EPIGRAM
ADAS.health

Manual Técnico de Software

X	ORIGINAL
	COPIA CONTROLADA
	COPIA NO CONTROLADA

MAN-SW-01	
ELABORADO R. SISTEMAS DE GESTIÓN	REVISADO Y APROBADO
FECHA: 2026	FECHA: 2026

EPIGRAM
TECHNOLOGIES

ÍNDICE

1. Introducción 6
1.1 Propósito del manual 6
1.2 Alcance del software 6
1.2.1 Objetivo del manual 6
1.2.2 Alcance del software 7
1.2.3 Público objetivo 7
1.2.4 Versiones del documento 7
2. Requerimientos del sistema 8
2.1 Hardware Necesario 8
2.1.1 Servidor central (para instalaciones locales o híbridas) 8
2.1.2 Servidor de grabación de cámaras (opcional) ..... **¡Error! Marcador no definido.**
2.2 Software de Base o Dependencias 8
2.3 Configuraciones Recomendadas 9
3. Arquitectura del Software 9
3.1 Diseño General y Componentes 9
3.2 Diagramas de Arquitectura 10
3.3 Relaciones entre Módulos 11
3.4 Diagramas de la Estructura General 11
3.5 Componentes y Módulos 12
3.6 Descripción de las Funcionalidades 13
3.7 Diseño de unidades de software 14
3.8 Interfaces entre unidades 17
3.9 Consideraciones de seguridad 18
4. Instalación y despliegue del sistema 18
4.1 Instalación del software 18
4.2 Configuración inicial del sistema 19

4.3 Resolución de problemas durante la instalación 20
4.4 Instalación por entornos 21
4.4.1 Entorno de desarrollo 21
4.4.2 Entorno de pruebas 21
4.4.3 Entorno de producción 22
4.5 Despliegue continuo y automatización 22
4.6 Gestión de configuración del software 23
5. Estructura del Código Fuente 24
5.1 Organización del repositorio 24
5.2 Descripción de los módulos y paquetes 24
5.3 Convenciones de codificación y estilo 25
5.4 Guía de nombres de archivos y directorios 25
6. Bases de Datos 25
6.1 Modelo de datos 25
6.1.1 Diagrama entidad-relación (ERD) 25
6.1.2 Descripción de tablas y relaciones 26
6.2 Scripts de base de datos ..... **¡Error! Marcador no definido.**
6.2.1 Creación de bases de datos ..... **¡Error! Marcador no definido.**
6.2.2 Migraciones y actualizaciones 26
6.3 Conexión y configuración 26
7. Documentación de Código 26
7.1 Herramientas de documentación 26
7.2 Generación de documentación automática 27
7.3 Mejores prácticas para comentarios y documentación 27
8. Descripción de módulos y funciones 27
8.1 Funcionalidad de cada componente 27
8.2 Relación con otros módulos 28
8.3 Parámetros y opciones de configuración 28
9. Interfaz de Usuario 28
9.1 Capturas de Pantalla y Explicaciones 28
9.2 Navegación a Través de la Aplicación 29

9.3 Consejos para el Uso Eficiente de la Interfaz 30
10. APIs y Servicios Externos 30
10.1 Documentación de APIs Internas y Externas 30
10.2 Ejemplos de Código y Uso 33
10.3 Directrices para la Integración con Otros Servicios 33
11. Seguridad 34
11.1 Medidas de seguridad implementadas 34
11.2 Guía de buenas prácticas de seguridad. Recomendaciones para mantener la seguridad 34
11.3 Pautas para la gestión de usuarios y permisos. Procedimientos en caso de vulnerabilidades detectadas 34
11.3.1 Gestión de claves y certificados 34
11.3.2 Control de acceso y roles 35
11.3.3 Manejo de datos sensibles 35
11.3.4 Procedimientos en caso de vulnerabilidades 35
12. Ciclo de vida y desarrollo del software 35
12.1 Desarrollo del software 35
12.2 Modelo de desarrollo 36
12.3 Roles y responsabilidades 36
12.4 Gestión de requisitos 37
12.5 Gestión del desarrollo 37
12.6 Gestión de versiones 38
12.7 Gestión de cambios 38
12.8 Liberación del software 39
13. Pruebas 39
13.1 Objetivo de la verificación 40
13.2 Alcance de la verificación 40
13.3 Métodos de verificación 40
13.4 Tipos de Pruebas Realizadas 40
13.5 Criterios de aceptación 41
13.6 Plan de pruebas del software 42

