MEDAC

Instituto Oficial de Formación Profesional

GuardiApp

Desarrollo de Aplicaciones Web

Tutor: Javier Ruiz Jurado

Curso académico: 2025-2026

Autores

- Javier Juárez Canals
- Raul Henares Palacios
- Antonio Moyano Garcia
- Jose Ramón Rejano Ruge

Febrero, 2026

“La mayoría de los errores médicos son consecuencia de sistemas defectuosos, no de personas defectuosas.” [Lucian Leape, Pionero mundial en seguridad del paciente.] (1930 – 2025)

Índice general

Parte I

Presentación

Capítulo 1

Introducción

1.1. Contexto del problema a resolver

La gestión de turnos y guardias en instituciones sanitarias es una tarea crítica que requiere precisión, organización y una correcta asignación de recursos humanos. En hospitales de tamaño medio, como el hospital de Pozoblanco, esta gestión influye directamente tanto en el correcto funcionamiento del servicio como en la satisfacción laboral del personal sanitario.

Actualmente, muchas de estas tareas siguen realizándose mediante herramientas genéricas como hojas de cálculo, lo que dificulta la automatización de procesos, la trazabilidad de la información y la reducción de errores humanos.

La ausencia de un sistema específico adaptado a las necesidades reales del hospital provoca ineficiencias que pueden derivar en conflictos organizativos y errores en la planificación.

1.2. Definición del problema real

Existe un problema con respecto a la gestión de las guardias del hospital de Pozoblanco.

En resumen, el problema real que se desea resolver con el desarrollo de este proyecto es el siguiente:

Actualmente, el sistema de guardias del hospital de Pozoblanco se gestiona mediante hojas de cálculo en formato Excel, las cuales son administradas manualmente por los empleados responsables. Estos empleados deciden, utilizando distintos términos y patrones no estandarizados, qué trabajadores se encargan de las guardias de cada día, plasmando finalmente la información en un documento PDF.

Este proceso resulta tedioso y propenso a errores, ya que gran parte de la gestión de guardias, horas trabajadas y días libres (entre otros aspectos) depende del factor humano. Esto puede provocar inconsistencias en los datos, errores de cálculo

y dificultades a la hora de realizar modificaciones o auditorías posteriores.

1.3. Definición del problema técnico

El problema técnico consiste en la ausencia de una herramienta software específica que permita gestionar de forma automatizada, segura y centralizada las guardias del hospital de Pozoblanco.

Desde el punto de vista técnico, se requiere el desarrollo de una aplicación web capaz de gestionar usuarios con distintos niveles de acceso, almacenar información estructurada sobre guardias y profesionales, y aplicar reglas de negocio para la planificación y validación de los turnos.

Para abordar este problema, se utilizará una arquitectura basada en una aplicación web con un backend encargado de la lógica de negocio y un frontend responsable de la interacción con el usuario, comunicados mediante una API REST.

1.3.1. Funcionamiento

Se desea implementar una aplicación web que permita realizar una gestión automatizada de las guardias del hospital, reduciendo la dependencia del factor humano y minimizando los errores derivados de la gestión manual.

Los usuarios que interactuarán con la aplicación son los siguientes:

- **Administrador**
  - Usuario con acceso completo a las funcionalidades del sistema.
  - Responsable de la gestión de usuarios, guardias, jefaturas y validaciones.
- **Usuario de consulta**
  - Usuario con acceso a funcionalidades de consulta de guardias.
  - Sin permisos para modificar información crítica del sistema.

1.3.2. Entorno

Existirán dos entornos de trabajo para esta aplicación:

- **Entorno de desarrollo y testeo**: entorno utilizado para desarrollar, probar y depurar la aplicación web antes de su despliegue.
- **Entorno de producción**: entorno en el que se encontrará la versión final de la aplicación web accesible para los usuarios finales.

Asimismo, existirán dos entornos de ejecución:

- **Entorno de ejecución del administrador**: acceso completo a las funcionalidades del sistema.
- **Entorno de ejecución del resto de usuarios**: acceso limitado a funcionalidades de consulta.

La interfaz será el medio de comunicación entre los usuarios y la aplicación web. Es fundamental que sea intuitiva y amigable para garantizar una buena experiencia de usuario. Los requisitos de la interfaz se describirán en la Sección 5.4.3.

1.3.3. Vida esperada

El ciclo de vida esperado de la aplicación web será elevado, ya que se ha diseñado siguiendo una arquitectura modular y escalable que permite su adaptación a futuras necesidades del hospital, como la incorporación de nuevas funcionalidades o la ampliación de los tipos de usuario.

1.3.4. Ciclo de mantenimiento

Dado que el desarrollo de la aplicación web se enmarca dentro de un proyecto académico, el mantenimiento posterior de la aplicación no correrá a cargo de su autor.

No obstante, se ha diseñado una estructura modular que facilite posibles tareas de mantenimiento, corrección de errores o ampliaciones futuras por parte de terceros.

1.3.5. Competencia

En la actualidad existen soluciones comerciales para la gestión de turnos y guardias, así como herramientas genéricas utilizadas para este fin. No obstante, muchas de estas soluciones no se adaptan completamente a las necesidades específicas del hospital de Pozoblanco.

Este proyecto se plantea como una solución personalizada, orientada a cubrir de forma concreta los requisitos del contexto hospitalario descrito.

1.3.6. Aspecto externo

En relación con el aspecto externo, se tendrán en cuenta los siguientes aspectos:

- **Interfaz de usuario**
  - Se desarrollará una interfaz totalmente responsiva, intuitiva y amigable. El Capítulo 10 describirá el diseño de la interfaz de la aplicación web.
- **Distribución de la aplicación**
  - La aplicación se podrá distribuir mediante un servidor web accesible a través de un navegador estándar.

1.3.7. Estandarización

Para el desarrollo de la aplicación web, se revisarán las recomendaciones propuestas por *World Wide Web Consortium* (W3C) [7], que "promueve el uso de estándares para reducir el coste y la complejidad del desarrollo, así como para incrementar la accesibilidad y usabilidad de cualquier documento publicado en la web".

También se debe tener en cuenta que los recursos que se van a utilizar son herramientas informáticas que están validadas por prestigiosas organizaciones que indican que cumplen con los estándares. Véase el capítulo 3 de Recursos.

1.3.8. Calidad y fiabilidad

La calidad y fiabilidad de la aplicación web estarán garantizadas por:

- Uso de un framework consolidado como Laravel y React.
- Aplicación de buenas prácticas de programación.
- Realización de pruebas funcionales y de seguridad.

1.3.9. Programa de tareas

El desarrollo del presente proyecto va estar compuesto por la siguientes fases:

- Introducción: descripción del problema, establecimiento de los objetivos, revisión de antecedentes, identificación de restricciones iniciales y estratégicas y selección de recursos.
- Análisis: especificación de requisitos (funcionales, no funcionales, de información y de la interfaz), descripción del modelo de datos y análisis funcional (casos de uso y diagramas de secuencia).
- Diseño: descripción del diseño de datos, clases, paquetes y de la interfaz.
- Implementación: codificación de la aplicación web teniendo en cuenta el diseño desarrollado.

- Pruebas: comprobación de que la aplicación web funciona correctamente, es robusta y amigable.

1.3.10. Pruebas

La fase de pruebas es esencial para garantizar que la aplicación web funciona correctamente. En particular, se pretende comprobar que la aplicación web:

- Hace lo que debe hacer.
- No provoca efectos secundarios que pueden desencadenar situaciones catastróficas.
- Contiene módulos que se ejecutan correctamente.
- Garantiza los privilegios de cada tipo de usuario.

Cada prueba tendrá la siguiente estructura para detectar los errores y corregirlos:

- Objetivo de la prueba. Se debe indicar en qué consiste la prueba y el resultado esperado.
- Problema detectado, en su caso. Si ocurre un error entonces se debe describir la causa que lo ha provocado.
- Solución adoptada, en su caso. Si se ha producido un error, se deben indicar las medidas tomadas para solucionarlo.

El Capítulo 11 de Pruebas describirá las pruebas realizadas.

1.3.11. Seguridad

Durante el desarrollo de la aplicación web se tendrán en cuenta aspectos fundamentales de seguridad, entre los que se incluyen:

- Autenticación y autorización de usuarios.
- Protección contra ataques de inyección SQL.
- Gestión segura de sesiones.
- Protección de datos sensibles.

Estos aspectos se describirán con mayor detalle en los capítulos correspondientes.

Capítulo 2

Objetivos

2.1. Objetivo principal

El objetivo principal de este proyecto es el desarrollo de una aplicación web que permita la gestión automatizada y centralizada de las guardias del hospital de Pozoblanco, sustituyendo el actual sistema manual basado en hojas de cálculo por una solución informática fiable, segura y fácil de usar.

La aplicación estará orientada a mejorar la eficiencia en la planificación de las guardias, reducir errores humanos y facilitar el acceso a la información tanto a los administradores como al personal del hospital.

