# *AsistGuard:* Control de asistencias y gestión de guardias del profesorado


Alumno: Adrian Pascual Marschal
Tutor: Miguel Angel Paris Peñaranda
2CFGS
2024/2025

# Índice de contenidos

# Introducción

*AsistGuard* es una plataforma que permite gestionar de manera eficiente el control de asistencia y las guardias del profesorado en instituciones educativas. Un aspecto clave del *software* es que permite automatizar el registro del tiempo trabajado por el docente y el correcto cambio de profesor en caso de ausencia.

Controla la entrada y salida de los docentes, la consulta de horarios, la gestión de ausencias y la asignación de guardias de forma rápida y confiable. Además, posee un registro que permite conocer las guardias cumplidas y el total de horas de trabajo de cada docente. Todo esto queda almacenado en una base de datos que sólo puede ser consultada por usuarios autentificados, lo que permite asegurar el seguimiento y la privacidad de la información.

# Módulos implicados en el desarrollo

En el lado del cliente, se ha utilizado *HTML5, CSS3* y *JavaScript* para construir una interfaz moderna, responsiva y accesible desde cualquier dispositivo. Todos los formularios han sido diseñados con medidas de seguridad para evitar la inyección de código y otros tipos de ataques comunes.

El sistema cuenta con un módulo de autenticación basado en credenciales personales del profesorado. Cada usuario tiene permisos específicos según su rol, los cuales son gestionados por el administrador del centro. Esta autenticación asegura que solo usuarios autorizados puedan acceder a las distintas funcionalidades del sistema.

Toda la información se almacena en una base de datos relacional gestionada con *MySQL*, donde se registran tanto los datos de jornada como las guardias asignadas y realizadas.

La comunicación entre el cliente y el servidor se lleva a cabo mediante una *API REST* desarrollada por nuestro equipo. Esta *API* permite un intercambio estructurado y eficiente de datos, facilitando la escalabilidad y el mantenimiento del sistema. Para el desarrollo del servidor se había planteado el uso del *framework Laravel* pero debido a un análisis sobre las diferentes funcionalidades, hemos decidido basarnos en abstenernos de él pero aun así seguir cumpliendo una arquitectura *REST*.

# Objetivos a cumplir

El objetivo general es ofrecer una solución digital que facilite y automatice el control de asistencia del profesorado y la gestión de guardias en centros educativos, mejorando la organización interna. Nuestro objetivo es otorgar una aplicación web segura, accesible y rápida. Debido a esto podemos gestionar de manera eficiente los diferentes apartados:

1. Optimizar la gestión de horarios
– Facilitar al profesorado la consulta y administración de su jornada laboral.

2. Mejorar la asignación de guardias
– Automatizar y visualizar en tiempo real la cobertura de turnos.

3. Agilizar el control de ausencias
– Permitir al personal autorizado registrar y tramitar bajas y permisos con rapidez.

4. Proporcionar información fiable
– Generar informes detallados de asistencia, guardias y ausencias según criterios de fecha, docente o periodo.

5. Garantizar la seguridad y el acceso controlado
– Asegurar la autenticación y la protección de datos según el rol de cada usuario.

6. Asegurar la disponibilidad multiplataforma
– Ofrecer acceso desde ordenadores y dispositivos móviles con una interfaz *responsive*.

7. Facilitar el mantenimiento y la escalabilidad
– Diseñar una arquitectura modular que permita incorporar nuevas funcionalidades sin afectar al sistema existente.

# Tipo de proyecto

*AsistGuard* esta destinada a automatizar tareas administrativas en instituciones educativas, particularmente en las áreas de asistencia de profesores y sistemas automatizados de gestión de sustituciones.

Está diseñado para funcionar desde cualquier ubicación en el mundo siempre que haya acceso a internet y no requiere instaladores ni configuraciones elaboradas por parte del usuario final.