13.7 Ejecución y registro de pruebas 42
13.8 Responsables de verificación 42
14. Mantenimiento y actualizaciones 43
14.1 Objetivo y alcance del mantenimiento 43
14.2 Tipos de mantenimiento 43
14.3 Gestión de incidencias y errores 44
14.4 Implementación de cambios 44
14.5 Verificación y pruebas tras mantenimiento 45
14.6 Liberación de versiones tras mantenimiento 45
14.7 Procedimientos para el mantenimiento rutinario 46
14.8 Aplicación de actualizaciones y parches 46
14.9 Migración de datos entre versiones 47
14.10 Buenas prácticas para el mantenimiento 48
14.11 Puntos de contacto para soporte técnico 48
14.12 Procedimiento de actualizaciones y parches 48
15. Resolución de problemas 49
15.1 Gestión de incidencias 49
15.2 Diagnóstico y solución de errores comunes 49
15.3 Preguntas frecuentes y sus respuestas 50
15.4 Canales de soporte técnico 50
16. Glosario 50
17. Bibliografía 51
18. Licencias y créditos de componentes de terceros 51
19. Comunidad de foros y soporte 51
20. Anexos 51
21. Historial de cambios 52
Anexo 1a – Ejemplos de Código y Uso 52
Ejemplo 1. Obtener Listado Compacto de Displays 52
Ejemplo 2. Creación de un nuevo punto de cuidado 54

1. Introducción

1.1 Propósito del manual

Este manual técnico tiene como finalidad servir como guía integral para la instalación, configuración, operación y mantenimiento del sistema **Epigram ADAS Health**, un software diseñado para optimizar la gestión de salas hospitalarias y la información asociada a los pacientes.

El sistema **Epigram ADAS Health** permite digitalizar procesos tradicionales de coordinación logística, brindando una plataforma centralizada, visual e interactiva para acceder a información organizativa y logística en tiempo real. Mediante el uso de pantallas táctiles, estaciones de trabajo y dispositivos móviles, el personal puede consultar y actualizar el estado de camas, asignaciones, avisos y otros datos relevantes.

Este manual está dirigido principalmente a personal técnico y administrativo, pero también puede ser útil para usuarios clínicos que participen en la administración o supervisión del sistema, con el objetivo de garantizar una implementación efectiva y una operación segura en el entorno hospitalario.

1.2 Alcance del software

1.2.1 Objetivo del manual

El presente manual técnico tiene como objetivo proporcionar una guía **clara, detallada y estructurada** para la instalación, configuración, uso y mantenimiento del software médico Epigram ADAS Health. Su propósito es **garantizar un uso eficiente y seguro** del sistema, facilitando la labor del personal sanitario y técnico, y optimizando el control de las salas y el seguimiento de pacientes.

Este manual ofrece una descripción detallada y estructurada del software **Epigram ADAS Health**, abordando:

- Arquitectura técnica y componentes principales.
- Requisitos de hardware y software para su despliegue.
- Procedimiento de instalación y configuración.
- Módulos funcionales disponibles y su uso.
- Buenas prácticas de operación, mantenimiento y seguridad.

El objetivo es proporcionar a los equipos responsables toda la información necesaria para desplegar, mantener y evolucionar el sistema de forma eficaz, garantizando un entorno seguro y funcional.

1.2.2 Alcance del software

Epigram ADAS Health es un software como producto sanitario (MDSW) destinado a la visualización, organización y actualización de información administrativa y organizativa relacionada con la ocupación y estado operativo de salas en entornos hospitalarios o sanitarios. El sistema facilita la coordinación interna mediante la gestión administrativa de asignaciones y estados operativos, proporcionando una visión estructurada del entorno organizativo.

Epigram ADAS Health no realiza funciones de diagnóstico, no proporciona recomendaciones terapéuticas, no procesa ni interpreta datos clínicos, no monitoriza pacientes ni influye en decisiones diagnósticas o terapéuticas. El software está destinado a ser utilizado por personal sanitario y administrativo autorizado y debidamente formado.

Epigram ADAS Health está clasificado como producto sanitario de clase I conforme al Reglamento (UE) 2017/745, con uso no invasivo y prolongado. Además, permite la integración con otros sistemas hospitalarios (HIS, LDAP, dispositivos médicos) y se adapta a diferentes escalas de implementación.

1.2.3 Público objetivo

El manual está dirigido a los siguientes perfiles:

- **Personal técnico**: Encargado de la instalación, configuración, integración, mantenimiento y soporte del sistema **Epigram ADAS Health**
- **Profesionales de salud** (médicos y personal de enfermería): Responsables del registro y consulta de la información relativa a salas y pacientes.
- **Personal de servicios generales** (limpieza, mantenimiento): Usuarios que requieren acceder a la información del sistema para planificar su actividad conforme al estado operativo de las salas.
- **Administradores del sistema**: Responsables de la gestión de usuarios, permisos y configuración avanzada del sistema.

