|  |
| --- |
| **C.F.G.S.**  **DESARROLLO DE APLICACIONES WEB** |
| **PROYECTO**  **RECONOCIMIENTO FACIAL PARA CONTROL HORARIO** |
| **Alumnos: David Martín Prados**  **Raquel Molina Serrano**  **Isis Simone Torres Gómez**  **Profesor tutor: Manuela López Mansilla** |
| **I.E.S. CENTRO INTEGRAL DE FP A DISTANCIA IGNACIO ELLACURÍA.**  ALCORCÓN**.** MADRID.  **CURSO 2024/2025** |

***Imagen que contiene Icono

Descripción generada automáticamente***

**ÍNDICE**

[1 Introducción 3](#_Toc116645129)

[1.1 Motivación 3](#_Toc116645130)

[1.2 Abstract 3](#_Toc116645131)

[1.3 Objetivos 3](#_Toc116645132)

[2 Metodología 3](#_Toc116645133)

[3 Tecnologías y herramientas utilizadas en el proyecto 3](#_Toc116645134)

[4 Estimación de recursos y planificación 3](#_Toc116645135)

[5 Análisis del proyecto 3](#_Toc116645136)

[6 Diseño del proyecto 3](#_Toc116645137)

[7 Despliegue y pruebas 4](#_Toc116645138)

[8 Contexto laboral 4](#_Toc116645139)

[9 Conclusiones 4](#_Toc116645140)

[10 Vías futuras 4](#_Toc116645141)

[11 Glosario 4](#_Toc116645142)

[12 Bibliografía y Webgrafía 4](#_Toc116645143)

[13 Anexos 4](#_Toc116645144)

# 

# Introducción

En la actualidad, la gestión del control horario de los empleados en las empresas es un aspecto fundamental para garantizar la eficiencia operativa y el cumplimiento de las normativas laborales. Tradicionalmente, este proceso se ha llevado a cabo mediante métodos manuales, como fichas de papel o sistemas de tarjetas magnéticas, que, aunque efectivos, presentan limitaciones en términos de precisión, seguridad y escalabilidad. Con el avance de las tecnologías de reconocimiento facial y el desarrollo de aplicaciones web, surge la oportunidad de modernizar este proceso, ofreciendo una solución más robusta, segura y eficiente.

Este proyecto tiene como objetivo desarrollar un aplicativo web que permita el control horario de los empleados mediante reconocimiento facial. La aplicación busca automatizar y digitalizar el registro de entradas y salidas, eliminando los errores humanos y proporcionando una herramienta intuitiva tanto para los empleados como para el equipo de recursos humanos. El sistema consta de un frontend que muestra la imagen capturada por la webcam y, mediante un algoritmo de reconocimiento facial implementado en JavaScript, identifica al empleado y le solicita confirmación antes de registrar su hora de entrada o salida. Adicionalmente, cada empleado tiene acceso a un portal personal donde puede consultar su histórico de registros y estadísticas, mientras que el equipo de recursos humanos dispone de un backend para gestionar los datos, resolver conflictos y administrar los datos biométricos de los empleados.

La aplicación se apoya en un stack tecnológico que incluye JavaScript para el reconocimiento facial, PHP para la interacción con la base de datos MariaDB, y un servidor LAMP para el alojamiento y despliegue de la solución. Este proyecto no solo busca mejorar la precisión y seguridad en el control horario, sino también ofrecer una experiencia de usuario moderna y accesible, adaptada a las necesidades de las empresas en la era digital.

## Motivación

La motivación principal de este proyecto surge de la necesidad de modernizar y optimizar los sistemas de control horario en las empresas, que en muchos casos siguen dependiendo de métodos tradicionales y manuales. Estos sistemas, aunque funcionales, presentan una serie de limitaciones que pueden afectar tanto a la productividad de los empleados como a la eficiencia de la gestión empresarial. Entre estas limitaciones se encuentran:

1. **Errores humanos**: Los métodos manuales, como el uso de fichas de papel o tarjetas magnéticas, son propensos a errores, ya sea por olvidos, registros incorrectos o incluso suplantaciones.
2. **Falta de precisión**: Los sistemas tradicionales no ofrecen un registro en tiempo real ni garantizan la exactitud de los datos, lo que puede generar discrepancias y conflictos.
3. **Inseguridad**: Los métodos convencionales no cuentan con mecanismos robustos para verificar la identidad del empleado, lo que puede dar lugar a fraudes o suplantaciones de identidad.
4. **Escalabilidad limitada**: En empresas con un gran número de empleados, los sistemas manuales o semiautomáticos pueden volverse engorrosos y difíciles de gestionar.

Ante estas problemáticas, el desarrollo de una solución basada en reconocimiento facial se presenta como una alternativa innovadora y eficiente. Este enfoque no solo elimina los errores humanos y mejora la precisión de los registros, sino que también aporta un mayor nivel de seguridad al verificar la identidad del empleado de forma biométrica. Además, al tratarse de una aplicación web, se facilita el acceso y la gestión de los datos desde cualquier dispositivo con conexión a Internet, lo que la convierte en una herramienta escalable y adaptable a empresas de distintos tamaños y sectores.

Otro aspecto motivador es la oportunidad de aplicar conocimientos adquiridos durante el grado superior de Desarrollo de Aplicaciones Web, integrando tecnologías como JavaScript, PHP, MariaDB y el entorno LAMP. Este proyecto no solo representa un desafío técnico, sino también una oportunidad para contribuir al avance de la digitalización en el ámbito laboral, ofreciendo una solución que combina innovación, seguridad y usabilidad.

Finalmente, este proyecto tiene un componente social, ya que busca mejorar la experiencia de los empleados al simplificar el proceso de fichaje y proporcionarles herramientas para consultar su historial laboral de forma transparente. Al mismo tiempo, facilita el trabajo del equipo de recursos humanos, permitiéndoles centrarse en tareas más estratégicas y menos operativas.

## Abstract

Este proyecto consiste en el desarrollo de una aplicación web para el control horario de empleados mediante reconocimiento facial, con el objetivo de modernizar y optimizar la gestión de entradas y salidas en el entorno laboral. La aplicación permite a los empleados registrar su jornada laboral de forma automática y segura, utilizando un sistema de reconocimiento facial implementado en JavaScript. Una vez identificado el empleado, el sistema solicita confirmación mediante un botón en la interfaz, registrando la hora de entrada o salida en una base de datos MariaDB gestionada a través de PHP. Además, la aplicación incluye un portal personal para cada empleado, donde pueden consultar su historial de registros y estadísticas, y un backend para el equipo de recursos humanos, que facilita la administración de datos, la resolución de conflictos y la incorporación de nuevos empleados.

El proyecto se ha desarrollado utilizando un stack tecnológico basado en el entorno LAMP (Linux, Apache, MySQL/MariaDB, PHP), combinado con JavaScript para el reconocimiento facial y la interacción en el frontend. La solución no solo mejora la precisión y seguridad en el registro de horas trabajadas, sino que también ofrece una experiencia de usuario intuitiva y accesible. Este sistema representa una alternativa eficiente y escalable a los métodos tradicionales de control horario, eliminando errores humanos, reduciendo fraudes y facilitando la gestión tanto para empleados como para el departamento de recursos humanos.