Este tipo de proyecto reúne elementos como diseño de bases de datos, programación de *backend* y *frontend,* seguridad de aplicaciones web, gestión de usuarios y se centra en resolver un problema real y específico en la educación.

La tecnología subyacente permite a las instituciones integrar *AsistGuard* como un servicio digital de manera fluida, lo que lo convierte en una solución rentable, fácil de gestionar y fiable para una amplia variedad de usuarios.

# Ideas determinantes durante la creación

El desarrollo se ha llevado a cabo en fases bien definidas, siguiendo un enfoque iterativo que ha permitido validar cada funcionalidad de manera progresiva y garantizar la estabilidad del sistema. Desde el principio, se ha priorizado la usabilidad, y, sobretodo la seguridad que garantiza al profesorado el uso de su documento personal.

Además, se ha integrado un sistema de autenticación con control de acceso por roles (administrador, profesorado), lo que permite restringir el uso de ciertas funcionalidades según los permisos asignados.

Todas las acciones importantes quedan registradas en la base de datos, asegurando la trazabilidad del uso del sistema. La estructura modular de la aplicación permite futuras ampliaciones o integraciones sin afectar el funcionamiento general.

# Estudio del sistema actual

En bastantes instituciones educativas, el registro de la presencia de los docentes y la organización de las sustituciones todavía se hacen a mano o con herramientas medio digitales, como plantillas de cálculo o apuntes en papel. De esta forma no se puede actuar con rapidez por la falta de comunicación tanto del profesor que cubre la guardia como del profesor que se ausenta. Esto supone la falta de registro de los cambios que se realizan en esas sesiones. Incluye también el riesgo de cometer varios errores de información. Estos métodos, aunque puedan servir en lugares pequeños, no son suficientes para responder a las exigencias de hoy en día en cuanto a rapidez, seguimiento y claridad en la gestión interna.

# Funcionalidad del sistema a partir de los objetivos a lograr

Tras sopesar los recursos al alcance y las verdaderas carencias de un colegio, se ha llegado a la conclusión de que instalar e utilizar *AsistGuard* es factible al 100%. Esto se apoya en varios puntos clave, entre ellos, que casi todos los colegios ya cuentan con una base de equipos informáticos. Se puede entrar a la aplicación desde distintos dispositivos (ya sea un ordenador o un móvil) con esto permite la mayor facilidad para la accesibilidad de tareas como lo son el control del fichaje de la jornada. La automatización de estas gestiones mejora, y mucho, el funcionamiento interno.

# Análisis de requerimientos del nuevo sistema

Latencia:
- < 300 ms para operaciones críticas (consulta de horario, fichaje).

Escalabilidad:
- Capacidad de escalar instancias PHP-FPM y base de datos en réplica para soportar picos de hasta 500 usuarios concurrentes.

- Tolerancia a fallos

- Clúster de base de datos con master-replica;

Protección de datos:

TLS 1.2+ para todo el tráfico; cifrado AES-256 en reposo.

Registro inmutable de eventos críticos (login, asignaciones, aprobaciones).

Cumplimiento

GDPR y normativa laboral: gestión de consentimiento, derecho al olvido, logs de acceso.

PHPUnit para backend (≥ 80 % de cobertura).

Readme, guías de instalación/despliegue

# Estudio sobre posibles soluciones

Cuando se inició la creación de la aplicación debimos afrontar varias decisiones respecto a la forma en el cual, *AsistGuard,* iba a ser creada pero debíamos plantearnos varios puntos a tener en cuenta:

1. Compatibilidad entre plataformas:

   Este entorno tiene la intención de facilitar el control de información hacia el docente, de tal forma que pueda entrar desde cualquier dispositivo para poder consultar, su horario, si desea consultar las guardias posibles, si necesita notificar una ausencia a un administrador… Por ello se ha preparado su producción para diferentes entornos.