2.2. Objetivos específicos

Los objetivos específicos de este proyecto son los siguientes:

- Diseñar e implementar un sistema de gestión de usuarios con distintos niveles de acceso, diferenciando entre administradores y usuarios de consulta.
- Desarrollar un sistema de generación automática de cuadrantes mensuales de guardias, aplicando reglas de negocio definidas por el hospital.
- Permitir la consulta de guardias por parte del personal del hospital mediante vistas diarias y mensuales, organizadas por sección.
- Implementar un mecanismo de solicitud y validación de cambios de guardia entre profesionales, sujeto a un flujo de aprobación controlado.
- Diseñar y desarrollar una base de datos relacional que permita almacenar de forma segura toda la información relacionada con usuarios, guardias, secciones y jefaturas.
- Proporcionar herramientas de gestión para el administrador que faciliten la supervisión, modificación y validación de las guardias.

- Diseñar una interfaz de usuario intuitiva, accesible y adaptable a distintos dispositivos y navegadores web.
- Garantizar la seguridad del sistema mediante mecanismos de autenticación, autorización y control de acceso adecuados.

- **Tipos de usuarios**

  Se deberán permitir los siguientes tipos de usuarios:

  - **Administrador**
    - Este tipo de usuario estará registrado en el sistema y dispondrá de control completo sobre la aplicación.
    - Tendrá competencias exclusivas para la gestión de usuarios, creación y modificación de guardias, asignación de turnos, consulta de estadísticas y generación de informes.
  - **Usuario estándar**
    - Podrá consultar las guardias que tiene asignadas, así como su histórico de horas trabajadas y días libres.
    - No dispondrá de permisos para modificar información crítica del sistema.

- **Base de datos relacional**

  Se deberá diseñar una base de datos relacional que permita gestionar toda la información relacionada con el sistema de guardias, incluyendo:

  - Usuarios registrados y sus roles.
  - Guardias y turnos asignados.
  - Fechas, horarios y tipos de guardia.
  - Históricos de asignaciones y modificaciones.

- **Módulos del sistema**

  Se deberán diseñar e implementar los siguientes módulos principales:

  - **Módulo del administrador**
    - Gestión de usuarios registrados.
    - Creación, modificación y eliminación de guardias.
    - Asignación y reorganización de turnos.
    - Generación de informes y consultas.
  - **Módulo del usuario estándar**
    - Consulta de guardias asignadas.
    - Visualización de calendario de turnos.
    - Acceso a información personal relacionada con horas trabajadas.

- **Diseño de la interfaz**

Se deberá diseñar una interfaz de usuario intuitiva, robusta, amigable y adaptable a distintos dispositivos y navegadores web, garantizando una experiencia de uso satisfactoria.

- **Seguridad y control de acceso**

  Se deberán implementar mecanismos de autenticación y autorización que garanticen que cada usuario solo pueda acceder a las funcionalidades correspondientes a su rol.

2.3. Objetivos personales

Se proponen los siguientes objetivos personales:

- Aprender y afianzar conocimientos en el desarrollo de aplicaciones web utilizando el framework Laravel y la librería React.
- Profundizar en el uso del patrón Modelo–Vista–Controlador (MVC).
- Mejorar habilidades en el diseño y gestión de bases de datos relacionales.
- Aplicar buenas prácticas de desarrollo de software y documentación técnica.
- Adquirir experiencia en el análisis, diseño, implementación y pruebas de un proyecto software completo.
- Conseguir que todos los miembros del grupo mejoren y se adapten al trabajo en equipo y tener a Raul contento.

Capítulo 3

Recursos

3.1. Recursos humanos

- **Raúl**

  Responsable principal del backend y la lógica del servidor. Su labor incluye la creación de las API, la comunicación con la base de datos y la implementación de la seguridad del sistema.

- **Javier**

  Responsable del análisis y la documentación técnica. Se encarga de la especificación de requisitos, la estructura de la documentación y la supervisión del seguimiento metodológico del proyecto.

- **Jose**

  Responsable de frontend y diseño de la interfaz. Se encarga de la maquetación, la usabilidad, el diseño responsive y la experiencia de usuario.

- **Antonio**

  Labores híbridas entre front y back apoyando a Raul y a Jose en gran medida con sus labores. Se encarga de unir backend y frontend, realizar pruebas, gestionar el despliegue en el servidor y solucionar incidencias del entorno.

3.2. Recursos de hardware

Para el desarrollo de la aplicación en el entorno local:

- Ordenador: PC de sobremesa / portátil de uso personal.
- Sistema operativo: Windows 10 / Windows 11 / MacOs.
- RAM instalada: 16 GB.

- Procesador: Intel Core i5 / i7 de 10ª generación o superior.

Para el despliegue de la aplicación web, se hará uso de un servidor VPS con las características siguientes:

- Platform: Servidor VPS en la nube.
- OS Package: Linux Ubuntu Server 22.04 LTS.
- Memory: 2 GB – 4 GB de RAM.
- SSD: 40 GB – 80 GB de almacenamiento SSD.

3.3. Recursos de software

Se van a utilizar los siguientes recursos de software para el desarrollo de la aplicación web:

- Editor de código fuente para el desarrollo de la aplicación web: Visual Studio Code [6]
- Editor de código fuente: Visual Studio Code [6].
- Lenguaje backend: PHP 8.4 con framework Laravel.
- Lenguaje frontend: HTML5, CSS, JavaScript y React.
- Sistema gestor de bases de datos: MySQL.
- Control de versiones: Git y plataforma GitHub.
- Servidor local de desarrollo: XAMPP.
- Navegadores para pruebas: Google Chrome y Mozilla Firefox.
- Herramientas adicionales: Postman para pruebas de API, Figma para prototipado de interfaz, Composer para gestión de dependencias, plesk para el servidor.

Parte II

Contextualización empresarial

Capítulo 4

CONTEXTUALIZACIÓN EMPRESARIAL

4.1. Denominación

El proyecto consiste en el desarrollo de una aplicación web denominada **GuardiApp**, orientada a la gestión automatizada y centralizada de las guardias de un hospital, con el objetivo de sustituir los procesos manuales actuales basados en hojas de cálculo.

GuardiApp permitirá la planificación automática de los cuadrantes mensuales, la consulta de guardias por parte del personal sanitario y la gestión administrativa de turnos, cambios y jefaturas, proporcionando una solución especializada para centros hospitalarios.

A nivel conceptual, GuardiApp se plantea como una herramienta software adaptable a las necesidades específicas de hospitales de tamaño medio, priorizando la eficiencia, la automatización y la facilidad de uso.

4.2. Justificación de la idea del proyecto

La idea del proyecto surge a partir de la detección de una necesidad real en el entorno hospitalario: la gestión de guardias continúa realizándose, en muchos casos, mediante procesos manuales apoyados en hojas de cálculo y documentos generados de forma artesanal.

Este enfoque conlleva una elevada carga administrativa, un alto riesgo de errores humanos y una escasa trazabilidad de los cambios realizados, especialmente en lo relativo a modificaciones de guardias, asignación de jefaturas y validaciones.

GuardiApp pretende dar respuesta a esta problemática mediante la automatización del proceso de planificación de guardias, ofreciendo una solución centralizada que facilite tanto el trabajo del personal administrativo como el acceso a la información por parte de los profesionales sanitarios.

4.3. Estudio de la competencia

Como competidores directos se pueden considerar aplicaciones de gestión hospitalaria que incluyen módulos de planificación de personal. Estas soluciones suelen ser completas, pero presentan un elevado coste económico y una complejidad de implantación que dificulta su adopción en hospitales de tamaño medio.

Por otro lado, existen competidores indirectos como herramientas genéricas de gestión de turnos o el uso continuado de hojas de cálculo, que, aunque no están diseñadas específicamente para la gestión de guardias hospitalarias, siguen utilizándose por su bajo coste y facilidad de acceso.

GuardiApp se diferencia de estas alternativas al ofrecer una solución específica, flexible y adaptada a la realidad de los hospitales, centrada en la automatización de guardias y en la simplificación de los procesos administrativos.

4.4. Oportunidad de negocio

La transformación digital del sector sanitario es una tendencia en crecimiento, impulsada por la necesidad de mejorar la eficiencia operativa, reducir costes y garantizar la trazabilidad de la información.

La gestión de guardias representa un proceso crítico dentro de cualquier hospital y, sin embargo, no siempre cuenta con herramientas específicas que se adapten a las necesidades reales de cada centro.

GuardiApp cubre una necesidad existente en el mercado, ofreciendo una solución que puede ser implantada de forma progresiva y con un coste reducido en comparación con sistemas comerciales complejos, lo que la convierte en una opción viable para hospitales de tamaño medio.

Por todo ello, el proyecto presenta una oportunidad de negocio clara dentro del ámbito de la digitalización de procesos administrativos en el sector sanitario.

4.5. Obligaciones fiscales

El desarrollo de GuardiApp, en el contexto académico del proyecto, no conlleva inicialmente la generación de obligaciones fiscales, laborales ni de prevención de riesgos laborales, más allá de las propias del desarrollo del software.

En un escenario de explotación comercial de la aplicación, sería necesario cumplir con las obligaciones fiscales correspondientes a la actividad empresarial, así como con la normativa laboral vigente en caso de contratación de personal.

Asimismo, dado que la aplicación gestionaría información sensible del ámbito sanitario, sería imprescindible garantizar el cumplimiento de la normativa vigente en materia de protección de datos personales, en especial el Reglamento General de

Protección de Datos (RGPD).

4.6. Financiación, ayudas y subvenciones

El desarrollo inicial de GuardiApp no requiere una inversión económica significativa, ya que se basa en el uso de tecnologías de software libre y herramientas de desarrollo gratuitas.

