APLICACIÓN WEB PARA LA GESTIÓN Y CONTROL DE TAREAS Y EMPLEADOS

PALOMA GÓMEZ QUINTERO

Resumen

El Sistema de Gestión de Tiempos y Tareas constituye una plataforma web empresarial integral desarrollada para optimizar radicalmente la administración de recursos humanos y el seguimiento de actividades laborales en organizaciones de diversos tamaños y sectores. Esta solución tecnológica proporciona un entorno digital unificado donde convergen funcionalidades de gestión administrativa, coordinación operativa y análisis estratégico, estructurado en tres ejes fundamentales: un módulo de administración completa de la plantilla laboral con capacidades de ciclo de vida del empleado, un sistema inteligente de asignación y seguimiento de tareas con algoritmos de distribución optimizada, y un conjunto avanzado de herramientas analíticas para la transformación de datos operativos en inteligencia empresarial accionable. La plataforma, desplegada en el dominio tiempo.desarrollos.jesusjg.es, presenta una arquitectura técnica basada en tecnologías web modernas que garantizan escalabilidad, seguridad y mantenibilidad, destacando por su interfaz responsiva adaptativa, su sistema de permisos granulares basado en roles y su capacidad de integración con ecosistemas empresariales existentes.

Características principales:

- Gestión completa de empleados con perfiles detallados
- Sistema de asignación y seguimiento de tareas en tiempo real
- Interfaz adaptativa que funciona en dispositivos móviles y de escritorio
- Control de acceso basado en roles (administrador vs. empleado)
- Visualización intuitiva del estado de tareas y carga de trabajo

URLs principales del sistema:

1. Panel de Administración - Empleados: https://tiempo.desarrollos.jesusjg.es/admin/empleados

2. Panel de Administración - Tareas: https://tiempo.desarrollos.jesusjg.es/admin/tareas

3. Perfil de Empleado (Ejemplo): https://tiempo.desarrollos.jesusjg.es/empleado/perfil/97

4. Login: https://tiempo.desarrollos.jesusjg.es/login

Índice General

1. Introducción

En el panorama empresarial contemporáneo, caracterizado por una aceleración digital sin precedentes y una creciente complejidad operativa, emerge como imperativo estratégico la implementación de soluciones tecnológicas que optimicen la gestión del capital humano y las actividades productivas. Las organizaciones modernas enfrentan desafíos sistémicos en la coordinación de equipos multidisciplinares, la asignación eficiente de responsabilidades y el seguimiento preciso del progreso de iniciativas, problemáticas que frecuentemente derivan de la persistencia de metodologías manuales, sistemas fragmentados y flujos de información desestructurados. Este proyecto surge como respuesta directa a estas necesidades críticas, con el propósito de desarrollar una plataforma web integral que centralice, automatice y optimice los procesos de gestión organizacional mediante la consolidación de tres dimensiones fundamentales: la administración del recurso humano, la coordinación de actividades productivas y el análisis del desempeño institucional. La solución propuesta se fundamenta en principios de diseño centrado en el usuario, arquitecturas tecnológicas modernas y mejores prácticas de desarrollo de software, configurándose no solo como una herramienta operativa sino como un habilitador estratégico para la transformación digital organizacional.

2. Objetivos

2.1. Objetivo General

Desarrollar e implementar una plataforma web integral, escalable y segura para la gestión unificada de recursos humanos, asignación de tareas y análisis de productividad organizacional, que optimice los procesos operativos, incremente la eficiencia administrativa y proporcione visibilidad en tiempo real sobre el desempeño institucional, mediante la aplicación de tecnologías web modernas, principios de diseño centrado en el usuario y metodologías ágiles de desarrollo.

2.2. Objetivos Específicos

2.2.1. Objetivos Funcionales y de Negocio

1. **Centralización de la Información Organizacional**: Consolidar en una única plataforma digital toda la información relativa a empleados, tareas, proyectos y métricas de desempeño, eliminando la fragmentación de datos y facilitando el acceso unificado a información crítica.

2. **Automatización de Procesos Administrativos**: Reducir significativamente la carga de trabajo manual mediante la automatización de procesos repetitivos como la creación de perfiles de empleados, asignación inicial de tareas, generación de reportes estándar y notificaciones sistemáticas.

3. **Optimización de la Asignación de Recursos**: Implementar mecanismos inteligentes para la distribución de tareas que consideren múltiples variables como habilidades específicas, disponibilidad temporal, carga laboral actual y preferencias individuales, maximizando la eficiencia en la utilización del capital humano.

4. **Mejora de la Visibilidad y Transparencia**: Establecer un sistema de dashboards y reportes en tiempo real que proporcionen a todos los niveles organizacionales (desde dirección hasta empleados individuales) una visión clara y actualizada del estado de proyectos, progreso de tareas y desempeño colectivo.

5. **Facilitación de la Colaboración Interdepartamental**: Crear canales estructurados de comunicación y colaboración que superen las barreras departamentales, permitiendo la coordinación efectiva entre equipos, el intercambio de conocimiento y la resolución colaborativa de problemas.

2.2.2. Objetivos Técnicos y de Desarrollo

1. **Implementación de Arquitectura Escalable y Mantenible**: Diseñar y desarrollar una arquitectura de software basada en principios de separación de responsabilidades,

modularidad y bajo acoplamiento, que permita el crecimiento progresivo del sistema y facilite su mantenimiento y evolución futura.

2. **Garantía de Seguridad y Protección de Datos**: Establecer un marco robusto de seguridad informática que incluya autenticación multifactor, encriptación de datos sensibles, control de acceso basado en roles, protección contra amenazas comunes (SQL injection, XSS, CSRF) y cumplimiento con normativas de protección de datos (GDPR, LOPDGDD).

3. **Optimización del Rendimiento y Disponibilidad**: Asegurar tiempos de respuesta inferiores a 2 segundos para operaciones críticas, disponibilidad del sistema superior al 99.5% y capacidad para soportar cargas concurrentes de usuarios activos simultáneamente, mediante técnicas de caché, optimización de consultas y balanceo de carga.

4. **Desarrollo de Interfaz de Usuario Intuitiva y Responsiva**: Crear una experiencia de usuario coherente, accesible y eficiente que funcione consistentemente en dispositivos de escritorio, tablets y smartphones, siguiendo principios de diseño UX/UI probados y estándares de accesibilidad WCAG 2.1.

5. **Creación de API RESTful Documentada**: Diseñar e implementar una interfaz de programación de aplicaciones (API) bien documentada, versionada y segura que permita la integración con sistemas externos (ERP, CRM, sistemas de nómina) y el desarrollo de aplicaciones complementarias.

2.2.3. Objetivos de Calidad y Sostenibilidad

1. **Establecimiento de Procesos de Calidad del Software**: Implementar una estrategia integral de aseguramiento de calidad que incluya pruebas unitarias, de integración, de rendimiento y de seguridad, con cobertura de código superior al 80% y procesos de revisión sistemática.

2. **Documentación Exhaustiva del Sistema**: Producir documentación técnica completa que abarque especificaciones funcionales, manuales de usuario, guías de instalación, documentación de API y procedimientos de mantenimiento, facilitando la transferencia de conocimiento y la sostenibilidad del proyecto.

3. **Formación y Transferencia de Conocimiento**: Diseñar e implementar programas de capacitación para administradores del sistema y usuarios finales, asegurando la adopción efectiva de la plataforma y el desarrollo de competencias internas para su gestión y explotación óptima.

4. **Planificación de Mantenimiento y Evolución**: Establecer un modelo operativo que defina claramente responsabilidades, procesos de soporte, ciclos de actualización y mecanismos para la incorporación de nuevas funcionalidades, garantizando la vida útil y relevancia continua del sistema.

3. Arquitectura del Sistema

3.1. Arquitectura General Monolítica (Laravel MVC)

El sistema desarrollado sigue una arquitectura monolítica basada en el patrón MVC (Modelo–Vista–Controlador) implementado mediante el framework Laravel, que actúa como núcleo del back-end. Todo el código del proyecto —lógica de negocio, controladores, modelos, vistas y capas auxiliares— se ejecuta dentro de una misma aplicación, organizada de forma modular para facilitar su mantenimiento.