1.2.4 Versiones del documento

FECHA	Nº VERSIÓN	SECCIONES MODIFICADAS	NATURALEZA DEL CAMBIO

2. Requerimientos del sistema

La correcta operación de **Epigram ADAS Health** requiere de una infraestructura adecuada tanto a nivel de hardware como de software. Este apartado describe los requisitos mínimos y recomendados para el despliegue eficiente del sistema, así como las configuraciones necesarias para garantizar un entorno seguro, estable y conforme a los requisitos técnicos, de seguridad y de calidad del software.

2.1 Hardware Necesario

2.1.1 Servidor central (para instalaciones locales o híbridas)

El servidor central se encarga del procesamiento de datos, gestión de usuarios, procesamiento de información administrativa y almacenamiento de datos. Debe cumplir con las siguientes especificaciones mínimas:

- **Procesador**: Intel Xeon (mínimo 4 núcleos) o equivalente AMD EPYC.
- **Memoria RAM**: 16 GB (recomendado: 32 GB para instalaciones medianas o grandes).
- **Almacenamiento:** 500 GB SSD (preferentemente NVMe para velocidad de acceso)
- **Conectividad:** Gigabit Ethernet o superior.
- **Redundancia energética:** Suministro eléctrico ininterrumpido (UPS) y sistema de respaldo (RAID/backup NAS).

Nota: Para entornos que incluyan **almacenamiento de imágenes y vídeos** de UCI de alta resolución, se recomienda un almacenamiento adicional de **24 TB** (discos configurados en RAID 5/6) para garantizar la disponibilidad y la seguridad de los datos destinados a supervisión operativa.

2.2 Software de Base o Dependencias

Sistema Operativo del servidor

Epigram ADAS Health puede desplegarse en los siguientes sistemas operativos:

- **Windows Server 2019** o superior
  Recomendado si se integra con Active Directory o servicios Microsoft.
- **Ubuntu Server 20.04 LTS** o superior
  Ideal para entornos de bajo consumo, contenedores y soluciones open source.

Sistema Operativo en clientes

- **Windows 10 o superior**
- **Navegadores compatibles**: Google Chrome, Microsoft Edge, Mozilla Firefox (últimas versiones).

Dependencias del sistema

- **.NET 6.0 o superior** (core del backend y microservicios).
- **.NET Framework 4.8** (si se ejecuta en entornos Windows clásicos).
- **Servidor web**:
  - IIS (entorno Windows).
  - Apache o Nginx (Linux).
- **Base de datos compatibles**:
  - PostgreSQL 13 o superior.
  - Microsoft SQL Server 2019 o superior.
- **Otras dependencias opcionales (según configuración avanzada)**:
  - **Node.js** (microservicios, web sockets).
  - **Python 3.x** (scripts de integración).
  - **Docker** y **Docker Compose** (despliegue en contenedores).
  - **EF Core** o **Dapper** para acceso a base de datos.

2.3 Configuraciones Recomendadas

Para garantizar rendimiento, disponibilidad y seguridad, se recomienda:

- **Resolución mínima recomendada**: 1920 x 1080 (Full HD), idealmente con monitores ≥ 21".
- **Pantallas táctiles** para zonas operativas del hospital con fines organizativos (control de pacientes, quirófanos, supervisión operativa).
- **Conexión cableada** en estaciones críticas (reducción de latencia y mayor estabilidad).
- **Backups automáticos** diarios con pruebas de restauración semanales.
- **Auditoría de logs** con revisión semanal por parte del personal técnico.
- **Autenticación integrada** con el sistema del hospital (LDAP/Active Directory).
- **Roles de usuario** bien definidos: acceso mínimo necesario para cada perfil.
- **Políticas de seguridad activas**:
  - Firewall con reglas por servicio.
  - Antivirus y antimalware actualizados.
  - Cifrado en tránsito mediante HTTPS (TLS 1.2 o superior).
  - Cifrado en reposo para información sensible (por ejemplo, base de datos con TDE o discos cifrados).

3. Arquitectura del Software

3.1 Diseño General y Componentes

El sistema **Epigram ADAS Health (adas-core)** está desarrollado siguiendo el enfoque de **Arquitectura Limpia (Clean Architecture)**, que permite separar las responsabilidades en capas bien definidas, independientes y desacopladas. Esta estructura garantiza una mayor

mantenibilidad, escalabilidad, capacidad de pruebas y adaptabilidad ante nuevas funcionalidades.

Gracias a su enfoque modular, el sistema puede incorporar nuevos componentes funcionales sin impactar negativamente la estructura general del proyecto. Esto facilita la evolución del sistema manteniendo su estabilidad, integridad y seguridad operativa.