En caso de evolucionar el proyecto hacia un producto comercial, podrían contemplarse distintas fuentes de financiación, como subvenciones públicas destinadas a la digitalización de servicios sanitarios, ayudas a la innovación tecnológica o programas de apoyo al emprendimiento.

Del mismo modo, podrían establecerse acuerdos de colaboración con centros hospitalarios interesados en la implantación de soluciones tecnológicas adaptadas a sus necesidades específicas.

Parte III

Análisis

Capítulo 5

Especificación de requisitos

5.1. Introducción

Se desea desarrollar una aplicación web que permita la gestión automatizada y centralizada de las guardias del hospital de Pozoblanco, sustituyendo el proceso actual basado en hojas de cálculo y generación manual de documentos.

Las siguientes secciones describen los actores del sistema (Sección 5.2), la descripción modular (Sección 5.3) y los requisitos del sistema (Sección 5.4), incluyendo requisitos funcionales, no funcionales, de interfaz y de información.

5.2. Actores del sistema

Se van a considerar los siguientes tipos de usuario en la aplicación web:

- **Administrador**: usuario registrado en el sistema con permisos completos para la gestión de datos, profesionales, especialidades y guardias. Es responsable de la generación automática de los cuadrantes mensuales, la asignación y validación de jefes de guardia, la aprobación de cambios de guardia, el cierre de meses y la administración de usuarios y auditoría.
- **Usuario de consulta**: usuario registrado con permisos limitados, orientado a la consulta de información, que puede visualizar guardias, vistas diaria y mensual por sección, y realizar solicitudes de cambio de guardia que deberán ser validadas.

5.3. Módulos de la aplicación

La aplicación web estará compuesta por módulos que corresponden a cada tipo de usuario.

5.3.1. Módulo del usuario de consulta

El usuario de consulta podrá:

- Consultar guardias por mes, sección y profesional.
- Visualizar la vista diaria con guardias agrupadas por sección, incluyendo profesional, tipo de guardia y jefe de guardia global.
- Consultar sus guardias asignadas en un mes determinado.
- Solicitar cambios de guardia con otros profesionales.
- Consultar el estado de las solicitudes de cambio realizadas.

5.3.2. Módulo del administrador

El usuario Administrador tendrá control total de la aplicación y se encargará de:

- Generación automática de los cuadrantes mensuales de guardias por sección.
- Gestión manual de guardias y jefes de guardia cuando sea necesario.
- Validación y aprobación de solicitudes de cambio de guardia.
- Gestión de profesionales, secciones y asociaciones entre ellos.
- Cierre y reapertura de meses de guardias con control de permisos.
- Gestión de usuarios, roles y auditoría de acciones.

5.4. Requisitos del sistema

Los requisitos del sistema hacen referencia a todas las características relacionadas con la aplicación web. Se describirán los siguientes tipos de requisitos:

- Requisitos funcionales: describen las tareas que la aplicación debe realizar para satisfacer las necesidades del problema (Sección 5.4.1).
- Requisitos no funcionales: describen cómo se tiene que satisfacer las necesidades del problema (Sección 5.4.2).
- Requisitos de la interfaz: describen cómo debe ser la comunicación entre el usuario y la aplicación (Sección 5.4.3).
- Requisitos de la información: describen las características de los datos que se van a gestionar (Sección 5.4.4).

5.4.1. Requisitos funcionales

Los requisitos funcionales indican lo que el sistema debe hacer. Se denotarán como RF-<nº requisito>y se agrupan por funcionalidades del sistema.

- **Generación automática del cuadrante de guardias**
  - RF-00. Generación automática del cuadrante mensual de guardias por sección del hospital.
  - RF-00.1. Garantizar que todas las guardias necesarias estén cubiertas para cada día del mes.
  - RF-00.2. Asociar cada guardia generada a una sección y a un profesional.
  - RF-00.3. Permitir la revisión y modificación manual del cuadrante generado por un administrador.
- **Importación del cuadrante de guardias**
  - RF-01. Subida de un fichero Excel correspondiente al cuadrante mensual.
  - RF-02. Validación de la estructura del Excel (columnas, formatos, especialidad, profesional, tipo CA/PF/LOC).
  - RF-03. Importación de guardias a la base de datos asociando mes, especialidad, profesional y tipo.
  - RF-04. Detección y evitación de guardias duplicadas.
  - RF-05. Informe de importación con registros correctos/erróneos y motivo.
  - RF-06. Repetición de importación de un mes solo con confirmación de un usuario autorizado.
- **Gestión de datos maestros**
  - RF-07. Gestión de especialidades (alta, baja y modificación).
  - RF-08. Importación de Excel con listado oficial de facultativos y fecha de inicio en el centro.
  - RF-09. Asociación de profesionales a una especialidad, si procede.
- **Gestión de tipos de guardia**
  - RF-10. Tipos predefinidos: CA (15:00–20:00), PF (24h), LOC (24h).
  - RF-11. Toda guardia debe asociarse obligatoriamente a uno de estos tipos.
  - RF-12. Horas asociadas por tipo: CA=5, PF=24, LOC=24.
- **Gestión del mes de guardias**
  - RF-13. Consulta de meses de guardias importados.
  - RF-14. Cierre de mes para evitar modificaciones tras validación.
  - RF-15. Reapertura de mes cerrado solo por usuarios autorizados.
- **Asignación de Jefe de Guardia Global**

- RF-16. Un único jefe de guardia por cada día del mes.
- RF-17. Candidatos: profesionales con guardia ese día (cualquier tipo/especialidad).
- RF-18. Asignación automática al profesional con mayor antigüedad.
- RF-19. Ningún profesional debe tener más de 3 jefaturas en el mismo mes.
- RF-20. Si el de mayor antigüedad supera el límite, asignar al siguiente que cumpla.
- RF-21. Si nadie cumple, marcar el día como “pendiente de revisión”.
- RF-22. Asignación manual en días pendientes, registrando motivo y usuario.
- RF-23. Mostrar el jefe de guardia en las vistas diaria y mensual.

- **Consultas y visualización de guardias**
  - RF-24. Consulta por mes y especialidad.
  - RF-25. Consulta por profesional en un mes determinado.
  - RF-26. Vista diaria con guardias agrupadas por especialidad, profesional, tipo y jefe global.
  - RF-27. Búsqueda de profesionales por nombre o identificador.
- **Cálculo de horas de guardia**
  - RF-28. Cálculo automático de horas totales por profesional en un mes según tipo.
  - RF-29. Mostrar desglose de horas por día y por tipo (CA, PF, LOC).
  - RF-30. Exportación del resumen de horas por profesional y por especialidad.
- **Cálculo de importe estimado por guardias**
  - RF-31. Configuración de tarifas por tipo de guardia (CA, PF, LOC).
  - RF-32. Cálculo del importe estimado mensual por profesional (horas, tipo y tarifas).
  - RF-33. Contemplar plus de jefatura configurable sumado al total cuando corresponda.
  - RF-34. Mostrar desglose del importe (horas, tarifas y pluses).
  - RF-35. Generar informe mensual de importes por profesional y por especialidad.
- **Generación de documentos y envío por correo**
  - RF-36. Generar PDF diario con fecha, jefe global y guardias agrupadas por especialidad (profesional y tipo).
  - RF-37. Permitir descarga del PDF diario.
  - RF-38. Envío automático por email del PDF diario a destinatarios configurados.

  - RF-39. Generar Excel mensual con total de horas por profesional.
  - RF-40. El Excel incluirá profesional, especialidad, total de horas y desglose por tipo.
  - RF-41. Permitir descarga del Excel mensual.

- **Usuarios, roles y auditoría**

  - RF-42. Autenticación de usuarios.
  - RF-43. Gestión de roles diferenciando al menos Administrador y Usuario de consulta.
  - RF-44. Registro de historial de acciones (importaciones, cambios manuales, cierres de mes y envíos de correo).

- **Gestión de cambios de guardia**

  - RF-45. Solicitud de cambio de guardia por parte de un usuario de consulta.
  - RF-46. Asociación de la solicitud a una guardia concreta y a un profesional receptor.
  - RF-47. Aceptación o rechazo inicial de la solicitud por el profesional receptor.
  - RF-48. Validación del cambio por el jefe de guardia del día afectado.
  - RF-49. Aprobación o rechazo final del cambio por parte de un administrador.
  - RF-50. Actualización automática del cuadrante tras la aprobación del cambio.
  - RF-51. Registro del cambio de guardia en el sistema de auditoría.

5.4.2. Requisitos no funcionales

Los requisitos no funcionales representan cómo tiene que trabajar la aplicación. Se denotarán como RNF-<nº requisito>.

- RNF-1. La aplicación solamente tendrá un usuario con el rol de administrador.
- RNF-2. El borrado de registros en la base de datos se realizará mediante borrado lógico (soft delete), cuando aplique, mediante la actualización de un campo.
- RNF-3. La aplicación deberá implementar autenticación segura mediante Laravel Sanctum para el acceso de usuarios registrados.
- RNF-4. La autorización deberá estar basada en roles, de forma que cada usuario solo pueda acceder a las funcionalidades correspondientes.
- RNF-5. Las contraseñas deberán almacenarse utilizando algoritmos de hash seguros proporcionados por el framework.