Laravel ofrece un conjunto de componentes que permiten estructurar la aplicación según buenas prácticas del desarrollo moderno: rutas, controladores, migraciones, seeders, middlewares, colas, plantillas Blade y su ORM nativo Eloquent.

En esta arquitectura:

- El cliente (navegador web) realiza peticiones HTTP a través de las rutas definidas.
- Los Controladores gestionan las solicitudes, validan datos y consultan los modelos.
- Los Modelos interactúan con la base de datos MySQL mediante Eloquent ORM.
- Las Vistas (Blade) componen el HTML dinámico que se envía al usuario final.

Esta estructura proporciona separación de responsabilidades, facilita la escalabilidad vertical y permite extender funcionalidades sin comprometer la estabilidad del sistema.

3.2. Infraestructura del Sistema

El proyecto se ejecuta en un entorno web compuesto por:

- **Lenguajes:** PHP, HTML5, CSS3, JavaScript.
- **Framework back-end:** Laravel.
- **Framework front-end:** Bootstrap.
- **Servidor web de producción:** cPanel.
- **Base de datos:** MySQL.
- **Navegación y consumo del sistema:** vía navegador web, accesible desde cualquier dispositivo (diseño responsive).

3.3. Estructura del Proyecto

La estructura interna del proyecto está dividida en dos grandes bloques:

- **Front-end**

Se encuentra principalmente en la carpeta **resources/views**, organizada por roles:

- /views/admin → Panel y secciones exclusivas para administrador
- /views/empleado → Secciones del empleado
- /views/auth → Login + autenticación QR
- /views/exports → Vistas generadas para exportación de informes

Dentro de cada módulo existen subsecciones, como:

(imagen)

El front-end utiliza:

- **Bootstrap** para estilos responsive.
- **Blade** para plantillas.
- **JavaScript** para acciones dinámicas en tablas, formularios y modales.

- **Back-end**

Contenido en la ruta **/app**, separado en **Models**, **Http/Controllers**, **Exports**, etc.

- **Carpeta Models**

Incluye las entidades principales del proyecto:

- Empleado.php
- Tarea.php
- RegistroTiempo.php
- Credencial.php
- Qr.php
- Rol.php
- User.php
- Otros modelos relacionados con la lógica de gestión del personal.

Estos modelos definen:

- Relaciones Eloquent (hasMany, belongsTo…)
- Reglas de asignación de campos
- Conexión directa con la base de datos MySQL

- **Carpeta Controllers**

Controladores encargados de la lógica de negocio y procesamiento de datos:

- AdminController.php
- EmpleadoController.php
- LoginController.php
- LoginQrController.php
- Controller.php

Cada controlador gestiona un módulo del sistema: asignación de tareas, gestión de empleados, registro de tiempos, escaneo QR, etc.

- **Carpeta Exports**

Incluye clases para exportar datos en PDF y Excel:

- EmpleadosMesExport.php
- EmpleadosPdfExport.php
- RegistroHorarioIndividualExport.php

Estas clases utilizan librerías como **Maatwebsite/Excel** o PDF wrappers compatibles con Laravel.

3.4. Componentes del Sistema

En la arquitectura intervienen los siguientes componentes:

- **Cliente (Front-end)**

  - Formado por HTML + CSS + JavaScript + Bootstrap.
  - Renderiza datos recibidos desde Laravel.
  - Maneja acciones de usuario: formularios, tablas, filtros, botones, exportaciones.

- **Servidor (Back-end Laravel)**

  - Gestiona autenticación y roles.
  - Aplica reglas de negocio.
  - Controla validaciones.
  - Expone datos a través de vistas.
  - Gestiona exportaciones y generación de informes.

- **Base de Datos (MySQL)**

  Responsable del almacenamiento de:

  - Empleados
  - Tareas
  - Registros de tiempo

- Datos del perfil
- Tokens QR
- Roles y permisos

3.5. Flujo General de Funcionamiento

1. El usuario accede al sistema mediante login o QR.
2. Laravel identifica el rol (Administrador o Empleado).
3. Según el rol, se cargan las vistas correspondientes:
   - Panel del Administrador
   - Perfil y datos del empleado
4. Las acciones (crear tarea, actualizar empleado, registrar tiempo, etc.) son enviadas a los controladores.
5. Los controladores consultan/actualizan los modelos Eloquent.
6. La base de datos devuelve los resultados.
7. Las vistas Blade muestran la información actualizada al usuario.

3.6. Justificación Tecnológica

La elección de Laravel, PHP y MySQL responde a:

- Rapidez de desarrollo
- ORM Eloquent potente y simple
- Sistema de rutas y middlewares muy flexible
- Plantillas Blade eficientes
- Escalabilidad del modelo MVC
- Compatibilidad total con hosting estándar
- Facilidad de despliegue

Bootstrap permite crear una interfaz moderna, limpia y responsive sin aumentar la complejidad del sistema.

3.7. Conclusión del Apartado

La arquitectura implementada proporciona una solución sólida, modular y mantenible. El uso del patrón MVC junto con Laravel permite dividir la aplicación en capas bien definidas, mientras que MySQL garantiza un rendimiento adecuado para la gestión de empleados, tareas, registros y credenciales.

La estructura monolítica es coherente con el tamaño y objetivos del proyecto, facilitando la ampliación futura (módulos adicionales, API REST, o migración a microservicios si fuera necesario).

4. Roles de Usuarios y Sistema de Permisos

En esta aplicación web se implementa un sistema de gestión de usuarios basado en **roles**, con el objetivo de controlar el acceso a las diferentes secciones del sistema y garantizar que cada usuario solo pueda visualizar y ejecutar las acciones correspondientes a sus responsabilidades.

El sistema utiliza el modelo tradicional de control de acceso por roles (RBAC), integrado con el middleware de autenticación de Laravel. Dependiendo del rol asignado, el usuario obtiene permisos específicos para acceder a determinadas vistas, ejecutar acciones sobre los recursos o interactuar con módulos concretos.

A continuación, se describen los distintos roles definidos en la aplicación.

4.1. Rol de Administrador

El **Administrador** es el usuario con mayor nivel de permisos dentro del sistema.
Es responsable de toda la gestión interna del personal y del control de tareas asociadas a la actividad laboral.

Entre sus funcionalidades principales se incluyen:

- Acceso a la página de gestión de empleados: https://tiempo.desarrollos.jesusjg.es/admin/empleados
- Gestión completa del personal: creación, edición, activación/inactivación y eliminación de empleados.
- Gestión de las tareas asignadas a cada empleado: https://tiempo.desarrollos.jesusjg.es/admin/tareas
- Acceso al panel general de control (dashboard administrativo).
- Generación de informes y exportaciones en PDF/Excel.
- Acceso total al historial y registro temporal de los empleados.
- Administración de credenciales y tokens QR de acceso.
- Visualización y edición completa del perfil de cualquier usuario del sistema.

El Administrador posee un perfil técnico-operativo y accede a todos los módulos sin restricciones.

4.2. Rol de Empleado

El Empleado representa al trabajador que forma parte de la organización y utiliza la plataforma para gestionar su información y sus tareas asignadas.

Sus funciones principales incluyen:

- Acceso a su perfil personal: https://tiempo.desarrollos.jesusjg.es/empleado/perfil/{id}

- Consulta y edición limitada de sus datos personales.

- Visualización de las tareas asignadas por el Administrador.

- Registro de su actividad laboral:

    - Entradas y salidas.

    - Registro de tiempos por tarea.

- Descarga de informes individuales (si el Administrador lo habilita).

- Escaneo y uso del sistema de acceso mediante QR.

Este rol tiene permisos restringidos, centrados únicamente en su propia actividad y su información personal, sin acceso a ningún módulo de gestión global.

4.3. Sistema de Permisos y Restricciones

La aplicación implementa un sistema de control basado en middleware y validaciones internas:

Permisos del Administrador

- Acceso total a módulos /admin/*.
- Manipulación de todos los modelos del sistema: Empleado, Tarea, RegistroTiempo, Credencial, Roles, etc.
- Capacidad para generar exportaciones y consultar estadísticas.