2. Comunicación y envío de datos:

   Para que toda la comunicación entre cliente y servidor este controlada y no haya ningun posible percance, es necesario que en la comunicación se reciba todas las diferentes peticiones y con errores preparados en caso de generar un mal registro de ausencia, un mal log in que no permita la entrada; en definitiva estar siempre atento para que el usuario final sea consciente de los cambios en el sistema.

3. Costes de producción y empleo:

   Es necesario tener en cuenta que si una aplicación requiere de mucho gasto, no todos aceptarán el uso de esta ya que agotaría tantos recursos monetarios como costes de obra por lo que si nos basamos en una producción eficiente a bajo costo muchas empresas estarán dispuestas a apostar por su uso sin tener excesivas perdidas en ello

4. Escalabilidad y adaptación

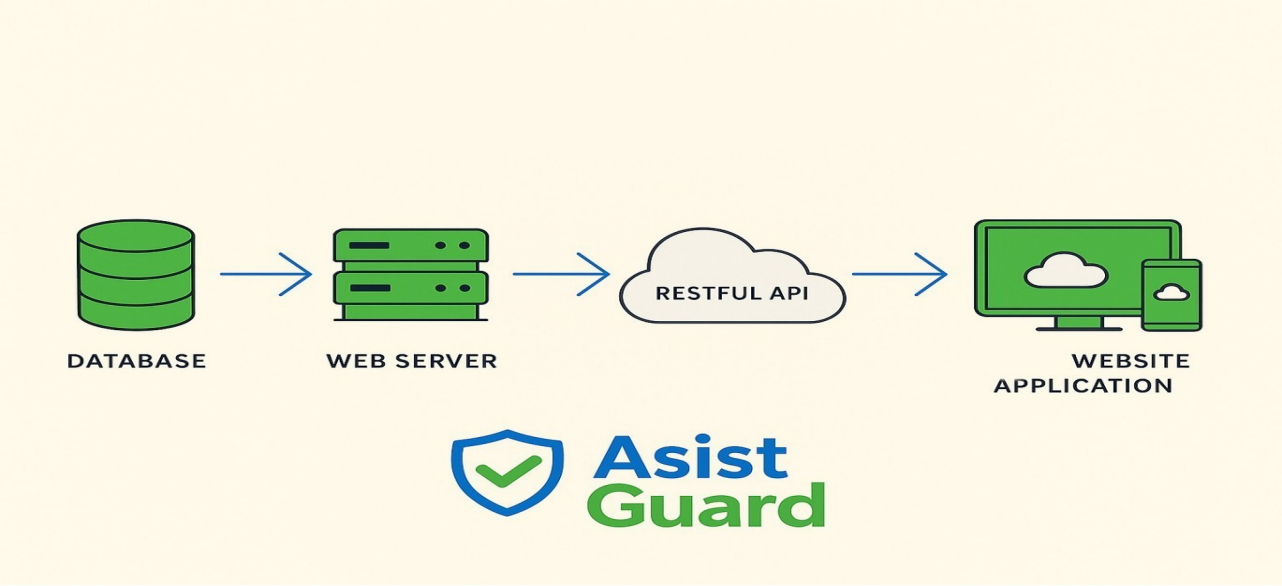
   Además, las instituciones y colegios pueden ir creciendo y manejar mayor cantidad de empleados y administrativos por lo que también se debe de tener en cuenta que la aplicación ha de estar preparada para recibir una cantidad masiva de peticiones, diferentes tipos de contenidos que pueden duplicar en cantidad de memoria en la base de datos… En resumen, poder manejar diferentes potencias de información con la misma eficiencia que en instituciones inferiores

# Solución con entorno escogido

Tras un estudio de las diferentes plataformas con la conexión de los diferentes entornos, se ha decidido utilizar un cliente personalizado desarollado con lengujes de marcación(*HTML5*) hojas de estilo en cascada que personalicen los colores y sombras(*CSS3*), archivos que manejan los diferentes eventos al pulsar diferentes eventos(*Javascript*), y permanencia de datos por sesión(PHP).