- RNF-6. La aplicación deberá protegerse frente a ataques comunes en aplicaciones web, tales como inyección SQL, XSS y CSRF.
- RNF-7. La comunicación entre el frontend (React) y el backend (Laravel) se realizará exclusivamente mediante una API REST.
- RNF-8. La API deberá devolver respuestas consistentes con códigos HTTP apropiados y mensajes de error tratables por el frontend.
- RNF-9. La API deberá permitir el acceso únicamente desde orígenes autorizados mediante una configuración de CORS.
- RNF-10. La gestión de sesión/autenticación en la SPA deberá mantenerse de forma segura utilizando los mecanismos proporcionados por Sanctum.
- RNF-11. Las operaciones críticas (importación de Excel, cierre/reapertura de mes y asignaciones manuales) deberán ejecutarse de forma transaccional para garantizar la integridad de los datos.
- RNF-12. El sistema deberá garantizar la integridad referencial entre entidades (guardias, profesionales, especialidades, meses, etc.).
- RNF-13. El sistema deberá registrar un historial/auditoría de acciones relevantes, indicando usuario, fecha y detalle de la operación.
- RNF-14. La interfaz deberá ser responsiva y compatible con los principales navegadores web actuales.
- RNF-15. La aplicación deberá ser usable por personal no técnico, priorizando claridad en textos, navegación y mensajes de validación.
- RNF-16. El código deberá seguir buenas prácticas de desarrollo y una estructura modular que facilite el mantenimiento y la escalabilidad.
- RNF-17. El proyecto deberá incluir control de versiones (por ejemplo, Git) y un flujo de trabajo en equipo que facilite la integración de cambios.
- RNF-18. Las solicitudes de cambio de guardia deberán seguir un flujo de validación que incluya al jefe de guardia y a un administrador.

5.4.3. Requisitos de la interfaz

La interfaz es el medio que permite la comunicación entre el usuario y el sistema. En esta sección se enumeran los requisitos que debe cumplir la interfaz para que pueda ser utilizada por todos los tipos de usuario.

Los requisitos de la interfaz especifican cómo debe ser la comunicación entre el usuario y la parte visible de la aplicación. Se denotarán como RINT-<nº de requisito>.

- RINT-1. La interfaz deberá ser responsiva y adaptarse a distintos tamaños de pantalla.

- RINT-2. La navegación deberá ser clara y consistente en todas las vistas de la aplicación.
- RINT-3. La interfaz deberá mostrar menús y opciones en función del rol del usuario autenticado.
- RINT-4. La aplicación deberá informar al usuario mediante mensajes claros de éxito o error tras cada operación relevante.
- RINT-5. La interfaz deberá mostrar indicadores de carga durante operaciones que requieran tiempo de procesamiento.
- RINT-6. Las vistas diaria y mensual deberán presentar la información de guardias de forma estructurada y fácilmente legible.

5.4.4. Requisitos de información

Los requisitos de información hacen referencia a los datos que debe gestionar la aplicación web. Se denotarán como RI-<nº de requisito>.

Se deberá almacenar la siguiente información:

- RI-1. Información de usuarios del sistema, incluyendo credenciales y rol asociado.
- RI-2. Información de profesionales (facultativos), incluyendo datos identificativos y fecha de inicio en el centro.
- RI-3. Información de especialidades del hospital.
- RI-4. Información de guardias, incluyendo fecha, profesional, especialidad y tipo de guardia (CA, PF, LOC).
- RI-5. Información de los meses de guardias y su estado (abierto o cerrado).
- RI-6. Información de la jefatura global diaria, incluyendo profesional asignado y motivo en caso de asignación manual.
- RI-7. Información de tarifas por tipo de guardia y plus de jefatura.
- RI-8. Información de documentos generados (PDF diario y Excel mensual).
- RI-9. Registro de auditoría de acciones relevantes realizadas en el sistema.
- RI-10. Información de las solicitudes de cambio de guardia, incluyendo guardia afectada, usuario solicitante, profesional receptor, estado de la solicitud y validaciones realizadas.

Una descripción más detallada de la información que se va a gestionar se puede consultar en el capítulo 6 de Modelo de Datos.

Capítulo 6

Modelo de datos

6.1. Introducción

En este capítulo se describirá conceptualmente el modelo de datos que empleará el sistema. Se identificarán las entidades que intervienen en el problema, sus atributos y las interrelaciones entre dichas entidades.

Para la confección del modelo, se empleará la notación del Modelo Entidad - Interrelación (modelo E-R) propuesto por Peter ChenEl esquema E-R describe la información representando los distintos elementos que la componen mediante un conjunto limitado de símbolos y reglas de relación entre ellos. Básicamente, los elementos principales del modelo son “tipo de entidad” y “tipo de interrelación”.

En las siguientes secciones se especifican con detalle los tipos de entidad y los tipos de interrelación que intervienen en el modelo, y se mostrará el diagrama E-R completo, que ofrecerá una visión global del problema.

6.2. Tipos de entidad

De acuerdo con el modelo E-R, un tipo de entidad representa a una serie de entes, objetos o personas reales o abstractos que forman parte del universo del problema a describir. Los tipos de entidad pueden ser fuertes o débiles.

- Tipo de entidad fuerte: su existencia no depende de la de otro tipo de entidad.
- Tipo de entidad débil: su existencia depende de la de otro tipo de entidad. La debilidad puede ser por existencia o por identificación.
    - Tipo de entidad débil por existencia: puede ser identificado por sí mismo a partir de sus atributos propios, pero requiere de la existencia de otro tipo de entidad del que depende.
    - Tipo de entidad débil por identificación: es un tipo de entidad débil por existencia que, además, requiere de algún atributo identificativo del tipo de entidad del que depende para poder ser identificado y diferenciado.

Las entidades de un determinado tipo de entidad se describen mediante un conjunto de atributos que representan cada una de las características o propiedades que lo describen.

Cada atributo tiene asociado un dominio de valores permitidos. Cada entidad se identifica y diferencia de forma inequívoca mediante un atributo identificador, que toma un valor único para cada entidad.

En esta sección se describirán todos los tipos de entidad que se han identificado que forman parte del problema descrito, indicando para cada uno de ellos la siguiente información:

- Descripción: definición de la entidad y función dentro del universo del problema.
- Restricción: indicación de si tiene alguna debilidad por identificación o existencia respecto de otra entidad.
- Características: se indicarán las siguientes:
  - Nombre del tipo de entidad.
  - Tipo: Fuerte o débil.
  - Atributos heredados.
  - Atributo identificador primario.
  - Atributo identificador alternativo.
  - Número de atributos, incluyendo los heredados.
- Atributos propios: por cada atributo, además del nombre del mismo, se indicará:
  - Definición: descripción del atributo.
  - Dominio: tipo de dato o valores que puede tomar el atributo.
  - Tipo: indicación de si es clave primaria o alterna, en su caso, atributo simple, etc.
  - Opcional: indicación de si el atributo puede contener un valor nulo o no.
  - Ejemplo: valor de muestra.
- Diagrama: representación gráfica del tipo de entidad, de acuerdo con la notación E-R.
- Ejemplo de entidad.

Los tipos de entidad que se han identificado y que se describirán a continuación son los siguientes:

- Tipo de entidad: Usuario
- Tipo de entidad: Trabajador
- Tipo de entidad: Especialidad
- Tipo de entidad: Guardia

6.2.1. Entidad Usuario

- Descripción: este tipo de entidad representa a los usuarios registrados en el sistema que pueden autenticarse y acceder a GuardiApp. Un usuario está asociado a un trabajador del hospital para relacionar la cuenta con la persona real.
- Restricciones: es una entidad fuerte por lo que no depende de otro tipo de entidad para existir.
- Características:
  - Nombre de la entidad: Usuario.
  - Tipo: fuerte.
  - Atributos heredados: ninguno.
  - Atributo identificador primario: id.
  - Atributo identificador alternativo: email.
  - Número de atributos: 6 propios.
- Atributos propios:
  - id
    - Definición: identificador numérico que permite distinguir de forma única a cada usuario.
    - Dominio: números enteros mayores que 0.
    - Tipo: clave primaria.
    - Opcional: no.
    - Ejemplo: 1.
  - name
    - Definición: nombre visible del usuario.
    - Dominio: conjunto de hasta 150 caracteres.
    - Tipo: atributo simple.
    - Opcional: no.
    - Ejemplo: Tomás de Aquino.
  - email
    - Definición: correo electrónico utilizado para identificar y autenticar al usuario.
    - Dominio: conjunto de hasta 250 caracteres con formato email.
    - Tipo: clave alterna (única).
    - Opcional: no.
    - Ejemplo: santotomas@alu.medac.es.
  - password
    - Definición: contraseña del usuario almacenada de forma segura.
    - Dominio: conjunto de caracteres (hash).
    - Tipo: atributo simple.

- Opcional: no.
- Ejemplo: $2y$10$....

- avatarUrl
  - Definición: dirección o ruta del avatar asociado al usuario.
  - Dominio: conjunto de hasta 255 caracteres (URL o ruta).
  - Tipo: atributo simple.
  - Opcional: sí.
- worker_id
  - Definición: identificador del trabajador al que está asociada la cuenta de usuario.
  - Dominio: números enteros mayores que 0.
  - Tipo: clave foránea.
  - Opcional: sí (según diseño; se permite usuario sin trabajador asociado).
  - Ejemplo: 12.