Permisos del Empleado

- Acceso únicamente a /empleado/*.
- No puede acceder a módulos administrativos ni modificar información externa a su perfil.
- Sus acciones están registradas en auditorías internas.

Protecciones adicionales

- Rutas protegidas por **middleware auth** y **middleware por rol**.
- Ocultación en menú de opciones no permitidas.
- Validaciones que impiden el acceso a recursos que no correspondan al usuario logueado.
- Gestión segura de contraseñas mediante hashing.

5. Páginas y Funcionalidades Principales

En este apartado se describen las principales páginas que componen la aplicación y las funcionalidades asociadas a cada rol del sistema. La plataforma está dividida en dos entornos claramente diferenciados:

- El panel administrativo, accesible solo para usuarios con rol Administrador
- El panel del empleado, con acceso restringido a su información individual

Las páginas se han desarrollado utilizando vistas Blade de Laravel, integradas con Bootstrap para asegurar una interfaz responsiva y consistente en todos los dispositivos.

5.1. Panel del Administrador

El Administrador cuenta con acceso total a la gestión del sistema. Las siguientes páginas forman parte de su entorno:

- Gestión de Empleados

URL: https://tiempo.desarrollos.jesusjg.es/admin/empleados

Esta página permite administrar todo el personal registrado en el sistema.

Incluye:

- Visualización del listado completo de empleados
- Búsqueda rápida mediante filtros
- Creación de nuevos empleados mediante formulario
- Edición de datos personales y laborales
- Activación y desactivación de perfiles
- Eliminación de empleados sin registros dependientes
- Acceso directo al perfil individual

La información se presenta en una tabla dinámica que permite ordenar, paginar y filtrar los datos.

- Gestión de Tareas

URL: https://tiempo.desarrollos.jesusjg.es/admin/tareas

En esta sección el Administrador define y controla las tareas laborales asignadas a los empleados.

Las opciones disponibles son:

- Crear nuevas tareas
- Asignarlas a uno o varios empleados
- Editar la descripción, categoría o estado de una tarea
- Consultar el historial de ejecución por empleado
- Eliminar tareas que ya no sean necesarias
- Exportar listados de tareas a formatos PDF o Excel

Esta página resulta esencial para la planificación de la actividad laboral.

- Panel de Control (Dashboard)

El dashboard del Administrador muestra un resumen general del sistema, incluyendo:

- Número total de empleados
- Tareas activas y completadas
- Registros diarios de actividad
- Accesos mediante QR
- Informes rápidos del rendimiento del equipo

Permite obtener una visión global y tomar decisiones operativas.

- Gestión de Credenciales y Acceso QR

El sistema integra un módulo de acceso por QR.

Desde esta página el Administrador puede:

- Generar códigos QR únicos
- Asociarlos a empleados
- Regenerar claves en caso de pérdida
- Desactivar accesos comprometidos
- Registrar y consultar accesos realizados mediante QR

- Exportaciones y Reportes

El Administrador tiene acceso a diferentes herramientas de exportación:

- Exportación mensual de empleados
- Informe PDF de empleados
- Reportes individuales de registro de tiempo

Estas funciones utilizan las clases del directorio /app/Exports.

5.2. Panel del Empleado

El entorno del empleado incluye únicamente las secciones relacionadas con su actividad.

- Perfil del Empleado

URL: https://tiempo.desarrollos.jesusjg.es/empleado/perfil/97

Esta es la página principal del empleado, donde se muestra su información personal:

- Nombre, apellidos y datos de contacto
- Horario asociado (si aplica)
- Tareas asignadas
- Historial de actividades o registros
- Opciones de actualización de ciertos datos permitidos

Es un panel simplificado pensado para un uso rápido y directo.

- Registro de Tiempo

Los empleados pueden registrar:

- Hora de inicio de jornada
- Pausas o interrupciones
- Finalización de jornada
- Tiempo invertido en cada tarea asignada

Estos datos alimentan el sistema de control horario, obligatorio y accesible desde el panel del Administrador.

- Consulta de Tareas Asignadas

El empleado tiene acceso a un listado con:

- Tareas en curso
- Tareas completadas
- Descripción y prioridad
- Fecha de asignación
- Estado y observaciones

También puede marcar tareas como completadas, si el Administrador lo ha permitido.

- Acceso mediante QR

Desde su perfil, el empleado puede:

- Visualizar su código QR personal
- Usarlo para registrar accesos (entrada/salida)
- Regenerarlo en caso de pérdida (si el Administrador lo autoriza)

5.3. Página de Autenticación

El sistema cuenta con un módulo propio de acceso.

- Inicio de Sesión

Incluye:

- Validación de username y contraseña
- Mensajes de error en caso de credenciales incorrecta
- Redirección automática según rol (Administrador / Empleado)

- Login mediante QR

Permite que el usuario acceda escaneando un código QR válido, evitando el uso de credenciales tradicionales.

6. Base de Datos - Diseño y Modelado

La base de datos utilizada en la plataforma ha sido diseñada siguiendo principios de normalización y optimización para garantizar un almacenamiento eficiente, seguro y escalable. El

sistema emplea MySQL como motor principal, gestionado desde Laravel mediante migraciones y el ORM Eloquent, lo que permite un control estructurado del modelo de datos y facilita futuras ampliaciones.

El diseño relacional se centra en tres ejes fundamentales:

- Gestión de empleados
- Gestión de tareas
- Control horario y autenticación mediante QR

A partir de estos elementos se establecen relaciones entre tablas siguiendo el enfoque clásico de un sistema empresarial de gestión de personal.

6.1. Modelo Entidad/Relación (E/R)

La base de datos está compuesta por un conjunto de entidades interrelacionadas que representan empleados, tareas, registros temporales, tokens QR y credenciales de acceso.

Las entidades principales identificadas son:

- Empleado
- Tarea
- Asignación de Tareas
- Registro de Tiempo
- QR / Tokens
- Usuario (User)
- Rol
- Credencial

A esto se suman tablas auxiliares para relaciones pivot, auditorías y autenticación del sistema.

El modelo sigue el patrón:

- 1:N entre usuarios y roles
- 1:N entre empleados y registros de tiempo
- N:M entre empleados y tareas (a través de una tabla intermedia)
- 1:1 entre empleados y credenciales/QR (según configuración del sistema)

6.2. Descripción de las Tablas Principales

A continuación, se detallan las tablas más relevantes del sistema, su propósito y sus relaciones.

- Tabla: tabla_empleados

Representa a los trabajadores del sistema.

Campos principales:

- Id
- Nombre
- Apellidos
- Fecha_nacimiento
- Dni
- Dirección
- Teléfono
- Latitud
- Longitud
- Credencial_id
- Rol_id
- Qr_id
- Activo
- Ip_conexion
- User_agent
- Ultimo_inicio_sesion
- created_at
- updated_at

Relaciones:

- 1 empleado → 1 credencial (**1:1**)
- 1 empleado → 1 QR (**1:1**)
- 1 empleado → N tareas creadas (**1:N**)
- 1 empleado → N asignaciones de tareas (**1:N** → tabla_asignaciones_tareas)
- 1 empleado → N registros de tiempo (**1:N** → tabla_registros_tiempo)

- Tabla: tabla_tareas

Define las tareas o actividades asignadas a empleados.

Campos:

- Id
- Titulo
- Descripción
- Área

- Tipo_tarea_id
- Prioridad
- Fecha_tarea
- Horas_tarea
- Estado
- Creador_tipo
- Admin_creador_id
- Empleado_creador_id
- Created_at
- Updated_at

Relaciones:

- 1 tarea → N asignaciones (**1:N**)
- 1 tarea → N registros de tiempo (**1:N**)
- 1 tipo de tarea → N tareas (**N:1**)
- 1 administrador → N tareas creadas (**N:1**)
- 1 empleado → N tareas creadas (**N:1**)

- Tabla: tabla_tipos_tarea

Tipos o categorías de tareas.

Campos:

- Id
- Nombre
- Descripción
- Color
- Activo
- Created_At
- Updated_at

Relaciones:

- 1 tipo de tarea → N tareas (1:N)

- Tabla: asignación_tareas

Tabla intermedia que conecta empleados con tareas.

