a probar una API que uno quiere usar o desarrollar, como Postman e Imsomia.

Axios

Axios es una librería basada en JavaScript que permite la realización de solicitudes HTTP tanto en el navegador como en el servidor utilizando Node.js. HTTP o "Hypertext Transfer Protocol" es un protocolo de comunicación para transferir de datos entre un cliente (generalmente una aplicación o navegador) y un servidor web a través de Internet. La comunicación se lleva a cabo mediante solicitudes y respuestas: el cliente realiza una solicitud HTTP al servidor para obtener un recurso (como una página, un archivo, una imagen o cualquier dato), y el servidor responde con la información solicitada en caso de éxito. Axios soporta métodos estándar de HTTP y es especialmente popular debido a su capacidad para manejar promesas de manera eficiente, lo que permite gestionar de forma más sencilla la asincronía en las aplicaciones web. Entre sus ventajas se destacan la posibilidad de interceptar solicitudes o respuestas, manejar errores de manera más eficaz y la capacidad de enviar datos en formato JSON de forma predeterminada. Dado su amplio uso en aplicaciones basadas en frameworks como React o Vue.js, Axios se ha consolidado como una de las opciones más elegidas por los desarrolladores.

Fig. 2.21. Logo de Axios

FetchAPI

Fetch API es una interfaz nativa en JavaScript que proporciona una forma moderna y flexible de hacer solicitudes HTTP desde el navegador. A diferencia de otras soluciones, como Axios, Fetch es parte del estándar de JavaScript, por lo que no requiere la instalación de bibliotecas adicionales. Esta API permite trabajar con promesas para manejar solicitudes asincrónicas y es compatible con los métodos HTTP más utilizados, como GET, POST, PUT y DELETE. Aunque Fetch no ofrece algunas características avanzadas como la transformación automática de respuestas JSON o manejo de errores, su simplicidad y flexibilidad lo convierten en una excelente opción para aplicaciones web modernas que requieren realizar solicitudes HTTP sin depender de bibliotecas externas.

Fig. 2.22. Logo de FetchAPI

Postman

Postman es una herramienta gráfica ampliamente utilizada para probar, desarrollar y gestionar APIs. Permite enviar solicitudes HTTP, analizar las respuestas y organizar las pruebas en colecciones, lo cual resulta muy útil durante el desarrollo de servicios web y microservicios. Postman es particularmente valioso cuando se trabaja con APIs REST. Una API REST es una API que sigue los principios REST (Representational State Transfer): opera mediante métodos del protocolo HTTP, que son GET para recuperar información del servidor, POST para enviar datos al servidor, PUT para actualizar esos datos, DELETE para borrarlos, y otros; cada recurso está asociado a una URL única; y cada solicitud contiene toda la información necesaria para ser completada, no solo no necesita ni depende de datos de otras solicitudes, sino que no se transmiten ni comparten datos de una petición a otra. También, Postman permite visualizar respuestas en diversos formatos, como JSON, XML y texto plano; y ofrece funciones avanzadas como la automatización de pruebas, la creación de entornos y la automatización del proceso de integración y despliegue del código. Todo esto lo convierte en una herramienta muy potente en el proceso de desarrollo y pruebas de APIs.

Fig. 2.23. Logo de Postman

Imsomnia

Insomnia es una herramienta similar a Postman, diseñada para facilitar la prueba y el consumo de APIs. Insomnia se distingue por su interfaz limpia y sencilla, que permite a los desarrolladores trabajar de manera eficiente sin distracciones. Al igual que Postman, Insomnia soporta solicitudes HTTP y puede gestionar entornos, variables y autenticación. Además, cuenta con soporte nativo para GraphQL,un lenguaje de consulta y entorno de ejecución para las APIs que permite obtener varios recursos con una sola consulta y obtener solamente los datos deseados, sin datos extra o campos que no queremos en ese momento. Una de sus principales ventajas es la posibilidad de crear pruebas automatizadas, lo que optimiza el proceso de desarrollo y mejora la calidad del código. Insomnia también es altamente personalizable y permite a los usuarios gestionar sus solicitudes de forma organizada, lo que lo hace adecuado tanto para desarrolladores principiantes como para profesionales experimentados.

Fig. 2.24. Logo de Imsomnia

cURL

cURL es una herramienta de línea de comandos ampliamente utilizada para realizar solicitudes HTTP a través de un terminal o script. Aunque no proporciona una interfaz gráfica, cURL es extremadamente versátil y potente, permitiendo a los desarrolladores interactuar con APIs mediante métodos como GET, POST, PUT y DELETE de forma directa. Su principal ventaja es su capacidad para ser integrado en scripts y automatizar tareas de manera eficiente, lo que lo convierte en una herramienta útil para aquellos que prefieren trabajar desde la línea de comandos o necesitan realizar pruebas en entornos sin interfaz gráfica. cURL es compatible con una amplia variedad de protocolos y es especialmente útil para realizar pruebas rápidas y depurar servicios web.