Para la comunicación entre estos clientes y el servidor se usará la *API REST.* La forma de trabajo se distingue sobre las peticiones que hace el cliente, pues si desea iniciar sesión por ejemplo, un método que permita enviar estos valores al servidor, compararlos con la base de datos y si coinciden permitir la entrada. Son ejemplos que explican que el sistema diferencia las peticiones de los clientes y dicta al servidor como debe responder.

Finalmente el servidor, individualiza las peticiones teniendo en cuenta las peticiones que recibe, que parámetros recibe en estas peticiones y manda las respuestas codificadas con la información necesaria que responde la petición



# Definición de las tareas y subtareas

Para empezar a la creación estamos en la obligación de instar *IDE's* que nos ayuden y fomenten nuestro proyecto y nos ayude a trabajar de forma mas rápida y eficiente. En nuestro caso, hemos decidido usar Visual Studio Code ya que aparte de que todo nuestro equipo esta acostumbrado a manejarlo, este otorga la ventaja de la posibilidad de instalar varias extensiones que te ayudan a acortar las tareas.

Es necesario que los datos sean introducidos y mantengan relaciones dentro la base de datos para a la hora de la programación sean accesibles desde otros puntos de accesos sin tener que acceder directamente, ademas de contar con su correcta estructura de tablas. Incluso deberemos proteger algunas tablas como la de usuarios de autenticación del menú del Inicio de sesión para que toda comunicación vaya cifrada.

Diseño de bocetos y árbol de ideas para el planteamiento del esquema principal haciendo referencia a que debe estar en lo que es la pagina principal, accesos directos y atajos, botones de fácil uso al registro de jornadas.

Control de permisos para que no todos los usuarios(profesores) tengan acceso a módulos que son creados para administrativos.

Finalmente la posibilidad de la comunicación con los administradores para registrar una ausencia o la consulta de guardias realizadas para mantener un informe detallado.

# Estimación de la producción de la aplicación

## Estimación en tiempo real(basado en semanas)

- Preparación del entorno: 1 semana

  Se instalan todos los paquetes necesarios para la producción de esquemas, código, base de datos y entornos de programación donde viene también incluido el *IDE* mencionado con anterioridad.

- Diseño de la base de datos: 2 semanas

  Se estudian los casos y análisis del funcionamiento de los institutos y se lleva a cabo la preparación y diseño de la base de datos, pues los campos donde se almacenará la información, que tipo de información y la estrecha relación que mantendrá con otras tablas.

- Creación del menú de inicio de sesión: 1 semana

  Se desarrollan las ideas de colores y diseño, además de la verificación de seguridad a llevar a cabo sobre los datos y la comparación de estos con la base de datos. Se asocian ficheros de registro para almacenar los intentos de accesos, el estado, y quien ha sido para poder mantener el control de acceso.

- Fabricación de la página principal de sesión: 1 semana

  Se centra en brindar las funciones de fichaje, consulta del horario diario del docente iniciado, además de hacer consultas de sus guardias a realizar y comunicación con la administración para notificar la ausencia.

- Controles administrativos y creación del registro del docente ausente: 1 semana

  Gestión de comprobación de permisos para el acceso al registro del docente que preavisó de su ausencia y/o comprobación de sus fichajes para notificar su ausencia.

- Generador de informes sobre las ausencias: 1 semana

  Creación de un generador breve de informes notificando que profesor a hecho tantas guardias y tantas ausencias tuvo y por quien fue cubierto.

- Horario de profesores ausentes y consultas realizadas: 2 semanas

  Recogida de notificaciones de ausencias y cantidad de ausencias cubiertas por ese mismo profesor controlado por fechas.