Campos:

- Id
- empleado_id
- tarea_id
- estado_asignacion
- comentarios
- fecha_asignacion
- fecha_completada
- created_at
- updated_at

Relaciones:

- N asignaciones → 1 empleado (**N:1**)
- N asignaciones → 1 tarea (**N:1**)
- Relación global:
  empleados N:M tareas

- Tabla: tabla_registros_tiempo

Registra las horas de entrada, salida y tiempos dedicados por empleado.

Campos:

- Id
- empleado_id
- tarea_id
- hora_inicio
- hora_fin
- total_minutos
- tipo_registro (entrada/salida/tarea)
- latitud
- longitud
- created_at
- updated_at

Relaciones:

- 1 empleado → N registros (**1:N**)
- 1 tarea → N registros (**1:N**)
- Relación opcional: un registro puede no tener tarea asociada

- Tabla: tabla_qr

Gestiona los códigos QR únicos de cada empleado.

Campos:

- Id
- Imagen_qr
- Código_unico
- Contenido_qr
- empleado_id
- activo
- expiracion
- created_at
- updated_at

Relaciones

- 1 QR → 1 empleado (**1:1**)
- 1 QR → N tokens de login (**1:N** → tabla_qr_login_tokens)

El QR se utiliza tanto para acceso como para registro de tiempo.

- Tabla: tabla_qr_login_tokens

Tokens generados al escanear un QR para permitir el login.

Campos:

- Id
- Qr_id
- token
- confirmed_at
- created_at

Relaciones

- 1 QR → N tokens (**1:N**)

- Tabla: tabla_credenciales

Gestiona claves de acceso adicionales o vinculadas al sistema QR.

Campos:

- Id
- Username
- Password
- Remember_token
- Rol_id
- Created_at
- Updated_At

Relaciones:

- Relación 1:1 con empleados.
- 1 credencial → 1 administrador (**1:1**)
- 1 credencial → 1 empleado (**1:1**)
- 1 rol → N credenciales (**N:1**)

- Tabla: tabla_roles

Define los roles del sistema:

- Administrador
- Empleado

Campos:

- Id
- Nombre
- Descripción
- Created_at
- Updated_at

Relaciones:

- Cada rol puede tener múltiples usuarios (1:N)
- 1 rol → N credenciales (**1:N** → tabla_credenciales)
- 1 rol → 1 administrador (**1:1** → tabla_admin)

- Tabla: tabla_admin

Registra la información adicional de los administradores.

Campos:

- Id
- Credencial_id
- Rol_id
- Created_at
- Updated_At

Relaciones:

- 1 administrador → 1 credencial (**1:1**)
- 1 administrador → N tareas creadas (**1:N** → tabla_tareas.admin_creador_id)

- Tabla: tabla_eventos_tiempo

Eventos asociados a un registro (inicio, pausa, fin…).

Campos:

- Id
- Registro_id
- Tipo (enum)
- created_at
- updated_at

Relaciones:

- 1 registro → N eventos (**1:N**)

6.3. Relaciones Principales del Sistema

Relacionamientos clave

- Empleado – RegistroTiempo
  Un empleado puede tener muchos registros de actividad.
- Empleado – Tarea (N:M)
  A través de la tabla asignacion_tareas.
- Empleado – QR (1:1)
  Cada empleado tiene un único código QR.
- Empleado – Credencial (1:1)
  Similar al QR, para claves secundarias.
- User – Rol (N:1)
  Define los permisos del sistema.
- Tarea – RegistroTiempo (1:N)
  Si el registro se asocia con una tarea concreta.

7. API y Endpoints

Aunque el proyecto está concebido como una aplicación monolítica bajo la arquitectura MVC de Laravel, la comunicación entre el cliente y el servidor se realiza mediante un conjunto de **endpoints HTTP** que actúan como una API interna. Estos endpoints permiten gestionar recursos como empleados, tareas, tiempo de trabajo y autenticación, respondiendo en muchos casos con vistas Blade, aunque pueden también devolver datos en formato JSON para operaciones asincrónicas.

Los endpoints están organizados por módulos y protegidos mediante middleware de autenticación y verificación de roles (Administrador o Empleado). A continuación se describen los principales grupos de rutas del sistema.

7.1. EndPoints del Módulo de Administración

URL Base: /admin/*

- Gestión de Empleados

Método	Endpoint	Descripción
GET	/empleados	Listado general
POST	/empleados/store	Crear empleado
GET	/empleados/{id}	Ver perfil individual
GET	/empleados/{id}/edit	Editar
PUT	/empleados/{id}	Actualizar
DELETE	/empleados/{id}	Eliminar

Este conjunto de endpoints implementa un CRUD completo.

- Búsqueda y validaciones

Método	Endpoint	Descripción
GET	/empleados/stats	Estadísticas
GET	/empleados/datatable	Datatable AJAX
GET	/empleados/buscar-por-dni/{dni}	Buscar por DNI
GET	/empleados/verificar-username/{username}	Verificar usuario duplicado
GET	/empleados/stats	Estadísticas

- Registro Horario

Método	Endpoint	Descripción
GET	/empleados/registros/{id}/datatable	DataTable de registros
GET	/empleados/registros/{id}/resumen	Resumen mensual
GET	/empleados/{empleadoId}/registros/{registroId}/detalles	Detalle específico

- Exportaciones

Método	Endpoint	Descripción
GET	/empleados/exportar-excel-mes	Excel mensual
GET	/empleados/exportar-pdf-mes	PDF mensual
GET	/empleados/{id}/exportar-registro-horario	Registro individual

- Gestión de Tareas

Método	Endpoint	Descripción
GET	/tareas	Vista tareas
GET	/tareas/datatable	DataTable
GET	/tareas/tipos	Tipos
GET	/tareas/empleados	Empleados disponibles
POST	/tareas	Crear
GET	/tareas/{id}	Mostrar
PUT	/tareas/{id}	Actualizar
DELETE	/tareas/{id}	Eliminar
POST	/tareas/{id}/asignar	Asignar empleados
POST	/tareas/{id}/duplicar	Duplicar

Incluye endpoints adicionales para la asignación de empleados a tareas si aplica.

- Gestión del Acceso por QR

Método	Endpoint	Descripción
GET	/empleados/{id}/qr-info	Obtener QR
POST	/empleados/generar-qr-preview	Generar QR previo

POST	/empleados/{id}/enviar-whatsapp	Envío por WhatsApp
GET	/empleados/{id}/generar-pdf	PDF con QR

7.2. EndPoints del Módulo de Empleado

URL Base: /empleado/*

Corresponden a las rutas accesibles únicamente por usuarios con rol Empleado y se centran en su información propia y el registro horario.

- **Perfil del Empleado**

Método	Endpoint	Descripción
GET	/perfil/{id}	Ver el perfil del empleado

- **Registro Horario**

Método	Endpoint	Descripción
POST	/registro/{id}/start	Inicio
POST	/registro/{id}/pause	Pausa
POST	/registro/{id}/stop	Fin
GET	/registro/{id}/estado	Estado actual
GET	/registro/{id}/historial	Historial
GET	/registro/{id}/datatable	Tabla AJAX
GET	/registro/{id}/resumen-periodo	Resumen por período
GET	/registro/{id}/progreso-semanal	Progreso semanal

- **Tareas Asignadas**

Método	Endpoint	Descripción
POST	/{id}/tareas/crear	Crear
GET	/{empleadoId}/tareas/{tareaId}/detalles	Detalles
POST	/{empleadoId}/tareas/{tareaId}/estado	Cambiar estado
PUT	/{empleado}/tareas/{tarea}/actualizar	Actualizar
DELETE	/tareas/{tarea}	Eliminar
GET	/{id}/tareas/datatable	Datatable
GET	/{id}/tareas/estadisticas	Estadísticas
GET	/tipos-tarea	Tipos de tarea accesibles

7.3. Endpoints de Autenticación y Acceso

Estos endpoints no dependen del rol del usuario y permiten el acceso inicial al sistema mediante credenciales o QR.