Fig. 2.25. Logo de cURL

2.1.5. Base de datos

Una base de datos es una recopilación de información que se almacena en un sistema informático y a la que se puede acceder mediante consultas. Existen dos tipos principales de bases de datos: relacionales y no relacionales.

En una base de datos relacional, los datos se almacenan en tablas, donde cada fila representa un registro y cada columna contiene un atributo o campo de ese registro. Cada registro suele tener un identificador único, conocido como clave primaria, que permite distinguirlo de los demás. Existen relaciones entre las tablas que permiten organizar los datos de forma lógica y estructurada siguiendo un modelo relacional. Para escribir y leer datos de este tipo de bases de datos, se utiliza un lenguaje de consulta estructurado (SQL), como MySQL, PostgreSQL o MariaDB.

Por otro lado, las bases de datos no relacionales o NoSQL ofrecen un enfoque más flexible ya que, en lugar de seguir una estructura rígida de tablas, almacenan la información en formatos como documentos JSON, pares clave-valor, grafos o familias de columnas. Esta estructura es especialmente útil cuando los datos cambian con frecuencia o no requieren un esquema fijo. Esto permite insertar, modificar y obtener los datos de forma más dinámica, lo que se hace mediante lenguajes de consulta más modernos. Este tipo de bases de datos ha ganado popularidad en el campo de las aplicaciones web modernas, microservicios y sistemas que manejan grandes volúmenes de datos no estructurados. MongoDB, Cassandra, Firebase y ElasticSearch se encuentran entre las bases de datos NoSQL más utilizadas a día de hoy.

Bases de datos relacionales

MySQL MySQL es uno de los sistemas de gestión de bases de datos más populares y utilizados en aplicaciones web. Originalmente desarrollado por MySQL AB y posteriormente adquirido por Oracle, MySQL es conocido por su fiabilidad, integridad, velocidad y facilidad de uso. Su arquitectura de replicación maestro-esclavo permite distribuir la carga de trabajo y mejorar la disponibilidad. Aunque es de código abierto, también ofrece versiones comerciales con soporte adicional. MySQL ha sido fundamental en la creación de aplicaciones dentro de la pila LAMP (Linux, Apache, MySQL, PHP/Python), y grandes plataformas como Facebook y Twitter lo emplean para manejar grandes volúmenes de

datos.

Fig. 2.26. Logo de MySQL

PostgreSQL PostgreSQL destaca por su capacidad para gestionar transacciones complejas y grandes volúmenes de datos, manteniendo siempre la integridad relacional. Cuando varios usuarios acceden simultáneamente a una base de datos, los sistemas tradicionales suelen bloquear los registros para evitar conflictos entre lecturas y escrituras. Una de sus principales ventajas reside en que gestiona esta concurrencia de manera más eficiente gracias a su implementación de MVCC (Control de Concurrencia Multiversión), que permite que las operaciones de lectura no interfieran con las de escritura y viceversa. Esto se traduce en una mejora significa del rendimiento y la experiencia en entornos con múltiples usuarios accediendo a los mismos datos. Además, PostgreSQL es compatible con múltiples lenguajes de programación, como JavaScript, C/C++, Python, Ruby y demás. Aplicaciones de alto perfil como Instagram, Reddit y Discord emplean PostgreSQL debido a su estabilidad, capacidad para manejar grandes cantidades de datos y su sistema avanzado de consultas.

Fig. 2.27. Logo de Postgresql

Maria DB MariaDB es un sistema de gestión de bases de datos relacional de código abierto que nació como una bifurcación de MySQL, creado por su fundador original, Michael Widenius, después de la compra de MySQL por Oracle. Este sistema mantiene una alta compatibilidad con MySQL, lo que facilita la migración entre ambos. MariaDB ofrece mejoras en términos de rendimiento y seguridad, incorporando nuevos motores de almacenamiento como Aria y XtraDB. Además, su enfoque en la escalabilidad y la alta disponibilidad lo hace adecuado para aplicaciones que requieren un rendimiento constante. Es ampliamente utilizado por empresas de gran tamaño y cuenta con una comunidad activa que garantiza una evolución constante.

Fig. 2.28. Logo de MariaDB

No relacionales

MongoDB MongoDB es una base de datos NoSQL que ha ganado una gran popularidad en aplicaciones web modernas. Su principal característica es que almacena los datos en formato de documentos BSON, una versión de JSON en formato binario; lo que permite un enfoque flexible y escalable. JSON es el formato en el que se intercambian los datos en JavaScript, por lo que combinar estas dos tecnologías es una configuración ágil para un proyecto. Estos documentos se pueden agrupar en colecciones para facilitar la organización de los datos. MongoDB es ideal para aplicaciones que necesitan escalabilidad horizontal, ya que permite distribuir los datos entre varios servidores de manera eficiente. Además, permite modificar fácilmente los datos almacenados sin tener que realizar grandes cambios en la base de datos. Empresas como Uber, eBay y Trello utilizan MongoDB debido a su capacidad para manejar grandes cantidades de datos sin la rigidez de las bases de datos tradicionales, además de su gran fiabilidad y alto rendimiento.