- Pruebas finales y validación: 2 semanas

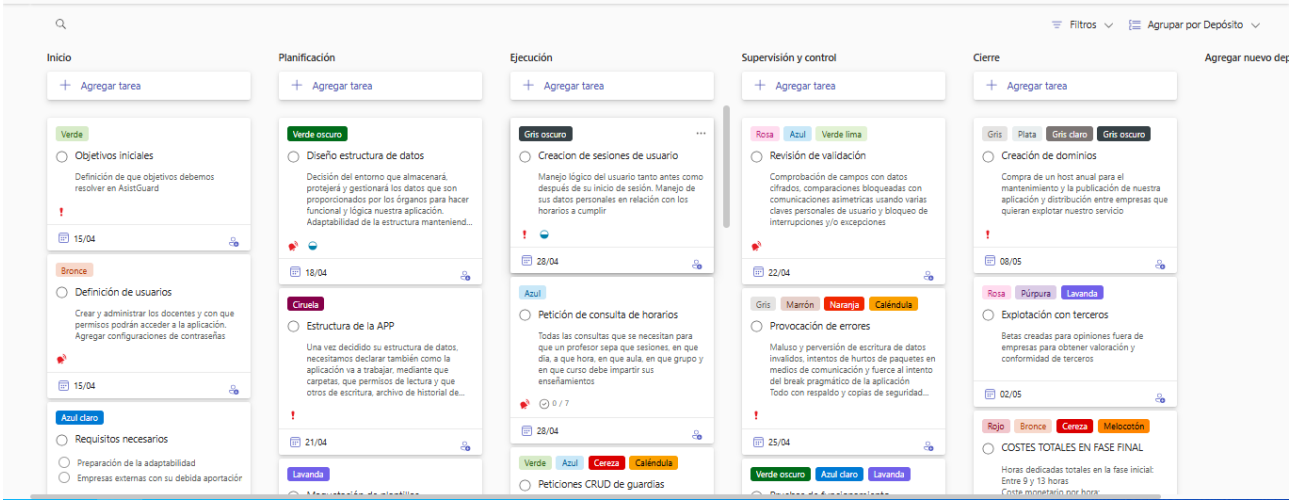
  Comprobaciones finales alrededor de toda la aplicación y prevención de errores.