- **Autenticación Tradicional**

URL Base: /login

Método	Endpoint	Descripción
GET	/login	Mostrar formulario de inicio
POST	/login	Autenticar usuario
GET	/logout	Cerrar sesión

- **Acceso mediante QR**

Método	Endpoint	Descripción
GET	/login_qr	Vista de login QR
POST	/generate-qr-token	Generar token
GET	/check-qr-login/{token}	Consultar token
POST	/confirm-qr-login/{token}	Confirmar
GET	/qr-scanner	Vista escáner
POST	/process-qr-scan	Procesar escaneo
GET	/qr-info/{id}	Info QR
POST	/cleanup-expired-tokens	Limpieza (solo admin)
GET	/qr-login-process	Redirección de login QR

7.4. Seguridad de los EndPoints

Todos los endpoints del sistema están protegidos por:

- Middleware auth: exige sesión activa.
- Middleware de roles (admin, empleado): diferencia permisos.
- Protección CSRF en rutas tipo POST/PUT/PATCH/DELETE.
- Restricción de acceso a través de controladores (doble validación backend).
- Validación de datos mediante Request personalizados si aplica.

Los endpoints administrativos no pueden ser invocados por empleados, y viceversa.

8. Flujos de Trabajo y Procesos de Negocio

Este apartado describe los procesos de negocio principales implementados en la plataforma, así como los flujos de trabajo que siguen los diferentes actores del sistema. El diseño se basa en una estructura funcional que involucra dos roles: **Administrador** y **Empleado**, cada uno con responsabilidades específicas dentro del ciclo operativo de la aplicación.

Los flujos de trabajo han sido modelados para optimizar la gestión de personal, la asignación de tareas y el registro horario, garantizando trazabilidad, eficiencia y cumplimiento normativo.

8.1. Flujo de Trabajo General del Sistema

El funcionamiento global del sistema sigue las siguientes etapas:

1. El usuario accede al sistema mediante login tradicional o autenticación por QR.
2. El sistema identifica el rol del usuario.
3. Según el rol, se redirige a:
    - Panel de Administrador
    - Perfil del Empleado
4. Se ejecutan los procesos correspondientes:
    - Gestión y administración del personal (rol Administrador).
    - Registro de actividad laboral y consulta de tareas (rol Empleado).
5. Los datos se almacenan en la base de datos para su posterior análisis o exportación.

8.2. Proceso de Autenticación y Acceso

- Autenticación por credenciales (usename y contraseña)
    1. El usuario accede a la pantalla de login.
    2. Introduce sus credenciales.
    3. El sistema valida que el usuario exista y la contraseña sea correcta.
    4. Se identifica el rol:
        - Si es Administrador → se redirige al panel /admin/.
        - Si es Empleado → se redirige al perfil /empleado/perfil/id.
    5. En caso de error se muestra mensaje correspondiente.

- Autenticación mediante QR
    1. El usuario abre la pantalla de acceso por QR.
    2. Escanea el código QR asociado a su cuenta.
    3. El sistema verifica:
        - Que el token QR exista.
        - Que esté activo.

        - Que no haya expirado.
    4. Al validarse el token, el usuario accede automáticamente sin introducir contraseña.
    5. El sistema redirige al panel correspondiente según el rol.

Este proceso agiliza el acceso, especialmente para empleados.

8.3. Proceso de Gestión de Empleados (Administrador)

1. El Administrador accede a /admin/empleados.
2. Puede realizar acciones como:
    - Crear nuevos empleados
    - Editar datos personales
    - Activar o desactivar empleados
    - Eliminar registros
3. El sistema valida los datos introducidos.
4. Se actualiza el registro en la base de datos.
5. Opcionalmente, se generan credenciales o códigos QR asociados al empleado.

El flujo garantiza que la información del personal esté siempre actualizada.

8.4. Proceso de Gestión de Tareas (Administrador)

1. El Administrador accede al módulo /admin/tareas.
2. Puede crear, editar o eliminar tareas.
3. Puede asignarlas a uno o varios empleados.
4. Los empleados reciben automáticamente la tarea en su panel.
5. El Administrador puede consultar:
    - Estado de cada tarea (pendiente, en proceso, completada).
    - Tiempo dedicado por empleado (según registros).
6. El sistema almacena cada cambio para mantener un historial.

El proceso permite controlar la carga laboral y supervisar el progreso.

8.5. Proceso de Registro Horario (Empleado)

1. El empleado accede a su perfil

    - Ruta: /empleado/perfil/{id}
    - El sistema consulta si existe un registro horario activo o pausado.
    - El panel muestra:

- Botón Iniciar Jornada si no hay registro
- Botón Pausar si hay jornada en curso
- Botón Reanudar si está pausada
- Botón Finalizar si está activa o pausada

2. El empleado inicia la jornada (Start)

   Acción: Clic en "Iniciar Jornada"
   El sistema:

   - Verifica que no exista otro registro abierto.
   - Crea un nuevo registro con:
     1. hora_inicio
     2. estado = activo
     3. tiempo_acumulado = 0
   - Muestra el cronómetro en marcha.

3. El empleado pausa la jornada (Pause)

   Acción: Clic en "Pausar"

   El sistema:

   1. Verifica que el estado actual sea activo.
   2. Calcula el tiempo trabajado hasta el momento.
   3. Cambia el estado a pausado.
   4. Detiene el cronómetro en pantalla.
   5. Habilita el botón "Reanudar".

4. El empleado reanuda la jornada (Resume)

   Acción: Clic en "Reanudar"

   El sistema:

   1. Detecta que existe un registro pausado, no crea uno nuevo.
   2. Cambia el estado nuevamente a activo.
   3. Reactiva el cronómetro.
   4. Continúa el conteo del tiempo.

5. El empleado finaliza la jornada (Stop)

   Acción: Clic en "Finalizar Jornada"

El sistema:

- Verifica que haya un registro en estado activo o pausado.
- Calcula el tiempo total de la jornada:
  - tiempo previo acumulado
  - diferencia desde la última reanudación
- Establece:
  - hora_fin
  - estado = finalizado
- Bloquea el registro para impedir nuevas modificaciones.
- El cronómetro desaparece y la vista vuelve a estado “sin registro activo”.

6. El empleado consulta el estado actual del registro

El sistema devuelve:

- Estado (activo / pausado / inactivo)
- Tiempos parciales
- ID del registro activo
- Si puede iniciar o debe finalizar

Esto permite restaurar el cronómetro al recargar la página.

8.6. Proceso de Generación e Informes y Exportaciones (Administrador)

1. El Administrador accede al módulo de exportación.
2. Selecciona el tipo de informe:
   - Informe mensual de empleados
   - Informe individual de registro horario
   - Exportación PDF del listado de empleados
3. El sistema ejecuta la clase correspondiente del módulo /app/Exports.
4. Genera un archivo PDF o Excel.
5. El archivo se descarga automáticamente o queda disponible en el sistema.

Este proceso permite documentar el cumplimiento laboral.

8.7. Proceso de Gestión del Acceso por QR

1. El Administrador genera un QR desde el panel.
2. El sistema crea un token único asociado al empleado.
3. El empleado puede escanearlo para:
   - Acceder al sistema
4. Si el Administrador desactiva un QR:

- El token deja de ser válido
- El acceso vía QR queda denegado

Este flujo mejora la seguridad y facilita el fichaje.

8.8. Interacción entre Roles

El sistema ha sido diseñado para que ambos roles colaboren en un ciclo operativo coordinado:

- El Administrador gestiona empleados, tareas, registros e informes.
- El Empleado ejecuta las tareas asignadas y registra su actividad.
- El sistema almacena toda la información y la estructura para consultas posteriores.

Esta dinámica garantiza eficiencia y transparencia.

9. Interfaz de Usuario

La interfaz de usuario (UI) del sistema ha sido diseñada siguiendo los principios de claridad, simplicidad y accesibilidad, con el objetivo de ofrecer una experiencia agradable para ambos tipos de usuario: Administrador y Empleado. La plataforma utiliza tecnologías web estándar como HTML5, CSS3, JavaScript, junto con el framework Bootstrap, lo que permite un diseño responsivo, consistente y adaptado a cualquier dispositivo —ordenadores, tablets y teléfonos móviles—.