Fig. 2.29. Logo de MongoDB

Firebase Firebase es un servicio de base de datos de Google que almacena los datos en la nube y está diseñado principalmente para aplicaciones web y móviles que requieren sincronización en tiempo real, dado que facilita la actualización instantánea de los datos en todas las plataformas de manera eficiente, lo que la convierte en una opción atractiva para proyectos como aplicaciones de mensajería o colaboración. Su integración con el ecosistema de Google permite aprovechar otros servicios como Firebase Authentication y Google Cloud Storage. Además, su facilidad de uso lo hace adecuado para desarrolladores que trabajan en proyectos pequeños o medianos, dado que permite desarrollar aplicaciones de manera rápida sin la necesidad de gestionar una infraestructura compleja. No obstante, Firebase puede generar dependencia de la plataforma Google, lo que podría convertirse en un inconveniente si se desea cambiar de proveedor a largo plazo. Además,

los costos asociados con su uso pueden incrementarse a medida que la aplicación escala, lo que puede ser una preocupación para proyectos de gran tamaño.

Fig. 2.30. Logo de Firebase

Cassandra Cassandra es una base de datos no relacional distribuida diseñada para manejar grandes volúmenes de datos en entornos de alta velocidad y disponibilidad y sin un único punto de fallo gracias a que opera mediante varios nodos o servidores cada uno de los cuales almacena una parte de los datos y responde a una parte de las consultas. Además, su arquitectura es distribuida, es decir, cada nodo tiene réplicas de los datos de los demás, y es por eso que, si uno falla, los demás pueden atender sus peticiones. Cassandra utiliza un modelo basado en columnas en lugar de filas, de forma que cada fila o registro puede tener columnas diferentes, y Cassandra puede acceder a una columna específica en lugar de obtener toda la fila, lo que aumenta la velocidad de acceso a la información. Empresas que requieren una infraestructura que pueda escalar horizontalmente y manejar grandes volúmenes de datos, como Netflix y eBay, utilizan Cassandra para gestionar su información.

Fig. 2.31. Logo de Cassandra

2.1.6. Computación en la nube

La computación en la nube o cloud computing es un término que se ha hecho popular en los últimos años por ser una herramienta que facilita enormemente el acceso a recursos informáticos permitiendo a las empresas y usuarios almacenar, procesar y gestionar datos sin necesidad de contar con infraestructura propia, lo que reduce costos, mejora la eficiencia y facilita la colaboración y el acceso remoto desde cualquier lugar del mundo. Amazon Web Services (AWS) fue pionero en esta tecnología con el lanzamiento de su servicio de computación en la nube en 2006. A partir de ese momento, esta tecnología comenzó a ganar terreno, y se popularizó y consolidó en los últimos 10 años, a medida

que más empresas y usuarios adoptaron la nube para almacenar datos, ejecutar aplicaciones y aprovechar las capacidades de procesamiento a gran escala de manera flexible y rentable. Estas herramientas no solo nos permiten ofrecer servicios a los usuarios de una aplicación, sino también alojar dicha aplicación. Algunos de los proveedores más significativos actualmente son, además de AWS: Microsoft Azure y Google Cloud Platform (GCP).

Amazon Web Services

Amazon Web Services es la plataforma en la nube más popular y ampliamente adoptada a nivel mundial. Ofrece una extensa gama de servicios que cubren prácticamente todos los aspectos de la infraestructura tecnológica, desde servidores y almacenamiento hasta bases de datos, redes y herramientas de inteligencia artificial. Con su servicio de cómputo EC2 (Elastic Cloud 2), los usuarios pueden crear y gestionar servidores virtuales, mientras que Amazon S3 (Simple Storage Service) proporciona un almacenamiento de alta seguridad y rendimiento. AWS también incluye servicios sin servidor como AWS Lambda, que permiten ejecutar código sin tener que gestionar servidores; herramientas para el análisis de datos, bases de datos gestionadas, redes, y mucho más. Su capacidad de escalabilidad, flexibilidad y fiabilidad lo convierte en la opción por al que optan muchas empresas que requieren una infraestructura altamente personalizable y capaz de manejar aplicaciones a gran escala.

Fig. 2.32. Logo de AWS

Microsoft Azure

Microsoft Azure es una plataforma en la nube de Microsoft diseñada para ayudar a las empresas a construir, desplegar y gestionar aplicaciones y servicios a través de una red global de centros de datos. Azure es muy conocida por su integración con otros productos de Microsoft, como Windows Server, SQL Server, y herramientas de desarrollo como Visual Studio. Ofrece soluciones completas para el desarrollo y despliegue de aplicaciones web, como Azure App Services, que permite crear aplicaciones web sin gestionar la infraestructura subyacente. Además, Azure cuenta con servicios de bases de datos, análisis de datos con Azure Synapse, herramientas de inteligencia artificial con Azure AI, y servicios de computación en contenedores con Azure Kubernetes Service (AKS). Es una plataforma versátil y especialmente atractiva para empresas que ya utilizan el ecosistema

de Microsoft, dado que proporciona un entorno seguro y fácil de integrar.