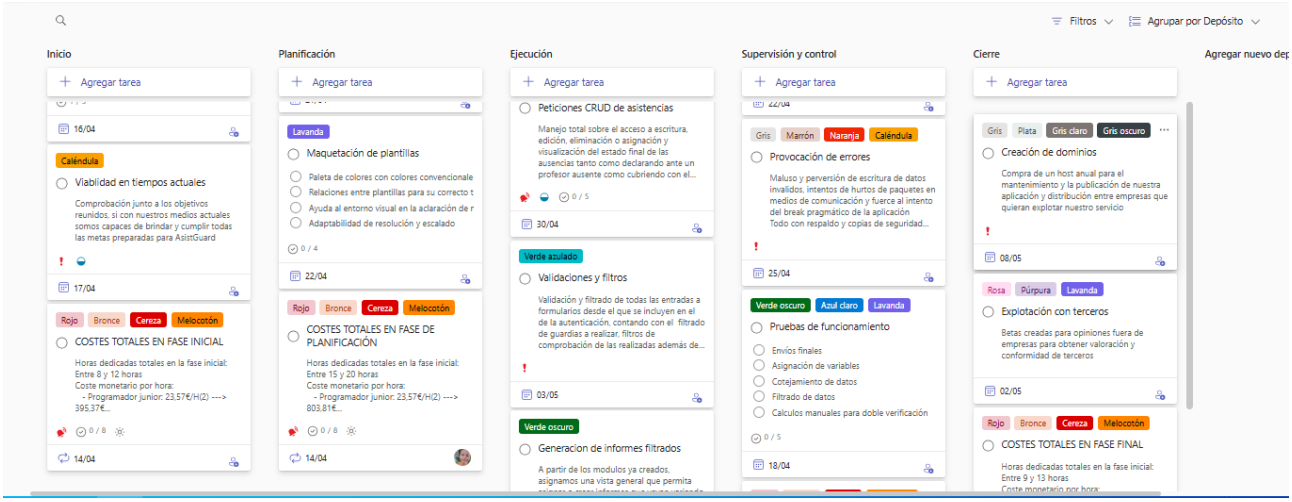
## Estimación económica

- Coste personal : Calculamos cuánto costará contratar a un desarrollador para hacer la aplicación. El desarrollador trabajará unas 30 horas cada semana durante 13 semanas, cobrando 40€ por hora. En total, el coste será de **15.600€**.
- Licencias, servidor y *hosting* (anual) : Para mantener la aplicación funcionando bien y segura, necesitamos pagar licencias de programas, alojamiento web (hosting) y mantenimiento del servidor. Esto cuesta aproximadamente **600€** cada año.
- Contingencias : A veces pueden surgir cosas inesperadas como cambios en la aplicación o problemas técnicos. Para cubrir estos gastos extra, reservamos un 5% adicional del coste total, que son **810€.**


**Total estimado**: **17.010€**

# Diagrama de *Gantt*




Este diagrama de *Gantt* a sido pensado y diseñado por *AsistGuard* y también es accesible pinchando en este enlace:

*DIAGRAMA DE GANTT DE ASISTGUARD*

# Dotación de recursos dedicados al desarrollo

En el desarrollo de *AsistGuard* se han dedicado recursos tales como:

- *Hardware* y dispositivos:

  - Ordenadores independientes actores como clientes de envío personalizado de peticiones.

  - Servidor dedicado para el salvaguardado y alojamiento de la *app.*

- *Software:*

  - *IDE(Interface Developer Equipment):* Uso de *Visual Studio*

  - Uso de *PhpMyAdmin* como un sistema gestor de bases de datos relacional *(MySQL).*

  - Uso del mismo conjunto de herramientas para pruebas de seguridad *(XAMPP, Postman*).

- Recursos humanos:

  - Dos desarrolladores web especializados en frontend mediante el uso de *HTML5, CSS3, JavaScript.*

  - Dos desarrolladores backend especializados en *PHP y APIs REST.*

  - Un administrador de bases de datos (*DBA).*

  - Dos responsables de seguridad para realizar pruebas de vulnerabilidades.

# Configuración del sistema

Para que la aplicación de *AsistGuard* funcione correctamente necesitamos los siguientes paquetes ejecutables:

1.- *XAMPP*: Es un servidor web desarrollado y mantenido gratuitamente por la organización de *Apache Friends* fundada en 2002. Aun sin estadísticas oficiales, aproximadamente el 74% de entornos desarrollados con *PHP* usan *XAMPP*. Este servidor web incluye:

1. Servidor web *Apache*
2. Servidor de base de datos(*MariaDB,MySQL*) conocido como *phpMyAdmin*
3. Entorno preparado para el procesamiento de *PHP*

2.- *Composer*: A pesar de que por si solo *XAMPP* ya incorpora la usabilidad del lenguaje de PHP para poder hacer funcionar por completo se incluye *Composer* por estas razones:

1. Gestión automática de bibliotecas
2. Control de versiones y compatibilidad
3. *Autoloading* (carga automática de clases)

3.- Sistema operativo: AsistGuard está desarrollada para el uso tanto de software libre como *Linux* como para software propietario como lo es *Windows*, ya que ambas funcionan correctamente con lo anteriormente mencionado

4.- Implementación: Es importante mencionar que cada sistema operativo tiene su carpeta raíz de *XAMPP* en sitios diferentes por lo que la aplicación cambiará de lugar. En caso de *Windows* la ruta raíz siempre es en el disco local de esta forma ***C:/xampp*** en cambio en Linux la carpeta se encuentra en ***/opt/lampp/htdocs/*** . La carpeta de la aplicación debe incluirse a continuación(viniéndose a ser la clonación del repositorio de GitHub).