- Diagrama (Figura 6.1):

Figura 6.1: Entidad Usuario

- Ejemplo de entidad (Tabla 6.1):

Usuario	
Atributo	**Valor**
id	1
name	Tomás de Aquino
email	santotomas@alu.medac.es
password	$2y$10$...
avatarUrl	https://.../avatar.png
worker_id	12

Tabla 6.1: Ejemplo de la entidad *Usuario*

6.2.2. Entidad Trabajador

- Descripción: este tipo de entidad representa a los profesionales/trabajadores del hospital que pueden ser asignados a guardias. Almacena información laboral relevante, incluida la especialidad a la que pertenece.
- Restricciones: es una entidad fuerte por lo que no depende de otro tipo de entidad para existir.
- Características:
  - Nombre de la entidad: Trabajador.
  - Tipo: fuerte.
  - Atributos heredados: ninguno.
  - Atributo identificador primario: id.
  - Atributo identificador alternativo: ninguno.
  - Número de atributos: 6 propios.
- Atributos propios:
  - id
    - Definición: identificador numérico único del trabajador.
    - Dominio: números enteros mayores que 0.
    - Tipo: clave primaria.
    - Opcional: no.
    - Ejemplo: 12.
  - name
    - Definición: nombre completo del trabajador.
    - Dominio: conjunto de hasta 150 caracteres.
    - Tipo: atributo simple.
    - Opcional: no.
    - Ejemplo: Tomás de Aquino.
  - rank
    - Definición: categoría/rango profesional del trabajador.
    - Dominio: conjunto de hasta 100 caracteres.
    - Tipo: atributo simple.
    - Opcional: no.
    - Ejemplo: Adjunta.
  - registration_date
    - Definición: fecha de alta o incorporación del trabajador al centro.
    - Dominio: fecha (YYYY-MM-DD).
    - Tipo: atributo simple.
    - Opcional: no.

    - Ejemplo: 2016-09-01.
  - discharge_date
    - Definición: fecha de baja del trabajador en el centro, si aplica.
    - Dominio: fecha (YYYY-MM-DD).
    - Tipo: atributo simple.
    - Opcional: sí.
    - Ejemplo: 2025-06-30.
  - id_speciality
    - Definición: especialidad a la que pertenece el trabajador.
    - Dominio: números enteros mayores que 0.
    - Tipo: clave foránea.
    - Opcional: no.
    - Ejemplo: 3.

- Diagrama (Figura 6.2):

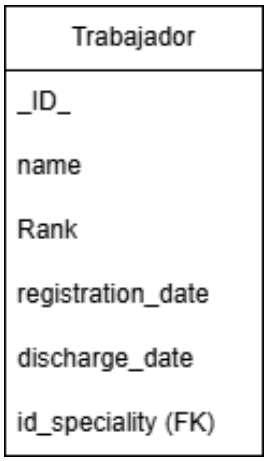

Figura 6.2: Entidad Trabajador

- Ejemplo de entidad (Tabla 6.2):

Trabajador	
Atributo	**Valor**
id	12
name	Tomás de Aquino
rank	Adjunta
registration_date	2016-09-01
discharge_date	
id_speciality	3

Tabla 6.2: Ejemplo de la entidad *Trabajador*

6.2.3. Entidad Especialidad

- Descripción: este tipo de entidad representa las especialidades o secciones del hospital (por ejemplo, urgencias, radiología, ginecología). Permite agrupar trabajadores y guardias por área.

- Restricciones: es una entidad fuerte por lo que no depende de otro tipo de entidad para existir.

- Características:

  - Nombre de la entidad: Especialidad.
  - Tipo: fuerte.
  - Atributos heredados: ninguno.
  - Atributo identificador primario: id.
  - Atributo identificador alternativo: ninguno.
  - Número de atributos: 3 propios.

- Atributos propios:

  - id
    - Definición: identificador numérico único de la especialidad.
    - Dominio: números enteros mayores que 0.
    - Tipo: clave primaria.
    - Opcional: no.
    - Ejemplo: 3.
  - name
    - Definición: nombre de la especialidad.
    - Dominio: conjunto de hasta 150 caracteres.
    - Tipo: atributo simple.
    - Opcional: no.
    - Ejemplo: Urgencias.
  - active
    - Definición: indica si la especialidad está activa en el sistema.
    - Dominio: 1 (Activa), 0 (Inactiva).
    - Tipo: atributo simple.
    - Opcional: no.
    - Ejemplo: 1.

- Diagrama (Figura 6.3):

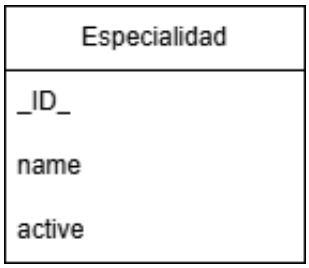

Figura 6.3: Entidad Especialidad

- Ejemplo de entidad (Tabla 6.3):

Especialidad	
Atributo	**Valor**
id	3
name	Urgencias
active	1

Tabla 6.3: Ejemplo de la entidad *Especialidad*

6.2.4. Entidad Guardia

- Descripción: este tipo de entidad representa una guardia asignada en una fecha concreta. Contiene el tipo de guardia, la especialidad a la que pertenece, el trabajador asignado y el jefe de guardia cuando se haya definido.
- Restricciones: es una entidad fuerte por lo que no depende de otro tipo de entidad para existir.
- Características:
  - Nombre de la entidad: Guardia.
  - Tipo: fuerte.
  - Atributos heredados: ninguno.
  - Atributo identificador primario: id.
  - Atributo identificador alternativo: ninguno.
  - Número de atributos: 6 propios.
- Atributos propios:
  - id
    - Definición: identificador numérico único de la guardia.
    - Dominio: números enteros mayores que 0.
    - Tipo: clave primaria.
    - Opcional: no.
    - Ejemplo: 120.

- date
  - Definición: fecha en la que se realiza la guardia.
  - Dominio: fecha (YYYY-MM-DD).
  - Tipo: atributo simple.
  - Opcional: no.
  - Ejemplo: 2026-03-10.
- duty_type
  - Definición: tipo de guardia asignada.
  - Dominio: valores enumerados (por ejemplo: CA, PF, LOC).
  - Tipo: atributo simple.
  - Opcional: no.
  - Ejemplo: PF.
- id_speciality
  - Definición: especialidad a la que pertenece la guardia.
  - Dominio: números enteros mayores que 0.
  - Tipo: clave foránea.
  - Opcional: no.
  - Ejemplo: 3.
- id_worker
  - Definición: trabajador asignado a la guardia.
  - Dominio: números enteros mayores que 0.
  - Tipo: clave foránea.
  - Opcional: no.
  - Ejemplo: 12.
- id_chief_worker
  - Definición: trabajador designado como jefe de guardia asociado a la guardia.
  - Dominio: números enteros mayores que 0.
  - Tipo: clave foránea.
  - Opcional: sí.
  - Ejemplo: 7.

- Diagrama (Figura 6.4):

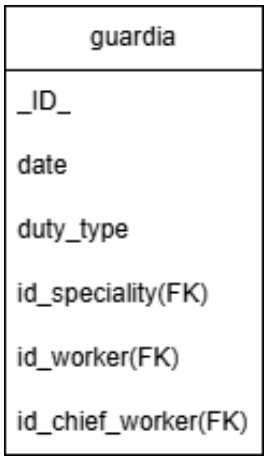

Figura 6.4: Entidad Guardia

- Ejemplo de entidad (Tabla 6.4):

Guardia	
Atributo	**Valor**
id	120
date	2026-03-10
duty_type	PF
id_speciality	3
id_worker	12
id_chief_worker	7

Tabla 6.4: Ejemplo de la entidad *Guardia*

6.3. Tipos de interrelación

En esta sección se identificarán y describirán las interrelaciones entre los tipos de entidad descritos en la sección 7.2.

Las interrelaciones pueden ser de tipo débil o fuerte.

- Un tipo de Interrelación Fuerte es aquella que representa la relación existente entre dos tipos de entidad fuertes.
- Un tipo de Interrelación Débil es aquella que representa la relación entre un tipo de entidad débil y otro fuerte o entre dos tipos de entidad débiles.

De acuerdo con la notación del modelo E-R, una interrelación se representa mediante un rombo del que parten flechas hacia los tipos de entidad que forman parte de la relación. Cada tipo de entidad interviene en la interrelación con una determinada cardinalidad, que indica el número mínimo y máximo de instancias de cada tipo de entidad que pueden participar en la interrelación. Se representa por dos valores entre paréntesis (mínimo y máximo). Las posibles cardinalidades son: (0,1),

(1,1),(0,n),(1,n),(m,n). Estas cardinalidades determinan el tipo de interrelación que se está definiendo, que puede ser:

- **1:1** Uno a uno. Cuando los dos tipos de entidad participan con una cardinalidad máxima de 1.
- **1:N o N:1** Uno a muchos, o muchos a uno. Cuando uno de los tipos de entidad participa con una cardinalidad máxima de 1 y el otro con una cardinalidad máxima de n.
- **N:N** Muchos a muchos. Cuando ambos tipos de entidad participan una cardinalidad máxima de n.

Para cada una de las interrelaciones que se han identificado en la definición del problema, se indicará la siguiente información:

- **Nombre**. Nombre del tipo de interrelación.
- **Descripción**. Definición del tipo de interrelación y de los tipos de entidad que participan en ella.
- **Tipo**. Indicación de si se trata de un tipo de interrelación débil o fuerte identificando los tipos de debilidad en su caso.
- **Cardinalidad**. Indicación de la cardinalidad del tipo de interrelación y cardinalidades mínima y máxima de los tipos de entidad intervinientes.
- **Atributos**. Indicación del número y descripción de los atributos del tipo de interrelación, en su caso.
- **Diagrama**. Representación gráfica del tipo de interrelación, de acuerdo con la notación E-R.
- **Ejemplo**. Valores de muestra.