Fig. 2.33. Logo de Microsoft Azure

Google Cloud Platform

Google Cloud Platform es la plataforma de computación en la nube de Google. Se distingue por sus soluciones avanzadas en big data, análisis y machine learning. GCP ofrece herramientas como Google Compute Engine para crear y gestionar máquinas virtuales, Google App Engine para lanzar aplicaciones sin servidor y Google Kubernetes Engine para gestionar contenedores. Firebase es parte de GCP y proporciona bases de datos en tiempo real, autenticación de usuarios y funciones de backend como análisis y notificaciones push, que son mensajes que una aplicación envía al usuario en tiempo real sin que este tenga que estar usando la app en ese momento. GCP también destaca por su capacidad para manejar grandes volúmenes de datos y ofrecer soluciones avanzadas en inteligencia artificial y aprendizaje automático, lo que lo convierte en una opción ideal para aplicaciones que requieren procesamiento de datos en tiempo real o capacidades analíticas avanzadas. Además, su infraestructura está respaldada por la misma red de centros de datos que Google utiliza para sus propios servicios, ofreciendo una alta fiabilidad y rendimiento.

Fig. 2.34. Logo de Google Cloud Platform

2.1.7. Inteligencia Artificial Generativa para el programador

En la actualidad, el avance de las herramientas de inteligencia artificial generativa ha llevado a una parte importante de la población a utilizarla para facilitarse tareas de todo tipo. La IA está transformando la forma en que trabajamos y en particular, para los programadores, estas tecnologías resultan especialmente útiles a la hora de abordar tareas mecánicas y repetitivas, como la generación de código, la depuración y la interpretación

de fragmentos de código. Los desarrolladores pueden obtener asistencia para comprender y solucionar problemas técnicos, optimizando su flujo de trabajo y acelerando el desarrollo de software. ChatGPT, Gemini, Codex y Github Copilot son algunas de las opciones más populares en este sector.

ChatGPT

ChatGPT, desarrollado por OpenAI, es una de las opciones más populares de inteligencia artificial generativa orientada a la programación. Esta herramienta utiliza modelos avanzados de lenguaje natural para asistir en tareas como la generación de código, depuración y explicación de fragmentos de programación. ChatGPT puede ser de gran ayuda para programadores novatos que necesitan comprender conceptos complejos, así como para desarrolladores experimentados que buscan optimizar su flujo de trabajo. Su capacidad para responder a preguntas técnicas, sugerir soluciones a problemas específicos y generar ejemplos de código en diversos lenguajes de programación hace de ChatGPT una herramienta versátil. Además, con su integración en plataformas como IDEs o herramientas de productividad, los desarrolladores pueden acceder a su soporte en tiempo real durante el proceso de codificación.

Fig. 2.35. Logo de ChatGPT

Gemini

Gemini es otro sistema de inteligencia artificial generativa que llegó en 2023 de la mano de Google DeepMind para posicionarse como principal competidor de ChatGPT. Puede asistir en tareas de programación, especialmente en la generación de código y la interpretación de problemas técnicos. Aunque se lanzó recientemente, Gemini ha sido diseñado con capacidades avanzadas para comprender y generar texto en múltiples lenguajes de programación. La diferencia clave de Gemini respecto a otras plataformas es su enfoque en la integración con otras herramientas de Google, como los servicios de Google Cloud, lo que lo convierte en una opción muy útil para quienes ya están trabajando en este ecosistema. Al igual que otros modelos de IA generativa, Gemini puede ayudar a los desarrolladores a encontrar soluciones a errores comunes, automatizar tareas repetitivas y generar código funcional a partir de descripciones en lenguaje natural.

Fig. 2.36. Logo de Gemini

Codex

Codex, también desarrollado por OpenAI, es otro modelo de inteligencia artificial generativa especializado en la programación. Está diseñado para comprender el lenguaje natural y generar código en múltiples lenguajes de programación, como Python, JavaScript, Ruby, entre otros. Codex puede integrarse con diversas plataformas y herramientas, como GitHub Copilot, proporcionando asistencia contextual mientras el usuario escribe código. Esta IA no solo ayuda a generar fragmentos de código, sino que también puede realizar tareas complejas, como la conversión de pseudocódigo a código real y la mejora de código existente. Codex ha demostrado ser útil para mejorar la productividad de los desarrolladores, ayudándoles a escribir código más rápido, depurar errores y generar soluciones para problemas complejos sin la necesidad de una intervención manual constante.

Fig. 2.37. Logo de Codex

Github Copilot

GitHub Copilot es una herramienta de inteligencia artificial desarrollada por GitHub en colaboración con OpenAI, diseñada para asistir a los desarrolladores en tiempo real mientras el escriben código. Ofrece autocompletado avanzado y generación de funciones completas a partir de descripciones en lenguaje natural. Una sus las principales ventajas es su análisis del contexto del archivo y del proyecto para ofrecer sugerencias más precisas y adaptadas a la tarea en la que se está trabajando. Copilot es compatible con múltiples lenguajes de programación y se integra con entornos de desarrollo populares como Visual Studio Code, Visual Studio y otros IDEs. Además, Copilot puede ayudar a crear pruebas unitarias, reducir errores, explicar código existente y proponer mejoras de forma rápida y eficiente.