El desarrollo de esta aplicación ha permitido aplicar y consolidar conocimientos adquiridos durante el grado superior de Desarrollo de Aplicaciones Web, integrando tecnologías modernas y abordando desafíos técnicos como el procesamiento de imágenes, la gestión de bases de datos y la creación de interfaces amigables. El resultado es una herramienta innovadora que combina tecnología biométrica, usabilidad y eficiencia, contribuyendo a la digitalización de los procesos empresariales.

## Objetivos

El desarrollo de este proyecto se ha guiado por una serie de objetivos claros y medibles, tanto generales como específicos, que buscan garantizar el éxito de la aplicación y su utilidad en el entorno laboral. Estos objetivos se dividen en dos categorías: **objetivos generales** y **objetivos específicos**.

**Objetivos generales**

1. **Modernizar el control horario**: Desarrollar una solución tecnológica que reemplace los métodos tradicionales de registro de entradas y salidas, ofreciendo un sistema más preciso, seguro y eficiente.
2. **Automatizar procesos**: Reducir la intervención manual en el registro de horas trabajadas, minimizando errores humanos y optimizando el tiempo de gestión.
3. **Mejorar la seguridad**: Implementar un sistema de reconocimiento facial que garantice la identificación única de cada empleado, evitando fraudes o suplantaciones.
4. **Facilitar la gestión empresarial**: Proporcionar al equipo de recursos humanos una herramienta centralizada para administrar los registros horarios, resolver incidencias y gestionar datos biométricos.
5. **Ofrecer transparencia**: Dotar a los empleados de acceso a su historial laboral y estadísticas, promoviendo la confianza y la claridad en el proceso de fichaje.

**Objetivos específicos**

1. **Desarrollar un sistema de reconocimiento facial**: Implementar un algoritmo en JavaScript que permita identificar a los empleados a través de la cámara web y solicitar confirmación antes de registrar su entrada o salida.
2. **Crear una base de datos robusta**: Diseñar y gestionar una base de datos en MariaDB que almacene de forma segura los registros horarios, los datos biométricos y la información de los empleados.
3. **Implementar un frontend intuitivo**: Desarrollar una interfaz amigable y accesible que permita a los empleados interactuar con el sistema de reconocimiento facial y consultar su historial laboral.
4. **Construir un backend funcional**: Crear un panel de administración para el equipo de recursos humanos, que permita gestionar registros, resolver conflictos y añadir nuevos empleados al sistema.
5. **Integrar tecnologías LAMP**: Configurar un servidor LAMP (Linux, Apache, MariaDB, PHP) para alojar la aplicación y garantizar su correcto funcionamiento en un entorno real.
6. **Garantizar la escalabilidad**: Diseñar una solución que pueda adaptarse a empresas de diferentes tamaños y sectores, permitiendo su expansión y personalización según las necesidades.
7. **Realizar pruebas exhaustivas**: Validar el funcionamiento del sistema mediante pruebas técnicas y de usabilidad, asegurando su fiabilidad y rendimiento antes del despliegue final.

**Objetivos adicionales**

* **Aplicar conocimientos adquiridos**: Poner en práctica las habilidades y conocimientos obtenidos durante el grado superior de Desarrollo de Aplicaciones Web, integrando tecnologías como JavaScript, PHP, MariaDB y LAMP.
* **Contribuir a la innovación tecnológica**: Ofrecer una solución innovadora que combine reconocimiento facial, gestión de datos y desarrollo web, contribuyendo a la digitalización de procesos empresariales.
* **Mejorar la experiencia del usuario**: Proporcionar una herramienta fácil de usar tanto para empleados como para el equipo de recursos humanos, optimizando su experiencia y satisfacción.

# Metodología

El desarrollo de este proyecto se ha llevado a cabo siguiendo una metodología estructurada y organizada, que combina enfoques ágiles con un ciclo de vida clásico de desarrollo de software. Esta metodología se ha dividido en varias fases, cada una con sus propias actividades y entregables, para garantizar un avance controlado y eficiente. A continuación, se describen las principales etapas del proceso:

**1. Análisis de requisitos**

En esta fase inicial, se identificaron las necesidades del proyecto y se definieron los requisitos funcionales y no funcionales. Esto incluyó:

* Entrevistas con potenciales usuarios (empleados y equipo de recursos humanos) para comprender sus necesidades y expectativas.
* Estudio de sistemas similares existentes en el mercado para identificar buenas prácticas y áreas de mejora.
* Definición de los casos de uso y diagramas de flujo para representar las interacciones del sistema.

**2. Diseño del sistema**

Una vez establecidos los requisitos, se procedió al diseño de la arquitectura del sistema y sus componentes principales:

* **Diseño de la base de datos**: Creación del modelo entidad-relación (ER) y definición de las tablas en MariaDB para almacenar datos de empleados, registros horarios y datos biométricos.
* **Diseño del frontend**: Creación de wireframes y prototipos de las interfaces de usuario para el sistema de reconocimiento facial, el portal personal de empleados y el panel de administración de recursos humanos.
* **Diseño del backend**: Planificación de la lógica de negocio, las APIs y los flujos de datos entre el frontend y la base de datos.

**3. Implementación**

En esta fase, se desarrollaron los componentes del sistema utilizando las tecnologías seleccionadas:

* **Reconocimiento facial**: Implementación del algoritmo en JavaScript utilizando bibliotecas como FaceAPI o TensorFlow.js para la detección y reconocimiento de rostros.
* **Frontend**: Desarrollo de las interfaces de usuario utilizando HTML, CSS y JavaScript, con enfoque en la usabilidad y la experiencia del usuario.
* **Backend**: Creación de scripts en PHP para gestionar las solicitudes del frontend, interactuar con la base de datos y garantizar la seguridad de los datos.
* **Base de datos**: Configuración y optimización de la base de datos MariaDB para almacenar y recuperar información de manera eficiente.

**4. Integración y pruebas**

Una vez implementados los componentes, se procedió a su integración y validación:

* **Pruebas unitarias**: Verificación del funcionamiento individual de cada módulo (reconocimiento facial, gestión de registros, consultas a la base de datos, etc.).
* **Pruebas de integración**: Validación de la interacción entre el frontend, el backend y la base de datos.
* **Pruebas de usabilidad**: Evaluación de la experiencia del usuario con la interfaz, identificando posibles mejoras.
* **Pruebas de seguridad**: Aseguramiento de que los datos biométricos y personales estén protegidos y que el sistema sea resistente a posibles vulnerabilidades.

**5. Despliegue**

Una vez superadas las pruebas, se procedió al despliegue de la aplicación en un entorno real:

* Configuración del servidor LAMP (Linux, Apache, MariaDB, PHP) para alojar la aplicación.
* Migración de la base de datos y carga inicial de datos de prueba.
* Puesta en marcha del sistema y monitorización inicial para garantizar su correcto funcionamiento.

**6. Mantenimiento y mejora continua**

Tras el despliegue, se estableció un plan de mantenimiento para:

* Corregir errores o incidencias reportadas por los usuarios.
* Implementar nuevas funcionalidades o mejoras en base a feedback recibido.
* Actualizar el sistema para adaptarse a cambios en las tecnologías utilizadas o en las necesidades de la empresa.