# Análisis y planificaciones

Nos importa que nuestros clientes sepan como es la estructura de AsistGuard por lo que se han planificado sesiones explicativas de unos 45 minutos razonando y describiendo cada módulo:


Todas las reuniones se desarollarán en el salón de actos programada su reunión en el ceontr seleccionado. Estos incluirán folletos transportables que resuman todas las re-uniones en caso de ausencia de alguna reunión

# Maquinaria necesaria junto a su programación y licencias(*HardWare*)

En la creación de la aplicación de AsistGuard, en la maquinaria sólida se incluyen:

| Categoría | Descripción | *Especificaciones mínimas* | Cantidad recomendada |
|---|---|---|---|
| **Servidores de aplicación** | Físico o virtual para la lógica de negocio | *8 vCPU, 16 GB RAM, SSD 500 GB* | 2 (alta disponibilidad) |
| **Servidores de base de datos** | Físico o virtual dedicado | *8 vCPU, 32 GB RAM, RAID SSD 1 TB* | 2 (master + réplica) |
| **Balanceador de carga** | *Appliance o software (e.g. HAProxy/Nginx)* | *4 vCPU, 8 GB RAM* | 1 |
| ***Firewall / UTM*** | *Appliance hardware o virtual* | *Throughput ≥ 1 Gbps, VPN, IPS/IDS* | 1 |
| ***Switches* gestionables** | Nivel 2/3 con *PoE* opcional | *24 puertos, 1 Gbps* | 2 |
| **Router de borde** | Conexión *WAN* | *Throughput ≥ 1 Gbps* | 1 |
| **Puntos de acceso Wi-Fi** | 802.11ac/ax | *Soporte WPA3* | 2-3 por sede |
| **Almacenamiento secundario (NAS)** | Copias de seguridad y archivo | *4 TB RAID 5* | 1 |
| ***SAI/UPS*** | Protección de energía | *1 kVA, autonomía ≥ 10 min* | Según *racks* |
| **Estaciones de desarrollo** | *PC* para desarrolladores | *Core i7, 16 GB RAM, SSD 512 GB* | Según equipo de dev. |
| **Equipos de prueba** | Dispositivos móviles y navegadores para *QA* | *iOS/Android, tablets, navegadores* | 3-5 dispositivos |

# Maquinaria necesaria junto a su programación y licencias(Soft*Ware*)

En la creación de la aplicación de AsistGuard, en la maquinaria programada se incluyen:

- **Sistema operativo servidores**: *Ubuntu LTS* ≥ 22.04
- **Servidor web**: *Apache 2.4 + PHP 8.1* (extensiones *PDO, JSON, mbstring*)
- **Base de datos**: *MySQL 8.0*
- **Caché y colas**: *Redis 6.x*
- **Monitorización**: *Prometheus (licencia Open Source)*
- **Gestión de *logs***: *ELK Stack*
- **Herramientas *CI/CD***: *GitHub Actions* (plan gratuito empresarial)
- **Certificados *TLS***: *Let's Encrypt* (gratuitos)

# Migración e implantación de servicios

Para garantizar una transición segura y ordenada, dividimos la migración en dos grandes modalidades:

1. **Migración en frío (“*cold migration*”)**

    - Seleccionamos los servicios no críticos cuya paralización temporal no impacte en la operativa diaria.
    - Realizamos una parada controlada de cada servicio: detenemos procesos, cerramos conexiones y deshabilitamos *endpoints* antiguos.
    - Exportamos datos y configuraciones en lote, aprovechando herramientas de backup para capturar snapshots completos.

2. **Migración en caliente (*"hot migration"*)**

- Configuramos un clúster de replicación en tiempo real.
- Sincronizamos continuamente los cambios del sistema origen al destino hasta alcanzar un desfase de replicación inferior a unos segundos.