Modelo de ciclo de vida aplicado

El ciclo de desarrollo adoptado combina elementos del **Modelo Ágil** y el **Modelo Incremental**, permitiendo entregas frecuentes de nuevas funcionalidades, con validación continua y mejora progresiva del producto.

- **Modelo Ágil**: Favorece iteraciones controladas, validación continua y adaptación a los requerimientos operativos, manteniendo la calidad, seguridad y trazabilidad del sistema.
- **Modelo Incremental**: Permite añadir nuevos módulos funcionales de forma aislada y escalonada (por ejemplo: integración progresiva de sensores, relés, autenticadores, etc.), conservando siempre la integridad del sistema principal.

3.2 Diagramas de Arquitectura

La arquitectura del sistema se organiza en múltiples capas, cada una con responsabilidades bien definidas:

- **Capa de Aplicación**

Contiene los casos de uso y la lógica de orquestación de alto nivel. Es la encargada de coordinar acciones entre los módulos internos (adas-core`.Application`).

- **Capa de Dominio**

Define las entidades centrales y las reglas de negocio puras, independientes de cualquier tecnología (`adas-core.Domain`). Constituye el núcleo del sistema.

- **Capa de Infraestructura**

Gestiona el acceso a recursos externos como bases de datos, dispositivos conectados o sistemas externos, y servicios de autenticación. (`adas-core.Infrastructure`).

- **Capa de Módulos Funcionales o Componentes funcionales específicos**

Incluye funcionalidades específicas desacopladas, como el control de relés, balizas, dispositivos intermedios (`Relays`, `LightBeacons`, `ProxyDevices`).

- **Módulos de Autenticación**

Secciones especializadas que gestionan la autenticación y autorización, ya sea mediante LDAP o login local (`apiLdapLogin`, `apiLocalLogin`).

- **Capa de Pruebas**

Contiene pruebas unitarias, de integración y pruebas de regresión automatizadas (`adas-core.Test`), como parte del aseguramiento de calidad del software.

3.3 Relaciones entre Módulos

La arquitectura sigue el principio de inversión de dependencias, imponiendo que todas las dependencias se dirijan hacia las capas internas del sistema, cuyo núcleo es la capa de Dominio (Domain):

- `Application` depende de `Domain` pero nunca al revés.
- `Infrastructure` depende de `Domain` y proporciona implementaciones para interfaces definidas en las capas internas.
- Módulos como `Relays`, `Authentication`, `apiLdapLogin`, `apiLocalLogin` consumen servicios expuestos por `Application`, respetando las interfaces y contratos definidos en `Domain`.
- La capa Test valida el comportamiento y la interacción entre las diferentes capas, sin introducir dependencias que comprometan la arquitectura ni la independencia del dominio.

Este enfoque asegura que los cambios en las capas externas (infraestructura, interfaz de usuario o sistemas externos) no afecten la lógica central del dominio, garantizando la mantenibilidad, estabilidad y seguridad del sistema.

3.4 Diagramas de la Estructura General

Aquí se representa la arquitectura general mediante un diagrama de capas concéntricas. Las flechas indican la dirección de las dependencias permitidas:

```
┌──────────────────────────────┐
│        External Systems      │
│ DB | LDAP | IoT | APIs       │
└──────────────▲───────────────┘
               │
┌──────────────┴───────────────┐
│     Presentation Layer       │
│ Controllers | Endpoints      │
└──────────────▲───────────────┘
               │
┌──────────────┴───────────────┐
│   Infrastructure Layer       │
```

```
│ Repositories | Services    │
│ Device Integrations        │
└────────────▲───────────────┘
             │
┌────────────┴───────────────┐
│     Application Layer      │
│ Use Cases | App Services   │
└────────────▲───────────────┘
             │
┌────────────┴───────────────┐
│        Domain Layer        │
│ Entities | Interfaces      │
└────────────────────────────┘
```

La arquitectura del sistema se representa mediante un diagrama de capas concéntricas que refleja la separación de responsabilidades y el principio de inversión de dependencias. Las flechas indican la dirección permitida de las dependencias, que siempre se orientan hacia las capas internas.

En el nivel más externo se encuentran los **sistemas externos**, como bases de datos, servicios LDAP, dispositivos IoT y APIs externas, que interactúan con el sistema, pero no contienen lógica de negocio.

La **capa de presentación** actúa como punto de entrada al sistema e incluye los controladores y endpoints responsables de recibir las solicitudes externas y delegarlas a las capas internas correspondientes.

La **capa de infraestructura** contiene las implementaciones técnicas necesarias para el funcionamiento del sistema, como repositorios, servicios e integraciones con dispositivos o sistemas externos. Esta capa implementa las interfaces definidas en las capas internas, sin modificar su lógica.