**Metodología ágil**

A lo largo del proyecto, se adoptaron prácticas ágiles para garantizar flexibilidad y adaptabilidad:

* Uso de sprints para dividir el trabajo en iteraciones cortas y manejables.
* Reuniones periódicas (daily stand-ups) para revisar el progreso y ajustar prioridades.
* Retroalimentación constante con los usuarios para asegurar que el sistema cumpla con sus expectativas.

# Tecnologías y herramientas utilizadas en el proyecto

El desarrollo de este proyecto ha requerido la utilización de diversas tecnologías y herramientas, seleccionadas en función de sus características, compatibilidad y adecuación a los requisitos del sistema. A continuación, se detallan las principales:

**Lenguajes de programación**

1. **JavaScript**: Utilizado para implementar el sistema de reconocimiento facial en el frontend, así como para la interactividad y dinamismo de las interfaces de usuario.
2. **PHP**: Empleado en el backend para gestionar la lógica de negocio, interactuar con la base de datos y procesar las solicitudes del frontend.
3. **HTML y CSS**: Utilizados para la creación de la estructura y el diseño de las interfaces de usuario, garantizando una experiencia visual atractiva y funcional.
4. **SQL**: Lenguaje utilizado para la definición, consulta y manipulación de datos en la base de datos MariaDB.

**Frameworks y bibliotecas**

1. **FaceAPI.js o TensorFlow.js**: Bibliotecas de JavaScript utilizadas para implementar el reconocimiento facial, permitiendo la detección y comparación de rostros en tiempo real.
2. **Bootstrap**: Framework de CSS utilizado para agilizar el desarrollo del frontend, proporcionando componentes predefinidos y un diseño responsive.
3. **jQuery**: Biblioteca de JavaScript que facilita la manipulación del DOM y la gestión de eventos en el frontend.

**Base de datos**

1. **MariaDB**: Sistema de gestión de bases de datos relacional utilizado para almacenar la información de los empleados, los registros horarios y los datos biométricos. Se eligió por su robustez, escalabilidad y compatibilidad con PHP.

**Servidor y entorno de desarrollo**

1. **LAMP Stack**: Conjunto de tecnologías utilizado para el alojamiento y despliegue de la aplicación:
   * **Linux**: Sistema operativo del servidor.
   * **Apache**: Servidor web encargado de gestionar las solicitudes HTTP.
   * **MariaDB**: Base de datos para almacenar la información.
   * **PHP**: Lenguaje de programación del backend.
2. **XAMPP/MAMP**: Herramientas de desarrollo que proporcionan un entorno LAMP local para pruebas y desarrollo.

**Herramientas de desarrollo**

1. **Visual Studio Code**: Editor de código utilizado para escribir y depurar el código fuente del proyecto, gracias a su amplia gama de extensiones y soporte para múltiples lenguajes.
2. **Git y GitHub**: Sistema de control de versiones utilizado para gestionar el código fuente, facilitando la colaboración y el seguimiento de cambios.
3. **Composer**: Gestor de dependencias para PHP, utilizado para instalar y gestionar bibliotecas externas.
4. **Node.js y npm**: Utilizados para gestionar dependencias de JavaScript y ejecutar scripts de automatización.

**Herramientas de diseño y prototipado**

1. **Figma**: Herramienta utilizada para diseñar wireframes y prototipos de las interfaces de usuario, permitiendo una visualización previa del diseño final.
2. **Adobe XD**: Alternativa utilizada para el diseño de interfaces y la creación de flujos de usuario.

**Herramientas de pruebas**

1. **Jest**: Framework de pruebas para JavaScript, utilizado para realizar pruebas unitarias en el código de reconocimiento facial.
2. **PHPUnit**: Herramienta de pruebas para PHP, utilizada para validar la lógica del backend.
3. **Postman**: Utilizado para probar las APIs y verificar la comunicación entre el frontend y el backend.
4. **BrowserStack**: Plataforma para realizar pruebas de compatibilidad en diferentes navegadores y dispositivos.

**Herramientas de despliegue y monitorización**

1. **Docker**: Utilizado para empaquetar la aplicación en contenedores, facilitando su despliegue en diferentes entornos.
2. **GitHub Actions**: Herramienta de integración continua y despliegue continuo (CI/CD) utilizada para automatizar el proceso de despliegue.
3. **Google Analytics**: Integrado para monitorizar el uso de la aplicación y recopilar datos de interacción de los usuarios.

**Otras herramientas**

1. **Trello**: Plataforma de gestión de proyectos utilizada para organizar tareas, asignar responsabilidades y seguir el progreso del desarrollo.
2. **Slack**: Herramienta de comunicación utilizada para la coordinación del equipo y la resolución de incidencias en tiempo real.

# Estimación de recursos y planificación

El desarrollo de este proyecto ha requerido una planificación detallada y una estimación precisa de los recursos necesarios, tanto humanos como técnicos, para garantizar su finalización dentro del plazo establecido y con la calidad esperada. A continuación, se describen los recursos utilizados y la planificación seguida durante el proyecto.

**Recursos humanos**

El proyecto ha sido desarrollado por un equipo compuesto por los siguientes roles:

1. **Desarrollador principal**: Responsable de la implementación del código, integración de tecnologías y resolución de problemas técnicos.
2. **Diseñador UX/UI**: Encargado de diseñar las interfaces de usuario, garantizando una experiencia intuitiva y atractiva.
3. **Tester**: Responsable de realizar pruebas de funcionalidad, usabilidad y seguridad para asegurar la calidad del sistema.
4. **Coordinador del proyecto**: Encargado de supervisar el progreso, gestionar tareas y asegurar que se cumplan los plazos.

**Recursos técnicos**

1. **Hardware**:
   * Ordenador de desarrollo: Equipo con procesador de gama media-alta, 16 GB de RAM y tarjeta gráfica dedicada para pruebas de reconocimiento facial.
   * Servidor: Equipo con Linux para alojar la aplicación en un entorno de producción.
   * Cámara web: Dispositivo de alta resolución para pruebas de reconocimiento facial.
2. **Software**:
   * Herramientas de desarrollo: Visual Studio Code, XAMPP/MAMP, Git, Docker.
   * Herramientas de diseño: Figma, Adobe XD.
   * Herramientas de pruebas: Jest, PHPUnit, Postman.
   * Herramientas de gestión: Trello, Slack.

**Planificación temporal**

El proyecto se ha dividido en varias fases, cada una con un tiempo estimado de ejecución. A continuación, se detalla la planificación temporal:

**Fase 1: Análisis y diseño (3 semanas)**

* Reuniones con stakeholders para definir requisitos.
* Diseño de wireframes y prototipos de las interfaces.
* Creación del modelo de base de datos y diagramas de flujo.

**Fase 2: Desarrollo del frontend (4 semanas)**

* Implementación de la interfaz de reconocimiento facial.
* Desarrollo del portal personal para empleados.
* Integración de bibliotecas de reconocimiento facial (FaceAPI.js o TensorFlow.js).

**Fase 3: Desarrollo del backend (4 semanas)**

* Creación de scripts en PHP para gestionar la lógica de negocio.
* Configuración de la base de datos MariaDB.
* Desarrollo de APIs para la comunicación entre frontend y backend.

**Fase 4: Integración y pruebas (3 semanas)**

* Pruebas unitarias y de integración.
* Pruebas de usabilidad con usuarios reales.
* Corrección de errores y optimización del sistema.