Se han identificado las interrelaciones que se indican a continuación y que se describirán en las siguientes subsecciones:

- Tipo de Interrelación: Usuario – Trabajador
- Tipo de Interrelación: Trabajador – Especialidad
- Tipo de Interrelación: Guardia – Especialidad
- Tipo de Interrelación: Guardia – Trabajador (Asignación)
- Tipo de Interrelación: Guardia – Trabajador (Jefatura)

6.3.1. Interrelación Usuario – Trabajador

- **Nombre**: Asociado_a.
- **Descripción**: relaciona el tipo de entidad *Usuario* con el tipo de entidad *Trabajador*, indicando que una cuenta de usuario puede estar asociada a un trabajador del hospital.
- **Tipo**: interrelación fuerte (participan dos entidades fuertes).
- **Cardinalidad**: 1:1.
  - Usuario: (0,1) (un usuario puede no estar asociado a ningún trabajador o estar asociado a uno).
  - Trabajador: (0,1) (un trabajador puede no tener usuario asociado o tener uno).
- **Atributos**: no presenta atributos propios.
- **Diagrama**: se representará en la figura correspondiente del diagrama E-R.
- **Ejemplo**: el usuario con id=5 está asociado al trabajador con id=12.

6.3.2. Interrelación Trabajador – Especialidad

- **Nombre**: Pertenece_a.
- **Descripción**: relaciona el tipo de entidad *Trabajador* con el tipo de entidad *Especialidad*, indicando que cada trabajador pertenece a una única especialidad.
- **Tipo**: interrelación fuerte.
- **Cardinalidad**: N:1 (Trabajador – Especialidad).
  - Trabajador: (1,1) (cada trabajador pertenece obligatoriamente a una especialidad).
  - Especialidad: (0,n) (una especialidad puede tener cero o muchos trabajadores).
- **Atributos**: no presenta atributos propios.
- **Diagrama**: se representará en la figura correspondiente del diagrama E-R.
- **Ejemplo**: el trabajador con id=12 pertenece a la especialidad con id=3.

6.3.3. Interrelación Guardia – Especialidad

- **Nombre**: Corresponde_a.
- **Descripción**: relaciona el tipo de entidad *Guardia* con el tipo de entidad *Especialidad*, indicando que cada guardia se asigna a una especialidad concreta.
- **Tipo**: interrelación fuerte.
- **Cardinalidad**: N:1 (Guardia – Especialidad).
  - Guardia: (1,1) (cada guardia está asociada obligatoriamente a una especialidad).
  - Especialidad: (0,n) (una especialidad puede tener cero o muchas guardias).
- **Atributos**: no presenta atributos propios.
- **Diagrama**: se representará en la figura correspondiente del diagrama E-R.
- **Ejemplo**: la guardia con id=120 corresponde a la especialidad con id=3.

6.3.4. Interrelación Guardia – Trabajador (Asignación)

- **Nombre**: Asignada_a.
- **Descripción**: relaciona el tipo de entidad *Guardia* con el tipo de entidad *Trabajador*, indicando qué trabajador realiza una guardia concreta.
- **Tipo**: interrelación fuerte.
- **Cardinalidad**: N:1 (Guardia – Trabajador).
  - Guardia: (1,1) (cada guardia debe estar asignada a un trabajador).
  - Trabajador: (0,n) (un trabajador puede tener cero o muchas guardias asignadas).
- **Atributos**: no presenta atributos propios.
- **Diagrama**: se representará en la figura correspondiente del diagrama E-R.
- **Ejemplo**: la guardia con id=120 está asignada al trabajador con id=12.

6.3.5. Interrelación Guardia – Trabajador (Jefatura)

- **Nombre**: Jefe_de_guardia.
- **Descripción**: relaciona el tipo de entidad *Guardia* con el tipo de entidad *Trabajador*, indicando qué trabajador actúa como jefe de guardia en una guardia determinada. Esta asignación puede no existir en el momento de creación de la guardia.

- **Tipo**: interrelación fuerte.
- **Cardinalidad**: N:1 (Guardia – Trabajador).
  - Guardia: (0,1) (una guardia puede no tener jefe asignado o tener uno).
  - Trabajador: (0,n) (un trabajador puede ser jefe de guardia en cero o muchas guardias).
- **Atributos**: no presenta atributos propios.
- **Diagrama**: se representará en la figura correspondiente del diagrama E-R.
- **Ejemplo**: la guardia con id=120 tiene como jefe al trabajador con id=7.

6.4. Diagrama del Modelo Entidad-Interrelación

El diagrama completo se muestra en la figura 6.5

Figura 6.5: Diagrama del modelo Entidad - Interrelación

Capítulo 7

Análisis funcional

7.1. Introducción

Este capítulo identificará a los tipos de actores que podrán interactuar con el sistema informático que se desea desarrollar. A continuación, se utilizarán los casos de uso y los diagramas de secuencia para describir las acciones que podrán realizar los diferentes actores.

7.2. Actores

Un actor es una representación de una persona, proceso o entidad externa que interactúa con el sistema. Se van a considerar los siguientes tipos de actores:

- **Administrador**
  - Usuario registrado en el sistema con control completo sobre la aplicación.
  - Tiene competencias exclusivas para la gestión de usuarios, importación de guardias, configuración de tarifas, cierre y reapertura de meses, generación de documentos y auditoría.
- **Usuario de consulta**
  - Usuario registrado con permisos limitados.
  - Puede consultar información relacionada con guardias, vistas diaria y mensual, búsquedas y resúmenes de horas e importes, sin capacidad de modificación.

7.3. Casos de uso

Los casos de uso describen las acciones que pueden desarrollar los actores del sistema. Se ha identificado los siguientes casos de uso principales, que son descritos

en las secciones que se indican:

- **CU-0. Diagrama de contexto** (Sección 7.3.1)
- **CU-1. Consultar información de guardias** (Sección 7.3.2)
- **CU-2. Administrar sistema de guardias**(Sección ??)

7.3.1. Caso de Uso 0. Diagrama de Contexto

El diagrama de contexto engloba los casos de uso principales que componen el sistemas. Véanse la Figura 7.1 y la Tabla 7.1.

Figura 7.1: Diagrama de Contexto

Tabla 7.1: CU-0. Diagrama de contexto

Caso de uso 0 - Diagrama de contexto	
Actores	Administrador
Descripción	Acciones principales del sistema
Precondiciones	El usuario debe acceder a la página inicial de la aplicación
Casos de uso	CU-1. CU-2. Administrar
Flujo principal	1a. El usuario público accede a la aplicación sin necesidad de iniciar sesión y podrá. 1b. El resto de usuarios se deberá identificar para acceder a los módulos correspondientes dentro de la aplicación.
Flujo alternativo	1. La aplicación no se carga correctamente. 2. Se informa del error al usuario. 3. Se informa al usuario que puede intentar cargar de nuevo la aplicación.

7.3.2. CU-1. Consultar información de guardias

El caso de uso *CU-1. Consultar información de guardias* describe las acciones que puede realizar un usuario de consulta o un administrador para visualizar información del sistema.

Este caso de uso está compuesto por los siguientes subcasos:

- CU-1.1. Consultar guardias por mes y especialidad.
- CU-1.2. Consultar guardias de un profesional.
- CU-1.3. Visualizar vista diaria de guardias.

Figura 7.2: CU-1.

Tabla 7.2: CU-1. Consultar información de guardias

Caso de uso 1 - Consultar información de guardias	
Actores	Usuario de consulta, Administrador
Descripción	Consulta de información relacionada con las guardias
Precondiciones	El usuario debe estar autenticado en el sistema
Casos de uso	CU-1.1, CU-1.2, CU-1.3
Flujo principal	1. El usuario selecciona una opción de consulta. 2. El sistema solicita los datos al backend. 3. El sistema muestra la información solicitada.
Flujo alternativo	1. Error en la obtención de datos. 2. El sistema informa del error al usuario.

7.4. Validación de casos de uso

La tabla 7.3 permite comprobar que los casos de uso cubren todos los requisitos funcionales de la aplicación web propuestos en la Sección 5.4.1.

Tabla 7.3: Tabla validación casos de uso

Requisito funcional	**Caso de uso**
RF-1	CU 1.1
RF-8	CU 2.1

7.5. Diagrama de secuencia

El diagrama de secuencia es una representación gráfica que pretende dar una visión de las acciones que se realizarán durante la ejecución de alguna operación en el sistema. A continuación, se muestran los diagramas de secuencias para la creación (Figura 7.3), búsqueda (Figura 7.5), modificación (Figura 7.6) y borrado (Figura 7.4) de instancias genéricas () que se corresponderán con...