Fig. 2.38. Logo de Github Copilot

2.2. Metodología

A lo largo de los años, la forma en que se abordan los proyectos de desarrollo web ha evolucionado significativamente, desde enfoques rígidos y estructurados hasta metodologías más flexibles y colaborativas. Cada método ha surgido en respuesta a distintas necesidades dentro del desarrollo de software y encaja con un tipo de proyecto u otro dependiendo de sus requisitos y equipo de trabajo. A continuación, se presentan algunas de las metodologías más utilizadas en proyectos de desarrollo web.

Metodología en Cascada

Este modelo es uno de los más antiguos en el desarrollo de software y sigue un enfoque secuencial, en el que cada fase debe completarse antes de pasar a la siguiente. Se divide en etapas bien definidas como análisis, diseño, implementación, pruebas y mantenimiento. Su principal ventaja es la claridad en la planificación, pero su rigidez puede dificultar la adaptación a cambios durante el proyecto.

Metodología Incremental

La metodología incremental es un enfoque de desarrollo de software que consiste en construir el sistema por partes, entregando versiones más desarrolladas en cada etapa. Cada incremento añade nuevas funcionalidades al producto, se avanza de forma progresiva hasta completar el sistema final. Este método facilita la detección temprana de errores y la incorporación de cambios, ya que se obtiene retroalimentación continua del cliente. Además, permite entregar valor desde las primeras fases del proyecto. Por otro lado, el sistema requiere de una planificación inicial bastante completa, ya que los incrementos dependen de una buena definición de requisitos desde el principio.

Metodología RAD

La metodología RAD (Desarrollo Rápido de Aplicaciones) se utiliza para desarrollar software partiendo del diseño de un prototipo funcional que se prueba con los usuarios desde etapas tempranas, permitiendo identificar fallos y nuevos requerimientos. Basándose en

esa retroalimentación, se establecen prioridades para realizar mejoras rápidas y continuas, enfocándose en la velocidad de ejecución y en la entrega de resultados en plazos cortos.

Metodología Agile

Agile es un enfoque iterativo e incremental que busca la entrega rápida de software funcional mediante ciclos cortos de trabajo llamados "sprints". Se basa en la colaboración entre equipos multidisciplinarios, la adaptación continua a los cambios y la entrega de valor al cliente de manera constante, además de priorizar el sofware frente a la documentación. Existen varias metodologías ágiles o entornos de trabajos que aplican diferentes enfoques para conseguir objetivos más concretos. Uno de los más populares es Scrum, donde el proceso comienza con la planificación del Sprint, seleccionando tareas del la lista priorizada de funcionalidades que el producto necesita (*Product Backlog*) para formar una lista reducida que debe estar hecha al terminar ese sprint (*Sprint Backlog*). Durante el Sprint, el equipo trabaja colaborativamente y realiza reuniones diarias para revisar el progreso. Al finalizar, se entrega un incremento del producto y se hace un trabajo de retrospección y análisis del proceso, con el objetivo de mejorar continuamente en el siguiente ciclo.

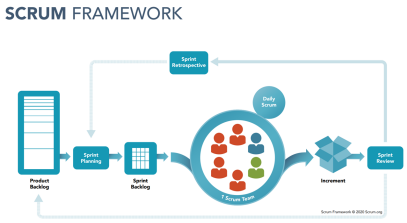


Fig. 2.39. Metodología SCRUM

Un marco de trabajo similar a Scrum es Extreme Programming o XP, aunque este tiene un enfoque más técnico con métodos para los desarrolladores como la programación en pareja, la integración continua, el desarrollo basado en pruebas, y la refactorización iterativa y continua mediante ciclos de retroalimentación. XP tiene un enfoque muy técnico para asegurar la calidad y flexibilidad del software.

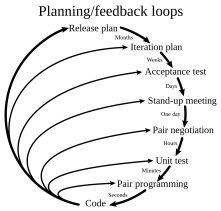


Fig. 2.40. Metodología Extreme Programming

Metodología Kanban

Kanban es una metodología con origen en Japón que ayuda a gestionar el flujo de trabajo de manera eficiente mediante un tablero con representaciones visuales del flujo de trabajo y las tareas a realizar, en proceso y terminadas. Estos tableros con columnas también pueden representar las diferentes etapas del desarrollo, permitiendo un control más flexible de las tareas y facilitando la identificación de cuellos de botella, que son limitaciones que restringen el flujo del trabajo o lo bloquean, y del número de tareas que se están llevando a cabo al mismo tiempo. Esta metodología es ideal para equipos que buscan mejorar la productividad de forma simple, rápida, visual y flexible.