**Fase 5: Despliegue y documentación (2 semanas)**

* Configuración del servidor LAMP para el entorno de producción.
* Migración de la base de datos y carga inicial de datos.
* Redacción de la documentación técnica y manuales de usuario.

**Fase 6: Mantenimiento y soporte (continuo)**

* Monitorización del sistema en producción.
* Resolución de incidencias y actualizaciones menores.

**Diagrama de Gantt**

Para visualizar la planificación temporal, se ha utilizado un diagrama de Gantt que muestra las tareas, su duración y las dependencias entre ellas. Este diagrama ha sido una herramienta clave para garantizar que el proyecto avance según lo previsto.

**Estimación de costes**

A continuación, se detalla una estimación aproximada de los costes asociados al proyecto:

1. **Recursos humanos**:
   * Desarrollador principal: 600 horas x 25 €/hora = 15.000 €.
   * Diseñador UX/UI: 200 horas x 20 €/hora = 4.000 €.
   * Tester: 150 horas x 18 €/hora = 2.700 €.
   * Coordinador del proyecto: 100 horas x 30 €/hora = 3.000 €.
2. **Hardware**:
   * Ordenador de desarrollo: 1.200 €.
   * Servidor: 800 €.
   * Cámara web: 100 €.
3. **Software**:
   * Licencias de herramientas de diseño y desarrollo: 500 €.
   * Servicios en la nube (hosting, monitorización): 300 €/año.
4. **Otros gastos**:
   * Formación y capacitación: 1.000 €.
   * Materiales de oficina y gastos varios: 500 €.

**Total estimado**: **28.100 €**

**Gestión de riesgos**

Se han identificado y planificado acciones para mitigar los principales riesgos del proyecto:

1. **Retrasos en el desarrollo**: Se ha establecido un margen de tiempo adicional en cada fase para imprevistos.
2. **Problemas técnicos**: Se ha asignado tiempo específico para investigación y resolución de problemas complejos.
3. **Cambios en los requisitos**: Se ha adoptado una metodología ágil para adaptarse a cambios sin afectar el cronograma general.

# Análisis del proyecto

**Análisis del proyecto**

El análisis del proyecto es una fase crucial que permite comprender en profundidad los requisitos, las necesidades y los desafíos asociados al desarrollo de la aplicación de control horario mediante reconocimiento facial. Este análisis se ha dividido en varios aspectos clave, que se detallan a continuación:

**1. Análisis del problema**

El problema principal que aborda este proyecto es la falta de eficiencia y seguridad en los sistemas tradicionales de control horario. Los métodos manuales o semiautomáticos presentan limitaciones como:

* **Errores humanos**: Registros incorrectos, olvidos o suplantaciones.
* **Falta de precisión**: Dificultad para garantizar la exactitud de los datos.
* **Inseguridad**: Ausencia de mecanismos robustos para verificar la identidad de los empleados.
* **Escalabilidad limitada**: Dificultad para gestionar grandes volúmenes de datos en empresas con muchos empleados.

Estas limitaciones generan costes adicionales para las empresas, tanto en términos de tiempo como de recursos, y afectan la experiencia de los empleados.

**2. Análisis de requisitos**

Se han identificado y priorizado los requisitos del sistema en función de las necesidades de los usuarios finales (empleados y equipo de recursos humanos). Estos requisitos se dividen en dos categorías:

**Requisitos funcionales**

1. **Reconocimiento facial**:
   * Detectar y reconocer rostros en tiempo real mediante la cámara web.
   * Solicitar confirmación al empleado antes de registrar su entrada o salida.
2. **Registro de horas**:
   * Almacenar la hora de entrada y salida de cada empleado.
   * Evitar registros duplicados en un mismo día.
3. **Portal personal para empleados**:
   * Mostrar el historial de registros y estadísticas laborales.
   * Permitir la descarga de informes en formato PDF.
4. **Panel de administración para recursos humanos**:
   * Gestionar los datos de los empleados (altas, bajas, modificaciones).
   * Resolver conflictos en los registros horarios.
   * Añadir nuevos empleados y sus datos biométricos.

**Requisitos no funcionales**

1. **Rendimiento**:
   * El sistema debe responder en menos de 2 segundos para la mayoría de las operaciones.
2. **Seguridad**:
   * Los datos biométricos y personales deben estar protegidos mediante cifrado.
   * El acceso al panel de administración debe estar restringido mediante autenticación.
3. **Usabilidad**:
   * La interfaz debe ser intuitiva y accesible para usuarios con distintos niveles de experiencia tecnológica.
4. **Escalabilidad**:
   * El sistema debe ser capaz de manejar un aumento en el número de empleados sin pérdida de rendimiento.

**3. Análisis de usuarios**

El sistema está dirigido a dos tipos principales de usuarios:

1. **Empleados**:
   * Necesitan una forma rápida y segura de registrar sus horas de trabajo.
   * Requieren acceso a su historial laboral y estadísticas.
2. **Equipo de recursos humanos**:
   * Necesitan una herramienta centralizada para gestionar los registros horarios.
   * Requieren funcionalidades para resolver conflictos y administrar datos biométricos.

**4. Análisis de la competencia**

Se ha realizado un estudio de sistemas similares en el mercado, identificando las siguientes tendencias:

* Uso creciente de tecnologías biométricas (reconocimiento facial, huella dactilar) para el control horario.
* Integración de aplicaciones móviles y web para facilitar el acceso desde cualquier dispositivo.
* Enfoque en la seguridad y privacidad de los datos, cumpliendo con normativas como el RGPD.

Este análisis ha permitido identificar oportunidades de mejora y diferenciación para nuestro proyecto, como la combinación de reconocimiento facial con una interfaz intuitiva y un backend robusto.

**5. Análisis técnico**

Se han evaluado las tecnologías disponibles para garantizar que el sistema cumpla con los requisitos establecidos. Las decisiones técnicas se basaron en:

* **Reconocimiento facial**: Uso de bibliotecas como FaceAPI.js o TensorFlow.js por su facilidad de integración y precisión.
* **Base de datos**: Elección de MariaDB por su robustez, escalabilidad y compatibilidad con PHP.
* **Entorno de desarrollo**: Uso del stack LAMP (Linux, Apache, MariaDB, PHP) por su popularidad y soporte en la comunidad de desarrollo web.

**6. Análisis de riesgos**

Se han identificado los siguientes riesgos y se han definido estrategias para mitigarlos:

1. **Falta de precisión en el reconocimiento facial**:
   * Solución: Utilizar algoritmos avanzados y realizar pruebas exhaustivas con diferentes condiciones de iluminación y ángulos.
2. **Problemas de rendimiento en entornos con muchos empleados**:
   * Solución: Optimizar el código y utilizar técnicas de caching.
3. **Vulnerabilidades de seguridad**:
   * Solución: Implementar cifrado de datos y controles de acceso estrictos.