Figura 7.3: Diagrama de secuencia de crear

Figura 7.4: Diagrama de secuencia de borrar

Figura 7.5: Diagrama de secuencia de buscar

Figura 7.6: Diagrama de secuencia de modificar

Parte IV

Diseño

Capítulo 8

Diseño de datos

8.1. Introducción

En este capítulo se definirán las estructuras de datos que conforman el sistema a partir de los elementos identificados durante el análisis de datos del Capítulo 7 (tipos de entidad y tipos de interrelación). Para ello, se llevarán a cabo los siguientes pasos:

- Obtención del modelo relacional. Definición de la estructuras (tablas) del modelo de datos.
- Normalización del modelo. Refinamiento del modelo, para la eliminación de errores de integridad.
- Obtención del esquema relacional.
- Diagrama relacional.

8.2. Modelo Relacional

A partir del Modelo Entidad - Interrelación descrito en el capítulo 7, se pueden obtener las tablas o relaciones del Modelo Relacional utilizando las reglas de transformación (RTECAR - véase el capítulo 5 de Bases de Datos. Desde Chen hasta Codd con ORACLE.[1]). En concreto, se han aplicado las siguientes reglas de transformación:

- RTECAR-1: Todos los tipos de entidad presentes en el esquema conceptual se transformarán en tablas o relaciones en el esquema relacional manteniendo el número y tipo de atributos, así como la característica de identificador de esos atributos.
- RTECAR-3.1: En las relaciones 1:N, la clave primaria de la entidad del lado 1 se convierte en una clave foránea en la tabla de la entidad del lado N.

Para cada tabla se mostrará la siguiente información:

- **Descripción**. Se describirá el origen de la tabla, indicando los elementos del modelo Entidad-Interrelación desde los que se ha obtenido.
- **Nombre de la tabla**
- **Atributos**. Se describirán los atributos que componen la tabla, distinguiendo su rol en cada caso con la siguiente notación:
  - Clave primaria
  - Clave ALTERNA, si existe
  - Claves **foráneas**, si existen
  - Resto de atributos
- **Esquema relacional**. Definición formal de la tabla, de acuerdo con el Modelo Relacional.

8.2.1. Tabla Usuario

- **Descripción**: la tabla *Usuario* se obtiene a partir de la entidad *Usuario* (modelo `User`). Almacena las credenciales y datos básicos necesarios para la autenticación y la identificación del usuario dentro del sistema. Además, se relaciona con la entidad *Profesional/Trabajador* mediante el atributo `worker_id`.
- **Nombre de la tabla**: Usuario (`users`)
- **Atributos**:
  - Clave primaria: id
  - Claves alternas: email (único)
  - Claves foráneas: **worker_id** $\rightarrow$ Trabajador(id) (si existe la tabla de trabajadores/profesionales)
  - Resto de atributos: name, password, avatarUrl, email_verified_at, remember_token, created_at, updated_at
- **Esquema relacional**: Usuario(**id**, name, *email*, password, avatarUrl, **worker_id**, email_verified_at, remember_token, created_at, updated_at)

8.3. Normalización del modelo

La normalización del modelo descrito en la sección anterior pretende detectar y corregir redundancias e inconsistencias en la información representada, para lo cual se aplicarán las medidas correctoras que garanticen que las tablas obtenidas cumplen las siguientes formas normalizadas [1].

- La tabla Usuario cumple la FN1 al almacenar valores atómicos en todos sus atributos.
- Cumple FN2 y FN3 al depender todos los atributos no clave de forma completa y no transitiva de la clave primaria.
- Además, se encuentra en FNBC, ya que los determinantes funcionales relevantes son claves candidatas (id y email).

Todas las tablas definidas se encuentran en la Primera Forma Normal, puesto que en ninguna de ellas existen atributos múltiples.

8.3.1. Tabla Usuario

- **Claves candidatas**: id
- **Normalización**: La tabla Usuario presenta una dependencia funcional formada por la clave primaria y el resto de atributos. La tabla se encuentra en FNBC: el único determinante funcional es la clave primaria, por lo que la dependencia funcional con el resto de atributos es completa.

Figura 8.1: Tabla Usuario en FNBC

8.3.2. Tabla Trabajador

- **Descripción**: la tabla *Trabajador* se obtiene a partir de la entidad *Trabajador* (modelo `Worker` de Laravel). Almacena la información laboral necesaria para la planificación de guardias, incluyendo rango/categoría, fechas de alta y baja, y la especialidad a la que pertenece.
- **Nombre de la tabla**: Trabajador (`worker`)
- **Atributos**:

  - Clave primaria: id
  - Claves alternas: —
  - Claves foráneas: **id_speciality** → Especialidad(id)
  - Resto de atributos: name, rank, registration_date, discharge_date

- **Esquema relacional**: Trabajador(**id**, name, rank, registration_date, discharge_date, **id_speciality**)

8.3.3. Tabla Trabajador

- **Claves candidatas**: id
- **Normalización**: La tabla Trabajador presenta una dependencia funcional formada por la clave primaria y el resto de atributos. La tabla se encuentra en FNBC: el único determinante funcional es la clave primaria, por lo que la dependencia funcional con el resto de atributos es completa.

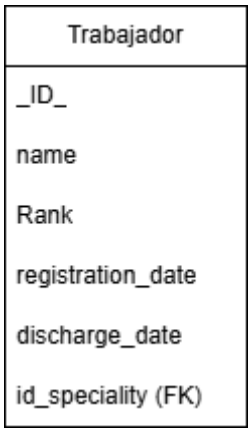

Figura 8.2: Tabla Trabajador en FNBC

8.3.4. Tabla Especialidad

- **Descripción**: la tabla *Especialidad* se obtiene a partir de la entidad *Especialidad* (modelo `Speciality` de Laravel). Almacena las secciones/especialidades del hospital a las que pertenecen los trabajadores y sobre las que se organizan las guardias.
- **Nombre de la tabla**: Especialidad (`speciality`)
- **Atributos**:
  - Clave primaria: id
  - Claves alternas: —
  - Claves foráneas: —
  - Resto de atributos: name, active
- **Esquema relacional**: Especialidad(**id**, name, active)

8.3.5. Tabla Especialidad

- **Claves candidatas**: id
- **Normalización**: La tabla Especialidad presenta una dependencia funcional formada por la clave primaria y el resto de atributos. La tabla se encuentra en FNBC: el único determinante funcional es la clave primaria, por lo que la dependencia funcional con el resto de atributos es completa.

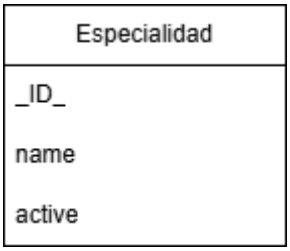

Figura 8.3: Tabla Especialidad en FNBC

8.3.6. Tabla Guardia

- **Descripción**: la tabla *Guardia* se obtiene a partir de la entidad *Guardia* (modelo `Duty` de Laravel). Representa cada guardia asignada en una fecha concreta, indicando su tipo, la especialidad asociada, el trabajador asignado y, cuando corresponda, el trabajador que actúa como jefe de guardia.
- **Nombre de la tabla**: Guardia (`duties`)
- **Atributos**:
  - Clave primaria: id
  - Claves alternas: —
  - Claves foráneas: **id_speciality** → Especialidad(id), **id_worker** → Trabajador(id), **id_chief_worker** → Trabajador(id)
  - Resto de atributos: date, duty_type
- **Esquema relacional**: Guardia(**id**, date, duty_type, **id_speciality**, **id_worker**, **id_chief_worker**)

8.3.7. Tabla Guardia

- **Claves candidatas**: id
- **Normalización**: La tabla Guardia presenta una dependencia funcional formada por la clave primaria y el resto de atributos. La tabla se encuentra en FNBC: el único determinante funcional es la clave primaria, por lo que la dependencia funcional con el resto de atributos es completa.

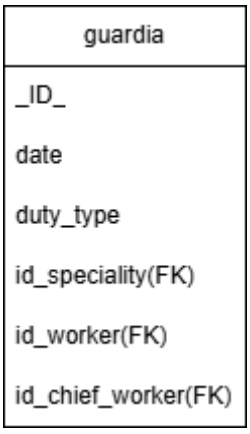

Figura 8.4: Tabla Guardia en FNBC

8.4. Esquema relacional

Tabla	Atributos
Usuario	(**id**, name, email, password, avatarUrl, **worker_id**, email_verified)
Trabajador	(**id**, name, rank, registration_date, discharge_date, **id_speciality**)
Especialidad	(**id**, name, active)
Guardia	(**id**, date, duty_type, **id_speciality**, **id_worker**, **id_chief_worker**)

8.5. Diagrama relacional

El diagrama relacional del modelo propuesto se muestra en la figura 8.5.

Figura 8.5: Diagrama relacional

Capítulo 9

Diseño arquitectónico

9.1. Introducción

9.2. Diagrama de despliegue

9.2.1. Descripción de los nodos

9.2.2. Descripción de los componentes

Capítulo 10

Diseño de la interfaz

10.1. Introducción

10.2. Características comunes

10.3. Interfaz del módulo Usuario...

10.3.1. Descripción general

hjk

10.3.2. Descripción detallada

hjk

10.3.3. Otros elementos de la interfaz

hjk

10.4. Interfaz del módulo Administrador

10.4.1. Descripción general

hjk

10.4.2. Descripción detallada

hjk

10.4.3. Otros elementos de la interfaz

hjk

Parte V

Pruebas

Capítulo 11

Pruebas

11.1. Introducción

11.2. Prueba de usuarios públicos

11.2.1. Pruebas CU-1.1

- Prueba realizada
- Resultado de la prueba
- Prueba realizada
- Resultado de la prueba

Parte VI

Conclusiones y futuras mejoras

Capítulo 12

Conclusiones

12.1. Introducción

En este capítulo se exponen la conclusiones obtenidas sobre el desarrollo del proyecto, y en relación con los objetivos planteados inicialmente en el capítulo 2 frente al resultado final alcanzado tras la finalización del mismo y las pruebas realizadas.