3. ***Backup* y validación de datos**

- Ejecutamos copias completas de base de datos y directorios críticos.
- Verificamos cifrados de ficheros y consistencia de esquemas.
- Probamos restauraciones en paralelo en un entorno de espera.

4. **Arranque y pruebas de regresión**

- Iniciamos los servicios en destino y comprobamos logs de arranque sin errores.
- Realizamos tests funcionales manuales sobre los flujos críticos: alta de ausencia, asignación de guardias, exportación de datos.

# Calendario de ejecución

El calendario de ejecución permite visualizar de un vistazo las fases clave del proyecto, sus fechas de inicio y fin, y los responsables asignados:

- **Planificación y análisis (01/07/2025 – 15/07/2025)**
  Durante esta fase definimos el alcance del proyecto, identificamos riesgos y establecemos los criterios de éxito. El Equipo de Proyecto recopila requisitos y valida procedimientos.
- **Preparación de entornos (16/07/2025 – 31/07/2025)**
  El Administrador TI provisiona servidores y configuraciones necesarias (desarrollo, pruebas y preproducción), instala dependencias y revisa conectividad de red, garantizando que todo esté listo para el arranque.
- **Pilotaje (01/08/2025 – 15/08/2025)**
  El Coordinador Piloto supervisa el despliegue inicial en un entorno controlado.
- **Despliegue en producción (16/08/2025 – 31/08/2025)**
  El Equipo de Proyecto ejecuta la migración, monitoriza la estabilidad del sistema y realiza pruebas de rendimiento y de carga. Se valida que todos los servicios estén activos y operativos en el entorno de producción.
- **Formación y comunicación (01/09/2025 – 10/09/2025)**
  El Responsable de RRHH coordina sesiones formativas con usuarios finales y administradores. Se distribuyen manuales, tutoriales y se envían comunicaciones periódicas para resolver dudas y facilitar la adopción del nuevo sistema.


| Fase | Fecha de inicio | Fecha de finalizacion | Responsable |
|---|---|---|---|
| Planificación y analisis | 01/07/2025 | 15/07/2025 | Equipo de proyecto |
| Preparación de entornos | 16/07/2025 | 31/07/2025 | Administrador TI |
| Pilotaje | 01/08/2025 | 15/08/2025 | Coordinador piloto |
| Despliegue en producción | 16/08/2025 | 31/08/2025 | Equipo de proyecto |
| Formación y comunicación | 01/09/2025 | 10/09/2025 | Responsable RRHH |
| | 11/09/2025 | 30/09/2025 | Servicio de soporte |

# Recursos bibliográficos

Los recursos técnicos y documentación oficial que han servido como base para tomar decisiones de diseño, implementación y buenas prácticas. Los enlaces visitados son:

I. *Laravel Official Documentation* (https://laravel.com/docs):

I.I Manual oficial del *framework Laravel,* utilizado para consultar su uso y la comparación de eficiencia(Aun así, ya nombrado anteriormente, esta idea fue descartada)

II. *W3Schools Web Development* (https://www.w3schools.com):

II.I Referencias y ejemplos prácticos sobre *HTML5, CSS3, JavaScript y MySQL.*

III. *MDN Web Docs* (https://developer.mozilla.org):

III.I Documentación avanzada sobre tecnologías web y buenas prácticas de desarrollo.

IV. *Bootstrap Official Documentation* (https://getbootstrap.com):

IV.I Guía de uso del framework *Bootstrap*, utilizado para el diseño responsivo de la interfaz.

V. *PHP Manual (https://www.php.net/manual/en/index.php)*

V.I Manual interno desarollado por los mismos que crearon el propio lenguaje.


VI. Documentación interna del centro educativo:

VI.I Esquemas de horarios, lista de profesorado y estructura de datos proporcionados en el archivo *guardias.sql,* base para la definición del modelo de datos.


AsistGuard www.asistguard.com