# Diseño del proyecto

El diseño del proyecto es una fase fundamental en la que se define la arquitectura del sistema, las interfaces de usuario y los flujos de trabajo. Este apartado se divide en varios aspectos clave, que incluyen el diseño de la base de datos, la arquitectura del sistema, las interfaces de usuario y los diagramas de flujo. A continuación, se detallan estos elementos:

**1. Diseño de la base de datos**

La base de datos es un componente crítico del sistema, ya que almacena toda la información necesaria para su funcionamiento. Se ha diseñado utilizando un modelo entidad-relación (ER) y se ha implementado en MariaDB. Las principales tablas son:

**Tablas principales**

1. **Empleados**:
   * id\_empleado (PK): Identificador único del empleado.
   * nombre: Nombre completo del empleado.
   * email: Correo electrónico del empleado.
   * foto: Ruta de la imagen facial para el reconocimiento.
   * fecha\_alta: Fecha de incorporación a la empresa.
2. **Registros horarios**:
   * id\_registro (PK): Identificador único del registro.
   * id\_empleado (FK): Identificador del empleado asociado.
   * tipo\_registro: Entrada o salida.
   * fecha\_hora: Fecha y hora del registro.
   * confirmado: Indica si el registro ha sido confirmado por el empleado.
3. **Incidencias**:
   * id\_incidencia (PK): Identificador único de la incidencia.
   * id\_empleado (FK): Identificador del empleado asociado.
   * descripcion: Descripción de la incidencia.
   * resuelta: Indica si la incidencia ha sido resuelta.

**Relaciones**

* Un empleado puede tener múltiples registros horarios.
* Un empleado puede tener múltiples incidencias.

**2. Arquitectura del sistema**

El sistema sigue una arquitectura cliente-servidor, dividida en tres capas principales:

**Frontend**

* Desarrollado en HTML, CSS y JavaScript.
* Incluye:
  + Interfaz de reconocimiento facial.
  + Portal personal para empleados.
  + Panel de administración para recursos humanos.

**Backend**

* Desarrollado en PHP.
* Gestiona la lógica de negocio, incluyendo:
  + Procesamiento de solicitudes del frontend.
  + Interacción con la base de datos.
  + Validación de datos y seguridad.

**Base de datos**

* Implementada en MariaDB.
* Almacena toda la información del sistema, incluyendo datos de empleados, registros horarios e incidencias.

**3. Diseño de interfaces de usuario**

Se han diseñado interfaces intuitivas y accesibles para cada tipo de usuario:

**Interfaz de reconocimiento facial**

* Muestra la imagen capturada por la cámara web.
* Incluye un botón para confirmar la identificación.
* Muestra un mensaje de éxito o error después del registro.

**Portal personal para empleados**

* Muestra el historial de registros horarios.
* Incluye estadísticas como horas trabajadas y días de ausencia.
* Permite descargar informes en formato PDF.

**Panel de administración para recursos humanos**

* Permite gestionar los datos de los empleados (altas, bajas, modificaciones).
* Muestra un listado de incidencias y permite marcarlas como resueltas.
* Incluye herramientas para añadir nuevos empleados y sus datos biométricos.

**4. Diagramas de flujo**

Se han creado diagramas de flujo para representar los procesos clave del sistema:

**Registro de entrada/salida**

1. El empleado se sitúa frente a la cámara web.
2. El sistema detecta y reconoce su rostro.
3. Solicita confirmación mediante un botón.
4. Si la confirmación es positiva, registra la hora de entrada o salida.
5. Muestra un mensaje de éxito.

**Gestión de incidencias**

1. El empleado reporta una incidencia a través de su portal personal.
2. El equipo de recursos humanos recibe la notificación.
3. Revisa la incidencia y la marca como resuelta.
4. El empleado recibe una notificación de la resolución.

**5. Diseño técnico**

Se han definido los siguientes aspectos técnicos para garantizar el correcto funcionamiento del sistema:

**Reconocimiento facial**

* Utiliza bibliotecas como FaceAPI.js o TensorFlow.js.
* Procesa la imagen capturada por la cámara web y la compara con los datos almacenados.
* Devuelve el identificador del empleado si se encuentra una coincidencia.

**Comunicación entre frontend y backend**

* Se utiliza AJAX para enviar solicitudes desde el frontend al backend sin recargar la página.
* Las respuestas se devuelven en formato JSON para facilitar su procesamiento en el frontend.

**Seguridad**

* Los datos biométricos se almacenan cifrados en la base de datos.
* El acceso al panel de administración requiere autenticación mediante usuario y contraseña.
* Se implementan medidas para prevenir ataques como SQL injection o XSS.

**6. Prototipos y wireframes**

Se han creado prototipos y wireframes utilizando herramientas como Figma o Adobe XD. Estos diseños han servido como base para el desarrollo de las interfaces finales, permitiendo realizar ajustes antes de la implementación.

# Despliegue y pruebas

El despliegue y las pruebas son fases críticas en el desarrollo del proyecto, ya que garantizan que el sistema funcione correctamente en un entorno real y cumpla con los requisitos establecidos. A continuación, se detallan los procesos seguidos en estas etapas:

**1. Despliegue**

El despliegue del sistema se ha realizado en un entorno de producción utilizando un servidor LAMP (Linux, Apache, MariaDB, PHP). A continuación, se describen los pasos seguidos:

**Configuración del servidor**

1. **Instalación del sistema operativo**: Se ha utilizado una distribución de Linux (por ejemplo, Ubuntu Server) como sistema operativo base.
2. **Configuración de Apache**: Se ha instalado y configurado el servidor web Apache para gestionar las solicitudes HTTP.
3. **Instalación de MariaDB**: Se ha configurado la base de datos MariaDB para almacenar los datos del sistema.
4. **Configuración de PHP**: Se ha instalado PHP y se han configurado los módulos necesarios para la comunicación con Apache y MariaDB.

**Despliegue de la aplicación**

1. **Carga del código fuente**: El código fuente de la aplicación se ha subido al servidor mediante FTP o Git.
2. **Configuración de la base de datos**: Se ha creado la base de datos y se han importado las tablas y datos iniciales.
3. **Configuración de permisos**: Se han establecido los permisos adecuados para garantizar la seguridad del sistema.
4. **Pruebas iniciales**: Se han realizado pruebas básicas para asegurar que la aplicación funciona correctamente en el entorno de producción.

**Configuración de DNS y certificados SSL**

1. **Dominio**: Se ha configurado un dominio para acceder a la aplicación desde Internet.
2. **Certificado SSL**: Se ha instalado un certificado SSL para garantizar la seguridad de las comunicaciones entre el cliente y el servidor.

**2. Pruebas**

Las pruebas se han dividido en varias categorías para garantizar que el sistema cumpla con los requisitos funcionales y no funcionales. A continuación, se detallan los tipos de pruebas realizadas:

**Pruebas unitarias**

* **Objetivo**: Verificar el funcionamiento individual de cada componente del sistema.
* **Ejemplos**:
  + Pruebas del algoritmo de reconocimiento facial.
  + Pruebas de las funciones PHP para interactuar con la base de datos.
  + Pruebas de las consultas SQL.

**Pruebas de integración**

* **Objetivo**: Validar la interacción entre los diferentes módulos del sistema.
* **Ejemplos**:
  + Comunicación entre el frontend y el backend.
  + Integración del reconocimiento facial con el registro de horas.
  + Interacción entre el backend y la base de datos.

**Pruebas de usabilidad**

* **Objetivo**: Evaluar la experiencia del usuario con la interfaz.
* **Ejemplos**:
  + Pruebas con empleados para verificar la facilidad de uso del sistema de reconocimiento facial.
  + Pruebas con el equipo de recursos humanos para evaluar la eficiencia del panel de administración.