La **capa de aplicación** coordina los casos de uso y los servicios de aplicación, orquestando el flujo de datos entre la capa de presentación, la infraestructura y el dominio.

En el núcleo del sistema se encuentra la **capa de dominio**, que contiene las entidades y las interfaces que definen las reglas de negocio. Esta capa es independiente de las demás y no depende de ninguna capa externa.

Esta estructura garantiza un bajo acoplamiento entre componentes, facilita el mantenimiento y permite modificar elementos externos sin afectar la lógica central del sistema.

3.5 Componentes y Módulos

- **Application**

Contiene los casos de uso del sistema y la lógica que coordina las operaciones entre los distintos servicios y módulos.

- **Domain**

Define las entidades del negocio (Paciente, Sala, Dispositivo, etc.), junto con sus interfaces, reglas y validaciones internas.

- **Infrastructure**

Implementaciones técnicas concretas: conexión a base de datos, integración con dispositivos externos, llamadas HTTP, acceso a servicios LDAP, entre otros.

- **Modules** -Módulos funcionales desacoplados

Módulos desacoplados con funcionalidades específicas:

  - `Relays`: Gestión de relés (encendido/apagado de señales físicas).
  - `LightBeacons`: Control de balizas luminosas por evento.
  - `ProxyDevices`: Administración de dispositivos puente o intermediarios.

- **Authentication**

Encapsula el proceso de autenticación: validación de credenciales, generación de tokens, y gestión de sesiones.

- **apiLdapLogin / apiLocalLogin**

Mecanismos alternativos de login:

- LDAP: Para entornos empresariales.
- Local: Para entornos sin directorio activo.

3.6 Descripción de las Funcionalidades

Cada módulo encapsula funcionalidades orientadas a resolver tareas específicas dentro del entorno hospitalario:

- **module.Relays**

Controla relés físicos o virtuales que permiten operar dispositivos conectados (luces, alarmas, sistemas de climatización).

- **module.LightBeacons**

Gestiona el encendido y comportamiento de balizas visuales que indican el estado de una sala (ocupada, libre, limpieza, emergencia).

- **module.ProxyDevices**

Administra dispositivos intermediarios que no se comunican directamente con el servidor central, pero que permiten la supervisión y el intercambio de información entre sistemas.

- **Authentication**

Valida la identidad del usuario mediante credenciales seguras, genera tokens de sesión, y gestiona la expiración y renovación de accesos.

- **Application**

Orquesta el flujo de trabajo entre capas, definiendo la lógica de negocio de alto nivel sin acoplarse a detalles técnicos.

3.7 Diseño de unidades de software

El sistema Epigram ADAS Health está compuesto por diferentes unidades de software que implementan las funcionalidades definidas en los requisitos del sistema.

Las unidades de software han sido definidas considerando los siguientes criterios:

- responsabilidad funcional única
- independencia lógica entre componentes
- capacidad de verificación mediante pruebas unitarias o de integración

Dado que el software **Epigram ADAS Health** está clasificado como **Clase A conforme a IEC 62304**, el nivel de detalle del diseño es proporcional al riesgo asociado.

<u>Unidad U-01 – Gestión de pacientes</u>

Responsabilidad

- creación de registros de paciente
- modificación de datos administrativos
- consulta de información del paciente
- eliminación de registros conforme a los permisos definidos

Entradas

- datos identificativos del paciente

- número de historia o identificador administrativo
- estado administrativo
- identificador del usuario que realiza la operación

Salidas

- confirmación de la operación realizada
- mensajes de error controlados
- registro en el sistema de auditoría

Reglas de negocio

- validación de campos obligatorios
- prevención de duplicidad de identificadores
- registro automático de fecha y hora de las operaciones

Unidad U-02 – Asignación de salas

Responsabilidad

- asignar un paciente a una sala
- modificar la asignación de sala
- liberar una sala cuando queda disponible
- consultar el estado de ocupación

Entradas

- identificador de paciente
- identificador de sala
- estado de disponibilidad

Salidas

- confirmación de asignación
- estado actualizado de la sala
- registro en el sistema de auditoría

Reglas de negocio

- no permitir la asignación a una sala ocupada
- un paciente solo puede tener una sala activa
- registro automático de cambios

Unidad U-03 – Gestión de usuarios y roles

Responsabilidad

- creación de usuarios
- modificación de permisos
- asignación de roles
- desactivación de cuentas

Roles definidos

- administrador
- enfermería
- usuario de consulta

Reglas de seguridad

- autenticación obligatoria
- control de acceso basado en roles (RBAC)
- registro de accesos

Unidad U-04 – Interfaz visual (Dashboard)

Responsabilidad

- visualización tipo panel o pizarra del estado de las salas
- visualización de pacientes asignados
- representación del estado administrativo de cada sala