El uso de Laravel Blade como motor de plantillas facilita la organización del contenido y la reutilización de componentes visuales, garantizando una experiencia homogénea en toda la aplicación.

9.1. Diseño General de la Interfaz

El diseño de la aplicación sigue una estructura coherente en todas las vistas:

- Barra de navegación fija en la zona superior, según el rol del usuario, junto con panel sen la parte superior en el módulo de administración para el acceso rápido a las diferentes secciones.
- Contenido central con la información principal de cada módulo.
- Uso de tablas dinámicas para representar empleados, tareas y registros.
- Botones de acción diferenciados por color para mejorar la claridad:
  - Verde → Acciones positivas (crear, guardar)
  - Azul → Ver o Detalles
  - Amarillo → Editar
  - Rojo → Eliminar o desactivar

El diseño prioriza la legibilidad, minimiza la carga visual y garantiza que el usuario encuentre rápidamente las opciones necesarias.

9.2. Interfaz del Administrador

La interfaz del Administrador se centra en la gestión y supervisión del sistema. Sus características principales incluyen:

- Panel de navegación superior

  Incluye enlaces rápidos a:

  - Gestión de empleados
  - Gestión de tareas
  - Informes y exportaciones

  Esta estructura facilita el trabajo diario del administrador y reduce el tiempo de navegación.

- Tablas de datos optimizadas

  Las vistas del Administrador utilizan tablas responsivas con:
  - Búsqueda interna
  - Ordenamiento por columnas
  - Paginación
  - Acciones rápidas (editar, ver, eliminar)

  Esto permite gestionar grandes volúmenes de información de forma eficiente.

- Formularios claros y validados

  Para crear o editar empleados y tareas, los formularios incluyen:
  - Validaciones visuales
  - Mensajes de error y confirmación
  - Estructura organizada en bloques

9.3. Interfaz del Empleado

La interfaz del empleado está diseñada para ser rápida, simple e intuitiva, favoreciendo su uso diario.

- Vista del Perfil

Incluye:

- Nombre y datos personales
- Botones de registrar entrada y salida
- Listado de tareas asignadas
- Historial de registros

Todo se presenta de forma clara para evitar errores y acelerar los procesos de registro.

- Registro de tiempo

  El empleado puede registrar:

  - Inicio de jornada
  - Fin de jornada
  - Inicio y fin de tiempo por tarea

  Los botones están diseñados con colores llamativos y un tamaño adecuado para su uso desde móvil.

- Lista de tareas asignadas

  Muestra:

  - Nombre de la tarea
  - Estado
  - Breve descripción
  - Botón de completar tarea

  El diseño prioriza la simplicidad y la movilidad.

10. Seguridad y Protección de Datos

La seguridad y la protección de datos constituyen un aspecto fundamental en el desarrollo del sistema, especialmente debido a que la aplicación gestiona información sensible relacionada con empleados, horarios de trabajo, credenciales de acceso, códigos QR y otros datos de carácter personal.

El proyecto está desarrollado utilizando Laravel, un framework que integra por defecto múltiples mecanismos de seguridad. A esto se han añadido prácticas específicas para garantizar un tratamiento adecuado de los datos, de acuerdo con las recomendaciones del RGPD (Reglamento General de Protección de Datos) y buenas prácticas de la ingeniería de software.

10.1. Autenticación y Control de Acceso

El sistema implementa un mecanismo de autenticación robusto basado en:

- Login con username y contraseña cifrada
- Autenticación mediante token QR
- Gestión de sesiones seguras
- Control de acceso por roles (RBAC)

Laravel utiliza por defecto:

- Hash BCrypt para almacenar contraseñas de manera irreversible
- Tokens CSRF para proteger formularios
- Middleware auth para evitar accesos no autorizados
- Middleware de roles para diferenciar permisos entre Administrador y Empleado

De este modo, ninguna zona del sistema queda expuesta sin la validación correspondiente.

10.2. Gestión Segura de Contraseñas

Las contraseñas de los usuarios se almacenan utilizando:

- Hashing seguro mediante bcrypt
- Prohibición de contraseñas en texto plano
- Validaciones de complejidad mínima

El sistema evita mostrar las contraseñas en cualquier interfaz y no permite su recuperación directa, únicamente la regeneración o cambio por parte del usuario.

10.3. Protección mediante CSRF, XXS y SQL Injection

Laravel integra medidas para mitigar ataques comunes:

- CSRF (Cross-Site Request Forgery)
    - Todos los formularios incluyen el token CSRF generado automáticamente por Laravel, evitando envíos fraudulentos.

- XSS (Cross-Site Scripting)
    - Las vistas Blade escapan contenido por defecto ({{ }}).
    - Se evita la ejecución de scripts maliciosos en entradas de usuario.
    - Solo se permite HTML cuando es estrictamente necesario.

- SQL Injection
    - Todas las consultas se gestionan mediante Eloquent ORM, que parametriza internamente las consultas.

- No se ejecutan consultas SQL directas sin validación previa.

10.4. Seguridad de los Códigos QR

Los QR generan un token único para validar al usuario:

- Los tokens no contienen información personal visible.
- El código QR no es un identificador directo, sino un token que se valida en el servidor.
- Pueden desactivarse o regenerarse en cualquier momento por el Administrador.
- No permiten acceso a otras partes del sistema sin pasar por validaciones adicionales.

Esto impide que un tercero obtenga información simplemente escaneando el código.

10.5. Seguridad en el Transporte de Datos

El sistema se encuentra alojado en un entorno con:

- Certificado SSL (HTTPS) para cifrar todas las comunicaciones.
- Protección frente a escuchas y ataques de intermediario (MITM).

Todo dato enviado entre cliente y servidor viaja cifrado.

10.6. Seguridad de Exportaciones (PDF y Excel)

Los archivos generados pueden contener datos sensibles.

Medidas aplicadas:

- Solo el Administrador tiene acceso a exportaciones.
- La URL de exportación está protegida por middleware.
- Los archivos se generan temporalmente y no quedan expuestos públicamente.

10.7. Seguridad en el Servidor

El entorno de despliegue incluye:

- Configuración reforzada de permisos de archivos
- Uso de .env protegido sin acceso público
- Variables sensibles no expuestas en vistas
- Control de errores sin mostrar información del sistema
- Limitación de rutas críticas

Además, Laravel incluye un filtro de excepciones que evita revelar stack traces en producción.

10.8. Copias de Seguridad y Recuperación

El sistema puede integrarse con:

- Backups automáticos del hosting
- Exportaciones manuales de datos
- Versionado del código mediante Git

Esto garantiza disponibilidad e integridad ante fallos.

11. Pruebas y Calidad del Software

Para asegurar la fiabilidad, estabilidad y buen funcionamiento del sistema, se llevó a cabo un plan de pruebas exhaustivo que abarca tanto pruebas funcionales como técnicas. El objetivo principal fue garantizar que todas las funcionalidades del proyecto cumplieran con los requisitos definidos, que la experiencia de usuario fuera adecuada y que no existieran fallos críticos que afectasen al uso diario de la plataforma.

Las pruebas se realizaron sobre el entorno de desarrollo y posteriormente en el servidor de producción, utilizando herramientas propias de Laravel, inspección manual, pruebas de caja negra y validaciones automáticas.

11.1. Estrategia de Pruebas

El proceso de verificación se realizó siguiendo estos pilares:

- Pruebas funcionales: verificar que cada módulo funciona como se espera.
- Pruebas de interfaz (UI/UX): evaluar la correcta visualización y navegación.
- Pruebas de seguridad: comprobar protección frente a accesos no autorizados.
- Pruebas de rendimiento básico: validar que el sistema responde adecuadamente.
- Pruebas de integración: asegurar la correcta comunicación entre módulos.
- Pruebas de regresión: tras incorporar nuevas funcionalidades.
- Pruebas en diferentes dispositivos: ordenadores, tablets y móviles.