**Pruebas de rendimiento**

* **Objetivo**: Asegurar que el sistema funcione correctamente bajo carga.
* **Ejemplos**:
  + Pruebas con múltiples usuarios accediendo al sistema simultáneamente.
  + Pruebas de tiempo de respuesta para las operaciones más críticas.

**Pruebas de seguridad**

* **Objetivo**: Garantizar que los datos estén protegidos y que el sistema sea resistente a ataques.
* **Ejemplos**:
  + Pruebas de SQL injection y XSS.
  + Validación del cifrado de datos biométricos.
  + Verificación de los controles de acceso al panel de administración.

**Pruebas de compatibilidad**

* **Objetivo**: Asegurar que el sistema funcione correctamente en diferentes navegadores y dispositivos.
* **Ejemplos**:
  + Pruebas en Chrome, Firefox, Safari y Edge.
  + Pruebas en dispositivos móviles y tablets.

**3. Resultados de las pruebas**

Tras realizar las pruebas, se han obtenido los siguientes resultados:

* **Funcionalidad**: El sistema cumple con todos los requisitos funcionales establecidos.
* **Rendimiento**: El tiempo de respuesta es inferior a 2 segundos para la mayoría de las operaciones.
* **Seguridad**: No se han detectado vulnerabilidades críticas en las pruebas de seguridad.
* **Usabilidad**: Los usuarios han valorado positivamente la facilidad de uso y la claridad de las interfaces.

**4. Corrección de errores y optimización**

Durante las pruebas, se identificaron algunos errores y áreas de mejora, que fueron corregidos antes del despliegue final:

* **Errores de reconocimiento facial**: Se ajustaron los parámetros del algoritmo para mejorar la precisión.
* **Problemas de rendimiento**: Se optimizaron las consultas a la base de datos y se implementó caching.
* **Mejoras en la interfaz**: Se realizaron ajustes en el diseño para mejorar la experiencia del usuario.

**5. Monitorización post-despliegue**

Una vez desplegado el sistema, se ha establecido un plan de monitorización para:

* Detectar y resolver incidencias en tiempo real.
* Recopilar feedback de los usuarios para futuras mejoras.
* Realizar actualizaciones periódicas para mantener la seguridad y el rendimiento.

# Contexto laboral

El desarrollo de esta aplicación de control horario mediante reconocimiento facial se enmarca en un contexto laboral en el que la digitalización y la automatización de procesos son cada vez más importantes. A continuación, se analiza el entorno laboral en el que se inserta este proyecto, así como su relevancia y aplicabilidad en el mundo empresarial actual.

**1. Digitalización en el entorno laboral**

En los últimos años, las empresas han experimentado una transformación digital acelerada, impulsada por la necesidad de optimizar procesos, reducir costes y mejorar la eficiencia. En este contexto, herramientas como la aplicación desarrollada en este proyecto cobran especial relevancia, ya que permiten:

* **Automatizar tareas repetitivas**: Eliminar la necesidad de registros manuales, reduciendo errores y ahorrando tiempo.
* **Mejorar la precisión**: Garantizar que los datos recopilados sean exactos y confiables.
* **Fomentar la transparencia**: Proporcionar a los empleados acceso a su información laboral, promoviendo la confianza y la satisfacción.

**2. Necesidad de control horario eficiente**

El control horario es un aspecto fundamental en la gestión de recursos humanos, ya que permite:

* **Cumplir con la normativa laboral**: En muchos países, las empresas están obligadas a registrar las horas trabajadas por sus empleados.
* **Optimizar la productividad**: Conocer el tiempo dedicado a cada tarea ayuda a identificar áreas de mejora y a distribuir mejor las cargas de trabajo.
* **Gestionar nóminas**: Los registros horarios son esenciales para calcular salarios, horas extras y otros conceptos retributivos.

Sin embargo, los métodos tradicionales de control horario (fichas de papel, tarjetas magnéticas, etc.) presentan limitaciones que dificultan su eficacia. La aplicación desarrollada en este proyecto ofrece una solución moderna y eficiente a estos problemas.

**3. Aplicación en diferentes sectores**

La aplicación de control horario mediante reconocimiento facial es adaptable a una amplia variedad de sectores, entre los que destacan:

1. **Industria**: En fábricas y plantas de producción, donde es necesario registrar la presencia de un gran número de empleados.
2. **Comercio**: En tiendas y centros comerciales, donde los turnos de trabajo son variables y es importante garantizar la puntualidad.
3. **Servicios**: En empresas de logística, transporte o atención al cliente, donde los empleados pueden trabajar en diferentes ubicaciones.
4. **Oficinas**: En entornos corporativos, donde la aplicación puede integrarse con otras herramientas de gestión empresarial.

**4. Impacto en la gestión de recursos humanos**

La implementación de esta aplicación tiene un impacto significativo en la gestión de recursos humanos, ya que:

* **Simplifica procesos**: Reduce la carga administrativa asociada al control horario, permitiendo al equipo de recursos humanos centrarse en tareas más estratégicas.
* **Mejora la comunicación**: Proporciona una plataforma centralizada para gestionar incidencias y resolver conflictos.
* **Fomenta la equidad**: Garantiza que todos los empleados sean tratados de la misma manera, evitando favoritismos o discrepancias.

**5. Adaptación a normativas laborales**

La aplicación ha sido diseñada teniendo en cuenta las normativas laborales vigentes, como el Reglamento General de Protección de Datos (RGPD) en la Unión Europea. Esto incluye:

* **Protección de datos biométricos**: Los datos faciales se almacenan de forma segura y solo se utilizan para el propósito específico del control horario.
* **Transparencia**: Los empleados tienen acceso a su información y pueden solicitar su modificación o eliminación.
* **Consentimiento**: Se solicita el consentimiento explícito de los empleados antes de utilizar sus datos biométricos.

**6. Futuro del control horario**

El uso de tecnologías biométricas, como el reconocimiento facial, está en auge y se espera que siga creciendo en los próximos años. Este proyecto se alinea con las tendencias actuales y futuras, ofreciendo una solución innovadora que combina tecnología, seguridad y usabilidad. Además, su escalabilidad y adaptabilidad la convierten en una herramienta valiosa para empresas de todos los tamaños y sectores.

# Conclusiones

El desarrollo de esta aplicación de control horario mediante reconocimiento facial ha permitido alcanzar los objetivos planteados inicialmente, ofreciendo una solución innovadora y eficiente para la gestión de entradas y salidas en el entorno laboral. A lo largo del proyecto, se han consolidado conocimientos técnicos y se han superado desafíos que han enriquecido tanto el proceso como el resultado final. A continuación, se resumen las principales conclusiones obtenidas:

**1. Cumplimiento de objetivos**

El proyecto ha cumplido con los objetivos generales y específicos establecidos en la fase de planificación:

* Se ha desarrollado un sistema de reconocimiento facial funcional y preciso, capaz de identificar a los empleados y registrar sus horas de trabajo de forma automática.
* Se ha creado una base de datos robusta y segura para almacenar la información de los empleados y sus registros horarios.
* Se han implementado interfaces de usuario intuitivas y accesibles, tanto para los empleados como para el equipo de recursos humanos.
* Se ha garantizado la escalabilidad del sistema, permitiendo su adaptación a empresas de diferentes tamaños y sectores.