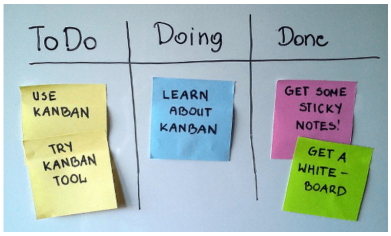


Fig. 2.41. Metodología Kanban

Metodología Lean

Derivada de los principios de manufactura de Toyota, Lean busca minimizar el desperdicio de tiempo y recursos, a la vez que maximizar la eficiencia. Se enfoca en la entrega de valor al cliente de la manera más rápida y eficiente posible, eliminando tareas innecesarias y fomentando la mejora continua. Esta metodología prescinde de una retroalimentación frecuente para asegurar una evaluación y mejora continua, tiene un enfoque perfeccionista.

2.3. Negocio

Hoy en día, existen varias plataformas y aplicaciones centradas en el aprendizaje colaborativo de idiomas, alguna de ellas, similar al proyecto aquí presentado, permiten la corrección de textos por parte de hablantes nativos. Estas herramientas varían en su enfoque, desde la corrección automática con inteligencia artificial hasta el intercambio lingüístico entre usuarios. Se expondrán varias de estas alternativas para estudiar en qué se diferenciará el proyecto aquí expuesto de estas y qué ideas, características y funcionalidades nuevas aporta.

2.3.1. HiNative

HiNative es una de las plataformas más conocidas en este ámbito, diseñada para resolver dudas sobre el uso de un idioma a través de preguntas y respuestas. Los usuarios pueden

consultar sobre gramática, pronunciación y expresiones cotidianas, obteniendo respuestas de hablantes nativos. No obstante, HiNative no está enfocada en la corrección de textos largos, sino en consultas puntuales, lo que limita su utilidad para quienes buscan mejorar su escritura de manera estructurada. Además, al depender de la comunidad, la calidad y rapidez de las respuestas pueden variar considerablemente.

Fig. 2.42. Logo de HiNative

2.3.2. Lang-8

Por otro lado, Lang-8 ofrece un sistema más orientado a la corrección de textos completos. Los usuarios publican textos en el idioma que están aprendiendo y reciben correcciones detalladas de hablantes nativos. Esta plataforma permite obtener retroalimentación más estructurada que HiNative, ya que el objetivo es mejorar la redacción en su conjunto. Sin embargo, el proyecto ha dejado de recibir actualizaciones y no cuenta con una integración eficiente en dispositivos móviles, lo que ahora limita su accesibilidad y usabilidad.

Fig. 2.43. Logo de Lang-8

2.3.3. Tandem

Otra alternativa es Tandem, una aplicación centrada en el intercambio lingüístico en tiempo real. A través de mensajes de texto, llamadas de voz y videollamadas, los usuarios pueden practicar con hablantes nativos, incluyendo la opción de corregir mensajes dentro del chat. No obstante, Tandem prioriza la interacción oral y escrita breve, sin ofrecer herramientas específicas para la corrección detallada de textos largos ni la posibilidad de almacenarlos para un seguimiento a largo plazo.

Fig. 2.44. Logo de Tandem

2.3.4. Grammarly

En el ámbito de la corrección automática, herramientas como Grammarly proporcionan análisis gramatical y sugerencias estilísticas mediante inteligencia artificial. Si bien son útiles para detectar errores básicos y mejorar la fluidez del texto, su funcionamiento se basa en reglas predefinidas y modelos de aprendizaje automático, por lo que no pueden ofrecer la misma precisión y contexto que un hablante nativo. Además, carecen de un enfoque social o de intercambio colaborativo entre usuarios.

Fig. 2.45. Logo de Grammarly

2.3.5. Duolingo

Duolingo es una de las aplicaciones más populares para el aprendizaje de idiomas, conocida por su enfoque gamificado y accesible. A través de ejercicios interactivos, los usuarios pueden mejorar su vocabulario, gramática y comprensión escrita y oral de un idioma. Cuenta con un sistema de recompensas que fomenta la constancia en el estudio. Sin embargo, aunque Duolingo ofrece ejercicios de traducción y escritura, su metodología se centra en frases cortas y respuestas predefinidas, sin permitir la redacción libre ni la corrección detallada de textos extensos.

Fig. 2.46. Logo de Duolingo

2.3.6. Conclusión

A pesar de existir varias opciones similares, ninguna ofrece una solución que combine la escritura libre con la interacción social de manera sencilla y atractiva. La aplicación propuesta en este documento se diferencia en que permite a los usuarios intercambiar textos personales con amigos o nuevos contactos, eligiendo la temática de sus escritos, como si fueran entradas de un diario. Además, incorpora un sistema de almacenamiento de textos, permitiendo a los usuarios llevar un registro de su evolución en la escritura del idioma que están aprendiendo. A diferencia de estas plataformas, la aplicación propuesta busca ofrecer un espacio donde los usuarios puedan expresarse con mayor libertad, intercambiar textos personales con otros estudiantes y recibir correcciones estructuradas en un entorno colaborativo. Uno de los objetivos principales de este proyecto es ofrecer una interfaz intuitiva y visualmente atractiva, diseñada específicamente para facilitar la corrección de textos de manera sencilla y fluida. A diferencia de otras plataformas que presentan sistemas de corrección poco intuitivos o sobrecargados de opciones, esta aplicación prioriza una experiencia de usuario agradable, donde las correcciones pueden realizarse de manera clara y organizada. La combinación de un diseño amigable con la posibilidad de conocer nuevas personas a través del intercambio de textos hace que esta plataforma no solo sea una herramienta educativa, sino también un espacio de interacción y aprendizaje dinámico.