11.2. Pruebas Funcionales

Estas pruebas verificaron que todos los módulos cumplían su objetivo principal:

- Módulo de autenticación

Inicio de sesión con credenciales válidas
Manejo de errores con credenciales incorrectas
Redirección según el rol (Administrador/Empleado)
Acceso mediante código QR
Cierre de sesión y eliminación de la sesión activa

- Gestión de empleados

    - Creación de un nuevo empleado
    - Edición de datos existentes
    - Activación/desactivación
    - Eliminación de empleados sin registros asociados
    - Visualización del perfil individual
    - Validaciones de campos requeridos

- Gestión de tareas

    - Creación de tareas
    - Edición y actualización
    - Asignación a empleados
    - Eliminación de tareas
    - Visualización del estado y seguimiento

- Registro de tiempo

    - Inicio de jornada
    - Registro de tareas trabajadas
    - Finalización de jornada
    - Cálculo de minutos trabajados
    - Relación correcta entre empleado, tarea y registro

. - Exportaciones

    - Generación de PDF de empleados
    - Exportación Excel mensual
    - Informe individual de horarios
    - Verificación del formato y contenido

Todas las pruebas funcionales fueron superadas, garantizando que el sistema actúa de acuerdo con su especificación.

11.3. Pruebas de Interfaz y Experiencia de Usuario

Se revisaron aspectos visuales y de usabilidad:

- Correcto funcionamiento de tablas, formularios y botones
- Responsividad en diferentes tamaños de pantalla
- Revisión de colores, iconos y elementos interactivos
- Reducción de elementos innecesarios para empleados
- Modales, alertas y mensajes de confirmación claros

Las pruebas se hicieron tanto manualmente como con herramientas de inspección del navegador.

11.4. Pruebas de Seguridad

Dado que el sistema maneja datos personales y fichajes laborales, se evaluaron aspectos clave de seguridad.

Las pruebas incluyeron:

- Acceso a rutas protegidas sin autenticación
- Acceso a rutas de administrador con rol de empleado
- Intentos de manipulación de formularios sin token CSRF
- Intentos de inyección SQL (mitigados por Eloquent)
- Validación de tokens QR inválidos o expirados
- Comprobación de hashes de contraseña en base de datos

Resultados:

✓ Todas las rutas rechazaron accesos no autorizados
✓ Los formularios respondieron correctamente ante ataques comunes
✓ Los tokens QR fueron invalidados correctamente cuando procedía

11.5. Pruebas de Rendimiento y Carga Básica

Aunque el proyecto no requiere un rendimiento extremo, se hicieron pruebas para asegurar fluidez y estabilidad:

- Respuesta del servidor bajo consultas repetidas
- Carga de tablas con grandes cantidades de empleados/tareas

- Verificación de tiempos de exportación
- Evaluación del tiempo de carga de páginas críticas

Conclusión:

✓ El sistema responde adecuadamente bajo carga moderada
✓ Las exportaciones funcionan sin saturar el servidor
✓ El panel del empleado carga de forma casi instantánea

11.6. Pruebas de Integración

Estas pruebas verificaron que los módulos interactúan de forma consistente:

- Relación entre empleados ↔ tareas
- Relación entre tareas ↔ registros de tiempo
- Relación entre empleados ↔ QR
- Sincronización entre vistas y controlador
- Exportaciones basadas en datos reales

✓ No se detectaron errores en la comunicación entre módulos.

11.7. Pruebas de Regresión

Cada vez que se añadió una funcionalidad nueva (por ejemplo, exportaciones, control horario, o QR), se revisó:

- Que el login siguiera funcionando
- Que el dashboard cargara correctamente
- Que la edición de empleados no produjera fallos
- Que las tablas y formularios no mostraran errores

✓ Todas las funcionalidades existentes siguieron funcionando tras los cambios.

11.8. Documentación y Registro de Errores

Durante las pruebas se registraron:

- Pequeños errores visuales
- Ajustes necesarios en formularios
- Validaciones faltantes
- Comportamientos inesperados menores

Todos ellos fueron corregidos antes de la publicación del sistema.

12. Despliegue y Mantenimiento

El despliegue y mantenimiento del sistema son procesos esenciales para garantizar la disponibilidad, estabilidad y continuidad operativa de la plataforma. Este apartado detalla los procedimientos seguidos para la puesta en producción de la aplicación, así como las tareas planificadas para asegurar su funcionamiento correcto a largo plazo.

El sistema está desarrollado en Laravel y utiliza MySQL como base de datos, lo que permite un proceso de despliegue estandarizado y compatible con la mayoría de los proveedores de hosting.

12.1. Preparación del Entorno de Producción

Para el despliegue de la aplicación se preparó un servidor web con las siguientes características:

- PHP 8.x o superior
- Extensiones necesarias para Laravel (pdo_mysql, openssl, mbstring, tokenizer, xml, ctype, json)
- Servidor web Apache/Nginx
- MySQL 5.7+ o MariaDB 10+
- Certificado SSL activo (HTTPS)
- Composer instalado para gestionar dependencias

Se creó un entorno aislado con usuario propio en el hosting para asegurar la integridad de los archivos y evitar interferencias con otras aplicaciones.

12.2. Proceso de Despliegue

El despliegue se realizó siguiendo las buenas prácticas del ecosistema Laravel:

- Subida de archivos

  El proyecto se subió mediante:
  - FTP/SFTP
  - o repositorio Git (si el hosting lo permite)

  Se incluyeron todas las carpetas excepto aquellas excluidas en .gitignore.

- Instalación de dependencias

Una vez subido el código:

composer install --no-dev

Esto genera las dependencias necesarias para el entorno de producción.

- Configuración del archivo .env

Se configuraron las variables críticas:

- Credenciales de base de datos
- URL pública
- Clave de aplicación (APP_KEY)
- Modo de entorno (APP_ENV=production)
- Ajustes de correo (si se usa)
- Configuración de almacenamiento

Después, se generó la clave de Laravel:

php artisan key:generate

- Migraciones y población inicial

Para crear las tablas:

php artisan migrate

Y si existía una carga inicial:

php artisan db:seed

- Configuración de permisos

El hosting requiere dar permisos a:

- /storage
- /bootstrap/cache

De esta forma Laravel puede escribir logs, cachés y archivos temporales.

- Configuración de dominio y certificado SSL

El dominio principal fue configurado para apuntar a la carpeta /public del proyecto. Adicionalmente, se instaló un certificado SSL proporcionando conexión segura (https).

12.3. Gestión y Mantenimiento del Sistema

Una aplicación en producción requiere tareas regulares para garantizar su estabilidad y rendimiento. En este proyecto se contemplaron las siguientes áreas:

- Mantenimiento Correctivo

Corrección de errores detectados durante el uso del sistema:

- Ajustes en formularios o validaciones
- Corrección de fallos lógicos en controladores
- Mejora de consultas lentas
- Depuración de componentes del frontend

Los errores se rastrean gracias a los logs automáticos de Laravel (storage/logs/laravel.log).

- Mantenimiento Evolutivo

Incluye mejoras o nuevas funcionalidades solicitadas por el cliente o detectadas durante el uso:

- Nuevos tipos de informes
- Mejora del módulo de tareas
- Optimización del registro horario
- Ampliación de la API interna
- Integración con herramientas externas

Se priorizan aquellas mejoras que ofrecen más valor al usuario.

- Mantenimiento Preventivo

Tareas periódicas para evitar fallos futuros:

- Actualización del framework Laravel
- Actualización de dependencias con composer update
- Optimización de índices en la base de datos
- Revisión de permisos y seguridad
- Limpieza de archivos temporales mediante comandos como:

  ```
  php artisan cache:clear
  php artisan config:clear
  php artisan route:clear
  ```

Estas acciones mejoran el rendimiento y la seguridad.

12.4. Actualizaciones y Ciclo de Vida del Software

El sistema está preparado para mantenerse a largo plazo mediante:

- Versionado del código (Git)
- Control de ramas para evitar errores en producción
- Buenas prácticas de despliegue continuo (cuando sea posible)
- Documentación técnica actualizada
- Pruebas de regresión antes de cada actualización

Esto asegura que las nuevas versiones no afecten al funcionamiento existente.