Características

- actualización periódica o manual de la información
- representación visual mediante códigos de color
- no muestra información clínica del paciente

Unidad U-05 – Persistencia de datos

Responsabilidad

- almacenamiento estructurado de la información
- mantenimiento de la integridad referencial
- soporte para copias de seguridad

Controles

- uso de transacciones atómicas
- integridad de claves
- control de concurrencia

Unidad U-06 – Registro de auditoría

Responsabilidad

- registrar eventos relevantes del sistema, incluyendo:
  - creación o modificación de registros
  - asignación de salas
  - accesos al sistema

Propósito

- garantizar la trazabilidad de las acciones realizadas
- facilitar la investigación de incidencias
- cumplir los requisitos de auditoría y control

3.8 Interfaces entre unidades

Las unidades del sistema interactúan mediante interfaces internas que permiten el intercambio controlado de información.

Ejemplos de interacciones entre unidades incluyen:

Unidad origen	Unidad destino	Tipo de interacción
Dashboard	Gestión de pacientes	API interna
Dashboard	Asignación de salas	API interna
Gestión de pacientes	Persistencia de datos	Acceso a base de datos
Asignación de salas	Persistencia de datos	Acceso a base de datos
Gestión de pacientes / salas	Auditoría	Registro de eventos
Gestión de usuarios	Todas las unidades	Control de acceso

Las interfaces se implementan mediante servicios internos del sistema y mecanismos de persistencia de datos.

3.9 Consideraciones de seguridad

El diseño del software incorpora medidas para proteger el acceso al sistema y la integridad de la información:

- autenticación obligatoria de usuarios
- gestión segura de sesiones
- control de acceso basado en roles
- protección frente a accesos no autorizados y manipulación de datos

4. Instalación y despliegue del sistema

Este apartado describe el procedimiento para la instalación, configuración y despliegue del software **Epigram ADAS Health** en los distintos entornos de operación. El objetivo es garantizar que el sistema pueda ser instalado y desplegado de forma controlada, reproducible y conforme a los requisitos técnicos definidos.

El proceso incluye la preparación del entorno, la instalación del software, la configuración inicial del sistema y, cuando corresponde, el despliegue automatizado mediante herramientas de integración continua.

4.1 Instalación del software

La instalación básica del software **Epigram ADAS Health** se realiza a partir del repositorio del proyecto y requiere disponer previamente de las dependencias necesarias del sistema.

Clonar el repositorio

El primer paso consiste en obtener el código fuente del sistema desde el repositorio correspondiente:

```
git clone https://git.teralevel.io/adas/adas-core.git
```

Restaurar dependencias

Una vez descargado el repositorio, se deben restaurar las dependencias del proyecto:

```
dotnet restore
```

Compilar la solución

Posteriormente se compila el proyecto para verificar que todas las dependencias están correctamente instaladas:

```
dotnet build
```

Configuración de variables de entorno

Antes de ejecutar el sistema es necesario configurar los parámetros del entorno en los archivos de configuración, normalmente en:

```
appsettings.json
```

Entre los parámetros que deben configurarse se incluyen:

- cadenas de conexión a la base de datos
- credenciales de autenticación LDAP (si aplica)
- parámetros de configuración de los módulos funcionales
- puertos de escucha del servidor

Migración de la base de datos

Cuando el sistema utiliza Entity Framework Core, se deben aplicar las migraciones necesarias para crear o actualizar la estructura de la base de datos:

```
dotnet ef database update
```

Ejecución de la aplicación

Una vez completados los pasos anteriores, el sistema puede ejecutarse mediante el siguiente comando:

```
dotnet run --project adas-core
```

Este comando inicia el servicio de la API y permite el acceso a la aplicación desde el entorno configurado.

4.2 Configuración inicial del sistema

Tras la instalación del software, es necesario realizar una configuración inicial para adaptar el sistema al entorno de operación.

Configuración de base de datos

Se debe verificar la correcta configuración de la cadena de conexión en el archivo de configuración correspondiente y comprobar que las tablas necesarias han sido creadas correctamente.

Configuración de autenticación

El sistema permite distintos mecanismos de autenticación según el entorno de despliegue:

- autenticación local mediante usuario y contraseña
- autenticación mediante LDAP o Active Directory

Los parámetros de conexión al servidor LDAP deben configurarse en el archivo de configuración del sistema.

Registro de dispositivos

En instalaciones que utilizan dispositivos externos, estos deben registrarse en la base de datos del sistema para permitir su control y supervisión.

Entre los dispositivos que pueden registrarse se incluyen:

- relés
- balizas luminosas
- sensores
- dispositivos intermediarios (proxy devices)

Configuración de parámetros del sistema

Adicionalmente se pueden ajustar diferentes parámetros operativos del sistema, como:

- puertos de comunicación
- habilitación o deshabilitación de módulos
- parámetros de integración con sistemas externos.