**2. Innovación y mejora de procesos**

La aplicación representa una mejora significativa respecto a los métodos tradicionales de control horario, ya que:

* Elimina los errores humanos asociados a los registros manuales.
* Aumenta la seguridad al verificar la identidad de los empleados mediante reconocimiento facial.
* Facilita la gestión de recursos humanos, reduciendo la carga administrativa y mejorando la eficiencia.

**3. Aplicación de conocimientos**

El proyecto ha permitido aplicar y consolidar los conocimientos adquiridos durante el grado superior de Desarrollo de Aplicaciones Web, integrando tecnologías como JavaScript, PHP, MariaDB y el entorno LAMP. Además, se han adquirido nuevas habilidades en áreas como el reconocimiento facial, la gestión de bases de datos y la seguridad informática.

**4. Desafíos superados**

Durante el desarrollo del proyecto, se han enfrentado y superado varios desafíos técnicos y organizativos:

* **Precisión del reconocimiento facial**: Se optimizaron los algoritmos para garantizar una identificación precisa en diferentes condiciones de iluminación y ángulos.
* **Integración de tecnologías**: Se logró una comunicación fluida entre el frontend, el backend y la base de datos, garantizando un funcionamiento eficiente.
* **Seguridad de los datos**: Se implementaron medidas de cifrado y controles de acceso para proteger la información sensible de los empleados.

**5. Impacto en el entorno laboral**

La aplicación tiene un impacto positivo en el entorno laboral, ya que:

* Mejora la experiencia de los empleados al simplificar el proceso de fichaje y proporcionar acceso transparente a su historial laboral.
* Facilita la gestión del equipo de recursos humanos, permitiéndoles centrarse en tareas más estratégicas.
* Contribuye a la digitalización de los procesos empresariales, alineándose con las tendencias actuales y futuras.

**6. Limitaciones y aprendizajes**

Aunque el proyecto ha sido un éxito, también se han identificado algunas limitaciones y áreas de mejora:

* **Dependencia de la calidad de la cámara**: La precisión del reconocimiento facial puede verse afectada por la calidad de la cámara utilizada.
* **Necesidad de formación**: Los usuarios pueden requerir una breve formación para familiarizarse con el sistema.
* **Escalabilidad en grandes empresas**: En empresas con miles de empleados, podría ser necesario optimizar aún más el rendimiento del sistema.

Estas limitaciones han servido como aprendizajes valiosos para futuras iteraciones del proyecto.

**7. Conclusión final**

En conclusión, este proyecto ha demostrado que es posible desarrollar una solución tecnológica innovadora y eficiente para el control horario de empleados, combinando reconocimiento facial, desarrollo web y gestión de datos. La aplicación no solo cumple con los requisitos técnicos y funcionales establecidos, sino que también tiene un impacto positivo en la experiencia de los usuarios y en la gestión empresarial. Este trabajo representa un paso importante hacia la digitalización de los procesos laborales y sienta las bases para futuras mejoras y ampliaciones.

# Vías futuras

El desarrollo de esta aplicación de control horario mediante reconocimiento facial ha sentado las bases para futuras mejoras y ampliaciones que podrían aumentar su funcionalidad, eficiencia y alcance. A continuación, se describen algunas de las vías futuras que podrían explorarse para seguir avanzando en este proyecto:

**1. Integración con otras tecnologías biométricas**

* **Reconocimiento de huella dactilar**: Añadir la opción de utilizar huellas dactilares como método alternativo de identificación, especialmente en entornos donde el reconocimiento facial pueda no ser viable.
* **Reconocimiento de voz**: Implementar un sistema de identificación por voz para ofrecer mayor flexibilidad a los empleados.

**2. Desarrollo de una aplicación móvil**

* **App para empleados**: Crear una aplicación móvil que permita a los empleados registrar sus horas de trabajo desde cualquier ubicación, utilizando la cámara del móvil para el reconocimiento facial.
* **App para recursos humanos**: Desarrollar una aplicación móvil para que el equipo de recursos humanos pueda gestionar incidencias y revisar registros desde cualquier lugar.

**3. Mejoras en el reconocimiento facial**

* **Mayor precisión**: Implementar algoritmos más avanzados de reconocimiento facial, como los basados en redes neuronales profundas, para mejorar la precisión en condiciones adversas (iluminación baja, ángulos difíciles, etc.).
* **Detección de emociones**: Añadir funcionalidades para detectar el estado emocional de los empleados, lo que podría ser útil para mejorar el bienestar laboral.

**4. Integración con sistemas de gestión empresarial**

* **ERP y CRM**: Conectar la aplicación con sistemas de planificación de recursos empresariales (ERP) y gestión de relaciones con clientes (CRM) para automatizar procesos como la generación de nóminas o la gestión de turnos.
* **Herramientas de análisis**: Integrar herramientas de análisis de datos para proporcionar informes más detallados y predictivos sobre la productividad y el absentismo laboral.

**5. Mejoras en la seguridad**

* **Autenticación multifactor**: Implementar un sistema de autenticación multifactor que combine reconocimiento facial con otros métodos, como contraseñas o tokens, para aumentar la seguridad.
* **Cifrado avanzado**: Utilizar técnicas de cifrado más avanzadas para proteger los datos biométricos y personales de los empleados.

**6. Escalabilidad y adaptabilidad**

* **Soporte para grandes empresas**: Optimizar el sistema para manejar un gran volumen de empleados y registros, garantizando un rendimiento óptimo incluso en empresas de gran tamaño.
* **Personalización**: Ofrecer opciones de personalización para adaptar la aplicación a las necesidades específicas de cada empresa, como la inclusión de campos personalizados o la integración con sistemas propietarios.

**7. Internacionalización y localización**

* **Idiomas**: Añadir soporte para múltiples idiomas, facilitando la adopción de la aplicación en empresas internacionales.
* **Normativas locales**: Adaptar la aplicación para cumplir con las normativas laborales y de protección de datos de diferentes países.

**8. Inteligencia artificial y aprendizaje automático**

* **Predicción de ausencias**: Utilizar técnicas de aprendizaje automático para predecir posibles ausencias o retrasos basándose en patrones históricos.
* **Optimización de turnos**: Implementar algoritmos de optimización para asignar turnos de trabajo de manera más eficiente, teniendo en cuenta las preferencias y disponibilidad de los empleados.

**9. Mejoras en la experiencia del usuario**

* **Interfaz más intuitiva**: Continuar refinando el diseño de las interfaces para mejorar la experiencia del usuario y reducir la curva de aprendizaje.
* **Feedback en tiempo real**: Añadir funcionalidades que permitan a los empleados recibir feedback en tiempo real sobre su registro horario, como notificaciones de confirmación o alertas de errores.

**10. Colaboración con otras empresas**

* **Alianzas estratégicas**: Establecer colaboraciones con empresas especializadas en tecnologías biométricas o gestión de recursos humanos para enriquecer la aplicación con nuevas funcionalidades y conocimientos.
* **Open source**: Considerar la posibilidad de liberar parte del código como open source, permitiendo a la comunidad contribuir al desarrollo y mejora del sistema.