13. Planificación y Metodología de Desarrollo

La planificación y metodología de desarrollo constituyen un elemento fundamental para garantizar que el proyecto se complete siguiendo un proceso estructurado, eficiente y orientado a resultados. Con el fin de organizar el trabajo de forma clara y controlar el avance del proyecto, se ha optado por una metodología ágil adaptada (Scrum/Kanban) que permite una evolución incremental del sistema y una rápida incorporación de mejoras.
El desarrollo del proyecto se dividió en fases bien definidas, abordando cada módulo de forma iterativa, integrando pruebas y revisiones continuas para asegurar la calidad del producto final.

13.1. Metodología utilizada

Para el desarrollo del sistema se siguió una metodología ágil, tomando como referencia elementos de Scrum:

- Trabajo dividido en iteraciones cortas (sprints).
- Reuniones periódicas de revisión (a nivel individual o del equipo).
- Desarrollo incremental: primero funcionalidades mínimas, después mejoras.
- Flexibilidad para modificar requisitos durante el proyecto.
- Priorización basada en valor para el usuario final.

El uso de metodologías ágiles se adapta a proyectos web con requerimientos cambiantes y con retroalimentación continua durante la implementación.

13.2. Fases del Proyecto

El proyecto se estructuró en varias fases principales, ordenadas de manera lógica y secuencial:

- Fase 1: Análisis de Requisitos

Incluyó:

- Identificación de necesidades del Administrador.
- Definición de funcionalidades para empleados.
- Revisión de requisitos legales (control horario).
- Identificación de datos necesarios para almacenar.
- Priorización de módulos esenciales.

Resultado: un documento de requisitos funcionales y no funcionales.

- Fase 2: Diseño del Sistema

En esta fase se definieron:

- Arquitectura del sistema (Laravel MVC).
- Diseño de la base de datos.
- Modelo entidad–relación.
- Estructura de rutas y controladores.
- Esquema de vistas para administración y empleados.

Se crearon prototipos básicos de la interfaz con referencias visuales de Bootstrap.

- Fase 3: Implementación

Es la fase de desarrollo del código, dividida en módulos:

1. Módulo de autenticación (login y QR).
2. Módulo de empleados (CRUD completo).
3. Módulo de tareas (CREAR/EDITAR/ASIGNAR).
4. Módulo de registro horario (entrada/salida/tareas).
5. Exportaciones PDF y Excel.
6. Integración de roles y permisos.
7. Optimización del panel del empleado.
8. Dashboard del administrador.

Cada módulo fue implementado siguiendo buenas prácticas de Laravel, Blade y Eloquent.

- Fase 4: Pruebas y Validación

Se realizaron pruebas:

- Funcionales (cada módulo)
- De seguridad

- De rendimiento básico
- De regresión
- De interfaz
- En dispositivos móviles

Esta fase permitió identificar y corregir errores antes del despliegue.

- Fase 5: Despliegue en Producción

Incluyó:

- Configuración del servidor
- Subida de código y dependencias
- Configuración del entorno .env
- Migraciones de la base de datos
- Activación de SSL y optimizaciones
- Comprobación final de funcionamiento

- Fase 6: Mantenimiento y Evolución

Tras el despliegue inicial:

- Corrección de incidencias menores
- Ajuste de funciones según feedback
- Mejoras en rendimiento
- Actualización de dependencias
- Preparación de futuras ampliaciones

13.3. Herramientas de Gestión y Desarrollo

Durante el proyecto se utilizaron las siguientes herramientas:

- Laravel para el framework del backend
- PHP como lenguaje principal
- MySQL para la base de datos
- Bootstrap para el diseño responsive
- Git (si se utilizó control de versiones)
- Composer para las dependencias
- Hosting con SSL para producción
- Inspección del navegador para depuración frontend
- phpMyAdmin para gestión de la base de datos

Estas herramientas facilitaron la organización del trabajo y la calidad del desarrollo.

13.4. Gestión de Tareas Durante el Desarrollo

Se siguió una organización basada en:

- Lista de pendientes (backlog)
- Tareas activas (en desarrollo)
- Tareas completadas

Este flujo tipo Kanban permitió visualizar el estado de cada actividad y priorizar las más relevantes.

14. Conclusiones y Recomendaciones

14.1. Conclusiones

El desarrollo de este sistema de gestión de empleados, tareas y control horario ha permitido alcanzar los objetivos planteados al inicio del proyecto, demostrando que las tecnologías utilizadas (Laravel, PHP, MySQL y Bootstrap) ofrecen una solución sólida, escalable y plenamente funcional para un entorno laboral real.

Los principales logros del proyecto pueden resumirse en:

✓ Cumplimiento de los objetivos funcionales

Se implementaron todas las funcionalidades esenciales:

- Gestión completa de empleados
- Gestión y asignación de tareas
- Registro horario detallado
- Inicio y fin de jornada
- Registro de tiempo por tarea
- Acceso mediante credenciales y mediante QR
- Exportaciones en PDF y Excel
- Roles diferenciados (Administrador y Empleado)

El sistema cubre plenamente las necesidades para las que fue diseñado.

✓ Arquitectura clara y mantenible

El uso de Laravel MVC permitió:

- Separación correcta entre lógica, datos y vistas
- Código limpio, modular y escalable
- Facilidad para añadir nuevas funcionalidades en el futuro
- Seguridad integrada y soporte a largo plazo

✓ Interfaz accesible y centrada en el usuario

La UI basada en Bootstrap ofrece:

- Facilidad de uso para empleados
- Panel de administración organizado y profesional
- Adaptación completa a dispositivos móviles

La experiencia de usuario es fluida, clara y sin barreras.

✓ Seguridad y protección de datos garantizada

El sistema cumple con:

- Control de acceso mediante roles
- Tokens CSRF
- Cifrado de contraseñas
- Protección ante ataques comunes (SQLi, XSS, CSRF)
- Uso de tokens QR seguros
- Comunicación cifrada mediante HTTPS

Se asegura así la integridad, disponibilidad y confidencialidad de la información gestionada.

✓ Fiabilidad y calidad del software

Tras el proceso de pruebas, se demostró que:

- El sistema es estable
- No existen fallos críticos
- Todas las interacciones generan respuestas correctas
- El rendimiento es adecuado para el entorno objetivo

✓ Viabilidad económica

El proyecto es económicamente viable gracias a:

- Uso de tecnologías open source
- Coste reducido de hosting
- Mantenimiento asumible
- Escalabilidad sin necesidad de licencias

14.2. Recomendaciones

Aunque el proyecto se encuentra en estado plenamente operativo, se han identificado diversas mejoras y evoluciones que podrían implementarse para aumentar su potencial:

1. Incorporar notificaciones automáticas

Mediante email o notificaciones internas para:

- Nuevas tareas asignadas
- Tareas próximas a vencerse
- Resúmenes mensuales para el administrador

2. Crear un módulo avanzado de estadísticas

Con gráficos y dashboards que muestren:

- Tiempos trabajados por empleado
- Rendimiento por tarea
- Comparativas mensuales
- Horas extra, ausencias, etc.

3. Implementar una API REST oficial

Para permitir:

- Integración con apps móviles
- Conexión con software externo de RRHH
- Sincronización con calendarios o sistemas de fichaje

4. Añadir autenticación de dos factores (2FA)

Especialmente útil para administradores y accesos sensibles.

5. Mejorar el sistema de permisos

Permitiendo:

- Permisos específicos (por ejemplo, solo ver, solo editar, etc.)
- Más roles (supervisor, recursos humanos…)

6. Añadir un módulo de comunicación interna

Mensajes entre empleados/administradores o comentarios en tareas.

7. Optimización para grandes volúmenes de datos

Si el sistema crece, sería recomendable:

- Añadir índices adicionales en la base de datos
- Implementar paginaciones avanzadas
- Usar caché para consultas frecuentes

8. Copias de seguridad automáticas

Integrar el sistema con:

- Backups automáticos diarios
- Notificaciones de verificación
- Exportación incremental

9. Mejoras en la interfaz

Aunque la UI es clara y funcional, se podrían añadir:

- Temas personalizables (modo oscuro)
- Menús dinámicos más modernos
- Animaciones y microinteracciones

10. Desarrollo de una aplicación móvil

Para empleados, facilitando:

- Registro de entrada/salida
- Gestión de tareas
- Escaneo más fluido del QR