4.3 Resolución de problemas durante la instalación

Durante la instalación pueden producirse incidencias relacionadas con dependencias, configuración o conectividad. A continuación se describen algunos problemas habituales y sus posibles soluciones.

Problema	Causa posible	Solución
No se encuentra la base de datos	Cadena de conexión incorrecta	Verificar la configuración en `appsettings.json`
Error al compilar	SDK de .NET no instalado o versión incorrecta	Instalar .NET 6.0 o superior
No se puede iniciar sesión	Configuración LDAP inválida	Revisar la conexión al servidor LDAP
La API no responde	Puertos bloqueados o configuración incorrecta	Verificar firewall y configuración de red

El análisis de errores adicionales puede realizarse mediante la revisión de los logs generados por el sistema.

4.4 Instalación por entornos

El sistema Epigram ADAS Health puede desplegarse en distintos entornos según la fase del ciclo de vida del software.

4.4.1 Entorno de desarrollo

El entorno de desarrollo se utiliza para la implementación de nuevas funcionalidades y la realización de pruebas técnicas durante el desarrollo.

Herramientas habituales en este entorno:

- Visual Studio 2022 o Visual Studio Code
- .NET SDK 6.0 o superior
- base de datos local (por ejemplo MongoDB)

Para facilitar el desarrollo iterativo puede utilizarse el comando:

```
dotnet watch run
```

que permite ejecutar la aplicación con recarga automática tras modificaciones en el código.

4.4.2 Entorno de pruebas

El entorno de pruebas se utiliza para verificar el funcionamiento del sistema antes de su liberación.

Características habituales de este entorno:

- infraestructura aislada del entorno de producción
- uso de datos simulados o anonimizados
- habilitación de sistemas de registro y monitorización

En este entorno se ejecutan actividades como:

- pruebas funcionales
- pruebas de integración
- pruebas de rendimiento.

4.4.3 Entorno de producción

El entorno de producción corresponde al sistema utilizado en operación real dentro de la organización sanitaria.

Para este entorno se recomienda:

- utilizar servidores dedicados o infraestructura cloud
- habilitar comunicaciones seguras mediante HTTPS
- configurar certificados digitales válidos
- aplicar reglas de firewall y control de acceso
- implementar mecanismos de monitorización del sistema
- establecer políticas de copias de seguridad automáticas.

4.5 Despliegue continuo y automatización

El software **Epigram ADAS Health** puede desplegarse mediante herramientas de integración y entrega continua (CI/CD), que permiten automatizar el proceso de compilación, prueba y despliegue del sistema.

Un pipeline de integración continua puede incluir las siguientes etapas:

1. envío de cambios al repositorio de código
2. ejecución automática de pruebas
3. compilación del software
4. generación de artefactos de despliegue
5. despliegue en el entorno correspondiente

Entre las herramientas que pueden utilizarse para este proceso se incluyen:

- GitHub Actions
- GitLab CI
- Azure DevOps

En entornos que utilizan contenedores, el despliegue puede realizarse mediante tecnologías como Docker o Docker Compose.

El uso de automatización en el despliegue reduce errores humanos y mejora la reproducibilidad del proceso de instalación.

4.6 Gestión de configuración del software

La gestión de configuración del software Epigram ADAS Health tiene como finalidad asegurar la identificación, control y trazabilidad de los elementos que componen el sistema a lo largo de su ciclo de vida.

Se consideran elementos de configuración, como mínimo:

- código fuente del software
- estructura de la base de datos
- documentación técnica del sistema
- scripts de instalación y despliegue

Estos elementos se gestionan mediante sistemas de control de versiones y procedimientos documentados de cambio, con el fin de garantizar la integridad del software y la coherencia entre las versiones liberadas.

Identificación de versiones

Las versiones del software **Epigram ADAS Health** se identifican mediante un esquema de versionado del tipo:

`MAJOR.MINOR.PATCH`

donde:

- **MAJOR** indica cambios estructurales importantes
- **MINOR** corresponde a nuevas funcionalidades o mejoras
- **PATCH** corresponde a correcciones de errores

Cada versión liberada se asocia a:

- los cambios implementados
- las pruebas ejecutadas
- la documentación técnica correspondiente.