**11. Sostenibilidad y responsabilidad social**

* **Reducción de papel**: Promover la eliminación del uso de papel en los procesos de control horario, contribuyendo a la sostenibilidad medioambiental.
* **Inclusión**: Asegurar que la aplicación sea accesible para todos los empleados, incluyendo aquellos con discapacidades, mediante la implementación de funcionalidades adicionales como el reconocimiento de voz o la compatibilidad con dispositivos de asistencia.

Este apartado sugiere posibles mejoras y ampliaciones que podrían realizarse en el futuro para seguir desarrollando el proyecto. Puedes ajustarlo para reflejar tus propias ideas o prioridades. ¡Espero que te sea útil!

# Glosario

**A**

* **Algoritmo**: Conjunto de pasos o reglas definidas para resolver un problema o realizar una tarea. En este proyecto, se utiliza para el reconocimiento facial.
* **Apache**: Servidor web de código abierto utilizado para gestionar las solicitudes HTTP en el entorno LAMP.

**B**

* **Backend**: Parte del sistema que se ejecuta en el servidor, gestionando la lógica de negocio, la base de datos y la comunicación con el frontend.
* **Base de datos**: Sistema organizado para almacenar, gestionar y recuperar información. En este proyecto, se utiliza MariaDB.
* **Biometría**: Técnicas de identificación basadas en características físicas o conductuales únicas, como el reconocimiento facial o de huellas dactilares.

**C**

* **Caching**: Técnica para almacenar temporalmente datos o resultados de operaciones, con el fin de mejorar el rendimiento del sistema.
* **Cliente-servidor**: Modelo de arquitectura en el que los clientes (frontend) solicitan servicios o recursos a un servidor (backend).

**D**

* **Despliegue**: Proceso de implementar y poner en funcionamiento una aplicación en un entorno de producción.
* **Docker**: Plataforma que permite empaquetar aplicaciones en contenedores, facilitando su despliegue en diferentes entornos.

**E**

* **ERP (Enterprise Resource Planning)**: Sistema de gestión empresarial que integra y automatiza procesos como contabilidad, recursos humanos y gestión de inventarios.

**F**

* **FaceAPI.js**: Biblioteca de JavaScript utilizada para implementar el reconocimiento facial en el frontend.
* **Frontend**: Parte del sistema con la que interactúa el usuario, incluyendo las interfaces gráficas y la lógica de presentación.

**G**

* **Git**: Sistema de control de versiones utilizado para gestionar el código fuente y colaborar en equipo.
* **GitHub**: Plataforma basada en Git para alojar y compartir proyectos de desarrollo de software.

**H**

* **HTML (HyperText Markup Language)**: Lenguaje de marcado utilizado para crear la estructura de las páginas web.

**I**

* **Integración continua (CI)**: Práctica de desarrollo que consiste en integrar cambios de código frecuentemente, realizando pruebas automáticas para detectar errores.

**J**

* **JavaScript**: Lenguaje de programación utilizado para implementar el reconocimiento facial y la interactividad en el frontend.
* **jQuery**: Biblioteca de JavaScript que simplifica la manipulación del DOM y la gestión de eventos.

**L**

* **LAMP**: Acrónimo de Linux, Apache, MariaDB/MySQL y PHP, un conjunto de tecnologías utilizado para desarrollar y alojar aplicaciones web.

**M**

* **MariaDB**: Sistema de gestión de bases de datos relacional, derivado de MySQL, utilizado en este proyecto para almacenar datos.
* **Metodología ágil**: Enfoque de desarrollo basado en iteraciones cortas y colaboración continua con los usuarios.

**P**

* **PHP**: Lenguaje de programación utilizado en el backend para gestionar la lógica de negocio y la interacción con la base de datos.
* **Postman**: Herramienta utilizada para probar APIs y verificar la comunicación entre el frontend y el backend.

**R**

* **Reconocimiento facial**: Tecnología que identifica o verifica a una persona a partir de una imagen o video de su rostro.
* **Responsive design**: Enfoque de diseño que garantiza que las interfaces se adapten a diferentes dispositivos y tamaños de pantalla.

**S**

* **SQL (Structured Query Language)**: Lenguaje utilizado para gestionar y consultar bases de datos relacionales.
* **Stack tecnológico**: Conjunto de tecnologías utilizadas para desarrollar una aplicación. En este proyecto, el stack incluye JavaScript, PHP, MariaDB y LAMP.

**T**

* **TensorFlow.js**: Biblioteca de JavaScript para entrenar y ejecutar modelos de machine learning en el navegador.
* **Testing**: Proceso de verificar que el software funciona correctamente y cumple con los requisitos establecidos.

**U**

* **Usabilidad**: Facilidad con la que los usuarios pueden interactuar con una aplicación para lograr sus objetivos.

**V**

* **Visual Studio Code**: Editor de código utilizado para desarrollar la aplicación, con soporte para múltiples lenguajes y extensiones.

**X**

* **XAMPP**: Paquete de software que incluye Apache, MariaDB/MySQL, PHP y Perl, utilizado para crear un entorno de desarrollo local.

# Bibliografía y Webgrafía

1. **Reconocimiento facial y JavaScript**:
   * TensorFlow.js Documentation. (2023). Face and body detection. Disponible en: <https://www.tensorflow.org/js>
   * FaceAPI.js GitHub Repository. (2023). Face recognition in the browser. Disponible en: <https://github.com/justadudewhohacks/face-api.js>
2. **Desarrollo web y tecnologías LAMP**:
   * MDN Web Docs. (2023). JavaScript, HTML, and CSS documentation. Disponible en: [https://developer.mozilla.org](https://developer.mozilla.org/)
   * PHP Manual. (2023). Official PHP documentation. Disponible en: <https://www.php.net/manual>
   * MariaDB Knowledge Base. (2023). MariaDB documentation. Disponible en: <https://mariadb.com/kb>
3. **Diseño de interfaces y usabilidad**:
   * Nielsen, J. (2023). Usability and user experience (UX) articles. Disponible en: [https://www.nngroup.com](https://www.nngroup.com/)
   * Figma Blog. (2023). Design tools and tutorials. Disponible en: <https://www.figma.com/blog>
4. **Seguridad y protección de datos**:
   * GDPR.eu. (2023). General Data Protection Regulation (GDPR) guidelines. Disponible en: [https://gdpr.eu](https://gdpr.eu/)
   * OWASP. (2023). Open Web Application Security Project. Disponible en: [https://owasp.org](https://owasp.org/)
5. **Gestión de proyectos y metodologías ágiles**:
   * Scrum.org. (2023). Scrum framework and resources. Disponible en: [https://www.scrum.org](https://www.scrum.org/)
   * Trello Blog. (2023). Project management tips and tools. Disponible en: [https://blog.trello.com](https://blog.trello.com/)
6. **Herramientas de desarrollo y pruebas**:
   * Postman Learning Center. (2023). API testing and development. Disponible en: [https://learning.postman.com](https://learning.postman.com/)
   * Jest Documentation. (2023). JavaScript testing framework. Disponible en: [https://jestjs.io](https://jestjs.io/)
7. **Despliegue y servidores**:
   * Apache HTTP Server Documentation. (2023). Apache web server guides. Disponible en: <https://httpd.apache.org/docs>
   * Docker Documentation. (2023). Containerization and deployment. Disponible en: [https://docs.docker.com](https://docs.docker.com/)

# Anexos

material utilizado, cuestionarios, etc.