2.4. Justificación de la Elección Tecnológica

Para el desarrollo de esta aplicación se ha optado por Next.js como tecnología de frontend debido a sus ventajas en rendimiento, optimización para SEO y compatibilidad con React. Entre los aspectos más atractivos de esta opción se encuentra que cuenta con una plataforma de lanzamiento y hosting de aplicaciones gratuito, rápido y simple: Vercel; además de que la configuración inicial de un proyecto web se hace muy sencilla y existen numerosas librerías para facilitar el desarrollo web y numerosos recursos y componentes disponibles. En el backend, se ha elegido Node.js con Express por su flexibilidad y la posibilidad de utilizar JavaScript en todo el stack de desarrollo. En cuanto a la base de datos, se ha decidido emplear MongoDB por su modelo flexible y su capacidad para manejar estructuras de datos dinámicas y en formato JSON, el mismo en que se manejan las estructuras de información en Javascript. Finalmente, para el almacenamiento de archivos se utilizará Google Drive API, permitiendo a los usuarios gestionar sus documentos de manera segura y accesible desde cualquier dispositivo. El desarrollo de aplicaciones web ofrece múltiples opciones tecnológicas, cada una con sus propias ventajas y desafíos. La elección de Next.js, Node.js con Express y MongoDB responde a la necesidad de construir una plataforma rápida, escalable y eficiente, capaz de gestionar un alto número de usuarios y permitir la colaboración en la corrección de textos. La integración con Google Drive añade un valor adicional al permitir a los usuarios almacenar y acceder a sus documentos de manera sencilla. Con esta arquitectura, la aplicación propuesta se encuentra

bien posicionada para ofrecer una experiencia fluida y efectiva en el aprendizaje de idiomas mediante la escritura y la corrección colaborativa. En cuanto a la metodología, se utilizará un modelo Kanban dado que es un enfoque ágil, simple y visual, ideal para proyectos en los que solo participa una persona. Permite gestionar el flujo de trabajo de manera rápida y flexible, sin necesidad de invertir demasiado tiempo en planificación detallada o documentación extensa. Su naturaleza visual facilita el seguimiento del progreso y la organización de tareas de forma intuitiva.

3. REQUISITOS

3.1. Requisitos funcionales

ID	Categoría	Requisito Funcional	MVP / Extra
RF001	Gestión de Usuarios	Permitir la creación de una cuenta (username, contraseña, correo).	MVP
RF002	Gestión de Usuarios	Enviar un correo de verificación para vincular y validar la cuenta.	MVP
RF003	Gestión de Usuarios	Permitir el inicio y cierre de sesión.	MVP
RF004	Gestión de Usuarios	Permitir al usuario vincular su almacenamiento con Google Drive.	Extra
RF005	Gestión de Perfiles	Permitir al usuario crear su perfil (idiomas que domina/aprende, biografía).	MVP
RF006	Gestión de Perfiles	Permitir la edición de los datos del perfil (excepto el nombre de usuario y el email).	MVP
RF006	Gestión de Perfiles	Permitir un cambio de la contraseña.	MVP
RF006	Gestión de Perfiles	Permitir la recuperación de la contraseña.	MVP
RF007	Gestión de Perfiles	Permitir la visualización del perfil propio y el de otros usuarios.	MVP
RF008	Gestión de Usuarios	Permitir la eliminación de la cuenta.	MVP
RF009	Gestión de Perfiles	Permitir compartir el perfil a través de un enlace.	MVP
RF010	Gestión de Amistades	Permitir la búsqueda de otros usuarios por nombre de usuario.	MVP
RF011	Gestión de Amistades	Mostrar sugerencias de amistad basadas en los idiomas que domina y aprende.	MVP
RF012	Gestión de Amistades	Permitir enviar solicitudes de amistad.	MVP
RF013	Gestión de Amistades	Permitir recibir y gestionar (aceptar/rechazar) solicitudes de amistad.	MVP
RF014	Gestión de Amistades	Permitir eliminar a un amigo de la lista de sus amigos agregados.	MVP
RF015	Cartas y Diarios	Permitir crear un nuevo diario o carpeta.	MVP
RF016	Cartas y Diarios	Permitir escribir una nueva carta (fecha, título, idioma, contenido).	MVP
RF017	Cartas y Diarios	Permitir editar una carta mientras no haya sido enviada.	MVP
RF018	Cartas y Diarios	Permitir enviar una carta a uno o dos amigos.	MVP
RF019	Cartas y Diarios	Permitir visualizar todas las cartas propias.	MVP