Con carácter general, el objetivo principal, consistente en el desarrollo de un sistema informático ...

12.2. Conclusiones específicas

12.3. Conclusiones personales y profesionales

A nivel personal, el presente proyecto nos ha permitido reciclar, actualizar ...

En cuanto al ámbito profesional, se han logrado los objetivos propuestos:

-

Capítulo 13

Futuras mejoras

13.1. Introducción

El sistema desarrollado en este proyecto es susceptible de mejoras y ampliaciones, como cualquier otro software. De hecho, se identificaron muchas oportunidades durante el desarrollo.

En las siguientes secciones se desglosan las mejoras y ampliaciones propuestas, según cada una de las categoría expuestas.

13.2. Mejoras y nuevas funciones

Basándonos en lo que se ha desarrollado hasta ahora, se identifican una serie de funciones adicionales y mejoras que se pueden realizar. Estas mejoras se describen a continuación, divididas por tipo de usuario y mejoras generales.

13.2.1. Mejoras para usuarios

- G

13.2.2. Mejoras para administrador

-

13.2.3. Mejoras de funcionamiento

-

13.2.4. Integración con otras aplicaciones

13.2.5. Exportabilidad

Capítulo 14

ANEXOS

Este apartado está destinado a incluir todo aquello que queráis añadir de manera adicional al trabajo.

Bibliografía

[1] . []

[2] Alejandro Castán Salinas. *Guía rápida de administración de MySQL*. [En línea. Última consulta:] URL: http://www.xtec.cat/~acastan/textos/Administracion%20de%20MySQL.html.

[3] Centro Formación Profesional. *Centro Formación Profesional*. [En línea. Última consulta:] URL: https://medac.es/fp-cordoba/escritora-maria-goyri-sn.

[4] Jessica Navil. *Arquitectura de MariaDB*. [En línea. Última consulta:] URL: https://prezi.com/oz6eimbnuyq4/arquitectura-de-mariadb/.

[5] Luismi Gracia. *¿Qué ofrece MongoDB Enterprise Advanced?* [En línea. Última consulta:] URL: https://unpocodejava.com/2019/07/22/que-ofrece-mongodb-enterprise-advanced/.

[6] Microsoft. *Visual Studio Code*. [En línea. Última consulta:] URL: https://code.visualstudio.com/.

[7] World Wide Web Consortium. *World Wide Web Consortium*. [En Línea. Última consulta:] URL: https://www.w3.org/.

Capítulo 15

README (BORRAR EN EL main.tex)

15.1. Ejemplos uso Overleaf comunes

Ejemplo cita de la bibliografía **MEDAC**[3] es un centro de Formación Profesional...

[4] [2] [5]

- Áreas de sistemas y departamentos de informática en cualquier sector de actividad.
- Sector de servicios tecnológicos y comunicaciones.
- Área comercial con gestión de transacciones por Internet.

Texto escrito en cursiva Texto escrito en letras rectas *Texto roman de estilo inclinado* Texto escrito en mayúsculas pequeñas

15.2. Justificación

El presente documento describe la guía de trabajo a seguir, para la correcta elaboración del documento sobre el que se soporta el desarrollo del módulo Proyecto, del segundo curso del C.F.G.S de Desarrollo de Aplicaciones Web.

Este módulo profesional complementa la formación de otros módulos profesionales en las funciones de análisis del contexto, diseño y organización de la intervención y planificación de la evaluación de la misma.

Así mismo, teniendo en cuenta que las actividades profesionales asociadas a estas funciones se aplican en:

- Áreas de sistemas y departamentos de informática en cualquier sector de actividad.

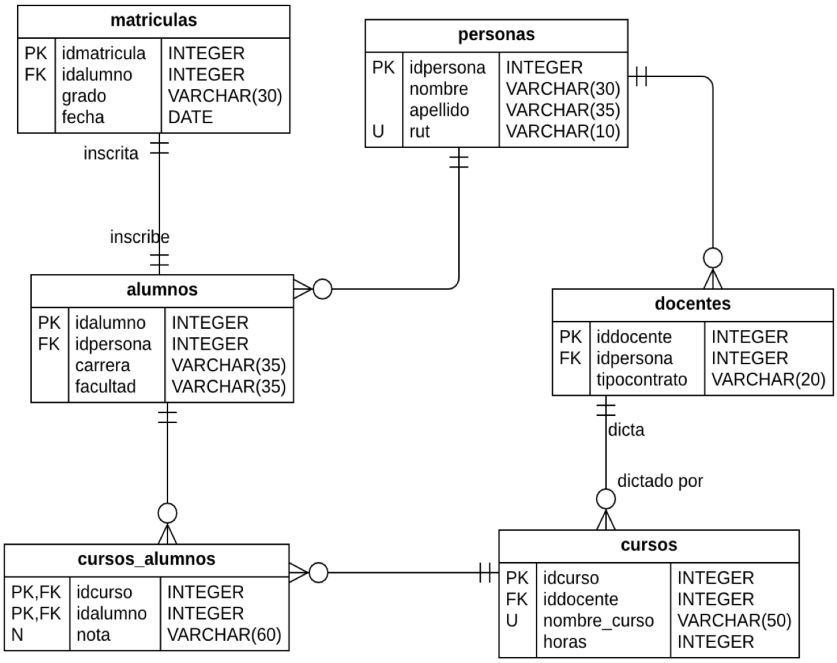

Figura 15.1: Ejemplo de adjuntar diagrama ER

- Sector de servicios tecnológicos y comunicaciones.
- Área comercial con gestión de transacciones por Internet.

Y, por otra parte, las líneas de actuación en el proceso de enseñanza aprendizaje que permiten alcanzar los objetivos del módulo versarán sobre:

- La ejecución de trabajos en equipo.
- La autoevaluación del trabajo realizado.
- La autonomía y la iniciativa.
- El uso de las TIC.

Por estos motivos se propone un Proyecto Integrado orientado al desarrollo de forma autónoma de nuevos productos y servicios innovadores, así como al autoempleo, con el propósito de que el alumno aplique de forma eficaz y realista, los conocimientos y destrezas adquiridos durante el ciclo. Este proyecto deberá seguir el índice de contenidos que se expone en el punto 3.

15.3. Formato

Los alumnos deberán hacer la entrega del proyecto siguiendo la siguiente normativa de manera obligatoria:

- Incluir Índice y Portada (disponible en la Plataforma Virtual).
- La Portada debe contener el nombre del proyecto (de forma clara, precisa y representativa del trabajo), nombre del alumno, curso académico, nombre del módulo y nombre del Centro y su logotipo.

El índice ha de ser automático, es decir, que ayude a localizar los distintos apartados del trabajo.

- Tras el índice deberá aparecer una página de cortesía.
- Enumerar páginas a partir del primer apartado, es decir excluir índice y página de cortesía.
- Justificar párrafos.
- Enumerar los distintos apartados.
- Corregir ortografía y gramática (no cometer faltas de ortografía, cuidar la expresión, etc.).
- Referenciar según normas APA.
- Utilizar la plantilla Word de Medac (disponible en la Plataforma).
- Tipo de letra: Times New Roman 12
- Interlineado 1,5.
- La extensión del proyecto debe ser de mínimo 18 páginas y máximo 30 páginas (Sin incluir portada, índice y anexos).

15.4. Orientaciones sobre evaluación

A parte de los criterios de evaluación dispuestos en la Real Decreto de la titulación, el tutor del proyecto evaluará el trabajo en base a los siguientes criterios de evaluación referentes a la realización del mismo, así como a la actitud del alumno ante su elaboración y trabajo en equipo:

- Originalidad de la idea.
- Inclusión de los contenidos reflejados en el índice.
- Cumplimiento de las normas de formato de entrega, descritas en esta programación.
- Coherencia y argumentación del contenido.
- Que el proyecto evidencie que el alumno ha adquirido conocimientos propios de su titulación.

- Puntualidad y asistencia tanto a las clases teóricas de Proyecto como a las tutorías propuestas por el tutor.
- Nivel de responsabilidad observado.
- Grado de iniciativa mostrado por el alumno.
- Predisposición para el trabajo en equipo.
- Compañerismo y respeto con todos los agentes implicados en la elaboración del proyecto: compañeros y tutores.

Por último, se detallan a continuación los criterios de evaluación sobre los que se apoyará el tribunal ante la defensa del proyecto:

- Originalidad en la presentación.
- Correcta expresión oral y corporal.
- Dominio de todo el contenido del proyecto.
- Indumentaria formal y acorde para la ocasión.
- Ajuste al tiempo indicado para la defensa.
- Trato respetuoso hacia todos los presentes en el aula durante la defensa.

Recordaros que aquellos trabajos en los que se detecte plagio, no serán evaluados, debiendo el alumnado entregar ese trabajo de nuevo en el periodo de recuperaciones correspondiente. La diferencia entre algo que está literalmente copiado y algo que ha sido citado o en lo que el alumnado se ha basado para elaborar su trabajo es clara. MEDAC persigue una evaluación justa y limpia persiguiendo la correcta competencia entre el alumnado y como tal evaluará.