RF020	Cartas y Diarios	Permitir adjuntar fotos, un audio, stickers o GIFs a las cartas.	Extra
RF021	Cartas y Diarios	Permitir conversión de texto en imágenes a carta.	Extra
RF022	Cartas y Diarios	Permitir exportar una carta (corregida o no) a formato PDF.	Extra
RF023	Corrección	Permitir recibir cartas (peticiones de corrección) de amigos.	MVP
RF024	Corrección	Permitir corregir una carta.	MVP
RF025	Corrección	Permitir añadir comentarios finales a la corrección.	MVP
RF026	Corrección	Permitir adjuntar un audio a la corrección (pronunciación).	Extra
RF027	Corrección	Permitir guardar y enviar la corrección al amigo.	MVP
RF028	Corrección	Permitir acceder al historial de todas las correcciones realizadas.	MVP
RF029	Corrección	Permitir recibir y ver las correcciones de cartas enviadas.	MVP
RF030	Corrección	Permitir validar las correcciones recibidas (otorgar 'like' o 'dislike').	Extra
RF031	Gamificación	Permitir consultar estadísticas de cualquier perfil (valoraciones, cartas corregidas).	MVP
RF032	Gamificación	Mostrar un podio con los usuarios de mejores estadísticas.	MVP
RF033	Gamificación	Permitir la creación de retos semanales/mensuales personalizados.	Extra
RF033	Gamificación	Sugerir diferentes temas para la redacción.	Extra
RF035	Gamificación	Asignar logros o insignias por metas alcanzadas.	Extra
RF037	Herramientas	Permitir enviar flashcards (palabras o frases) a un amigo.	Extra
RF038	Herramientas	Permitir al usuario repasar las flashcards recibidas.	Extra

TABLE 3.1. REQUISITOS FUNCIONALES

3.2. Requisitos no funcionales

ID	Categoría	Requisito Funcional
ID	**Categoría**	**Descripción / Métrica**
RNF001	Rendimiento	El 90% de las solicitudes clave deben responder en < 2 **segundos**.
RNF002	Rendimiento	Soportar un mínimo de **100 usuarios concurrentes** activos.
RNF003	Rendimiento	La arquitectura debe permitir **escalado horizontal** para el crecimiento.
RNF004	Usabilidad	Mantener una **apariencia y navegación coherentes** (React UI).
RNF005	Usabilidad	Ser **completamente funcional** en dispositivos de escritorio y móviles.
RNF006	Usabilidad	El proceso de corrección debe ser **visual y funcionalmente intuitivo**.
RNF007	Seguridad	Usar **JWT** y/o **Google Sign-In** para autenticación.
RNF008	Seguridad	Todas las comunicaciones deben usar **HTTPS**.
RNF009	Seguridad	Implementar validaciones en **MongDB** para la integridad de datos.
RNF010	Seguridad	Implementar medidas para prevenir ataques como **CSRF y XSS**.
RNF011	Tecnología	Usar **React, Node.js/Express, MongoDB, Google Cloud API y Google Sign-In**.
RNF012	Mantenibilidad	Seguir **patrones de diseño** y guías de estilo para el mantenimiento.
RNF013	Disponibilidad	Garantizar una disponibilidad del **99.5%** del tiempo.
RNF014	Disponibilidad	Proporcionar **mensajes de error claros** y registrar logs internos.

TABLE 3.2. REQUISITOS NO FUNCIONALES

BIBLIOGRAPHY

[1] Parlamento Europeo. “Reglamento (ue) 2016/679 del parlamento europeo y del consejo: Reglamento general de protección de datos (rgpd).” Accedido: 2026-02-27. [Online]. Available: https://eur-lex.europa.eu/legal-content/ES/TXT/PDF/?uri=CELEX:32016R0679

[2] Par amento Español. “Ley orgánica 3/2018, de 5 de diciembre, de protección de datos personales y garantía de los derechos digitales (lopdgdd).” Accedido: 2026-02-27. [Online]. Available: https://www.boe.es/buscar/act.php?id=BOE-A-2018-16673

[3] Parlamento Español. “Ley 34/1988, de 11 de noviembre, sobre la sociedad de la información y de comercio electrónico (lssice).” Accedido: 2026-02-27. [Online]. Available: https://www.boe.es/buscar/act.php?id=BOE-A-1988-24729

[4] Parlamento Español. “Texto refundido de la ley de propiedad intelectual.” Accedido: 2026-02-27. [Online]. Available: https://www.boe.es/buscar/act.php?id=BOE-A-1996-8930

[5] Microsoft. “Visual studio code.” Accedido: 2026-02-27. [Online]. Available: https://code.visualstudio.com/

[6] Mozilla Foundation. “Firefox developer edition.” Accedido: 2026-02-27. [Online]. Available: https://www.mozilla.org/es/firefox/developer/

[7] MongoDB Inc. “Mongodb compass.” Accedido: 2026-02-27. [Online]. Available: https://www.mongodb.com/products/compass

[8] Postman Inc. “Postman.” Accedido: 2026-02-27. [Online]. Available: https://www.postman.com/

[9] GitHub Inc. “Github.” Accedido: 2026-02-27. [Online]. Available: https://github.com/