5. Estructura del Código Fuente

5.1 Organización del repositorio

El repositorio del proyecto **Epigram ADAS Health** sigue una estructura modular y basada en Clean Architecture, lo cual permite escalar, mantener y probar el software de forma eficiente. A continuación, se describe la estructura general de carpetas:

bash

```
/adas-core/
│
├── Domain/                # Entidades y lógica de negocio central
├── Application/           # Casos de uso, interfaces y lógica de
aplicación
├── Infrastructure/        # Repositorios, servicios externos, DB
├── Presentation/          # Controladores, endpoints API, UI
├── Modules/               # Módulos funcionales independientes (ej:
Relays, LightBeacons)
├── Authentication/        # Gestión de usuarios, login, seguridad
├── Tests/                 # Pruebas unitarias y de integración
├── appsettings.json       # Configuración general de la aplicación
└── README.md              # Documentación básica del proyecto
```

5.2 Descripción de los módulos y paquetes

Carpeta / Módulo	Descripción
Domain/	Contiene las entidades del negocio (por ejemplo, `Paciente`, `Sala`, `Dispositivo`) y las interfaces para repositorios y servicios.
Application/	Aquí se desarrollan los casos de uso (por ejemplo, "Asignar paciente a sala", "Encender baliza") y la lógica de flujo de negocio.
Infrastructure/	Implementaciones técnicas concretas como acceso a base de datos, servicios REST externos, conexión con hardware (relés, sensores).
Presentation/	Controladores HTTP (endpoints), validaciones de entrada, respuestas API. Puede incluir UI si es una SPA o aplicación embebida.
Modules/	Cada módulo representa una funcionalidad independiente (por ejemplo: `Relays`, `ProxyDevices`, `LightBeacons`).
Authentication/	Encapsula todo lo relacionado con la autenticación: login local, LDAP, generación de tokens, manejo de permisos.
Tests/	Pruebas unitarias, de integración y mocks para cada capa del sistema.

5.3 Convenciones de codificación y estilo

- Lenguaje principal: **C#**
- Estilo: **.NET Style Guide**, incluyendo:
  - PascalCase para clases y métodos públicos
  - camelCase para variables locales y parámetros
  - Prefijo `I` para interfaces (ej. `IRepositorioSala`)
- Principios SOLID aplicados en todas las capas
- Separación clara entre lógica de negocio (Application/Domain) y lógica técnica (Infrastructure)

5.4 Guía de nombres de archivos y directorios

- Las carpetas deben describir su responsabilidad (`UseCases`, `Entities`, `Controllers`, `Repositories`).
- Los nombres de archivos deben reflejar claramente su contenido y propósito:
- `Paciente.cs` → entidad de dominio
- `AsignarSalaHandler.cs` → caso de uso / handler
- `SalaRepository.cs` → implementación de repositorio
- `SalaController.cs` → controlador de la API

6. Bases de Datos

6.1 Modelo de datos

El modelo de datos de **Epigram ADAS Health** está orientado a representar actividades operativas y de infraestructura hospitalaria. Incluye tablas para:

- Pacientes
- Salas
- Estados de salas
- Dispositivos (sensores, relés, balizas)
- Personal médico y de enfermería
- Registros de actividad e historial
- Configuración de módulos y eventos

6.1.1 Diagrama entidad-relación (ERD)

```
Paciente ──< RegistroActividad >── Sala
   │                  │
   └──< SignosVitales     EstadoSala
```

```
Dispositivo ──┐
              ├── ProxyDispositivo
              └── TipoDispositivo

Usuario ──< Permisos >── Rol
```

- Las líneas con >──< indican relaciones muchos a muchos.
- Las líneas simples ── indican relaciones uno a muchos.

6.1.2 Descripción de tablas y relaciones

Tabla	Descripción
`Pacientes`	Datos identificativos, estado y localización del paciente.
`Salas`	Identificación, tipo y estado de cada sala hospitalaria.
`EstadoSala`	Registro histórico de cambios en el estado de cada sala.
`Dispositivos`	Representa dispositivos conectados (balizas, relés, sensores, etc.).
`Usuarios`	Personal médico o técnico autorizado a acceder al sistema.
`Roles`	Define los niveles de acceso (admin, médico, mantenimiento, etc.).
`Permisos`	Define las acciones que puede realizar cada rol en cada módulo.
`Historial`	Guarda eventos importantes, errores, accesos y cambios.

6.2 Migraciones y actualizaciones

Las migraciones se ejecutan automáticamente con el inicio del api:

- Uso de la librería **MongoMigrations.Core**

6.3 Conexión y configuración

"ConnectionString": "mongodb://adas:123456@127.0.0.1:27017/adas?authSource=admin"

"DatabaseName": "adas"

7. Documentación de Código

7.1 Herramientas de documentación

El proyecto **Epigram ADAS Health** utiliza las siguientes herramientas para documentar el código fuente:

- **XML Documentation (C#)**: Comentarios en el código con el formato estándar de .NET (`/// <summary>...`)

- **DocFX**: Generador de documentación basado en esos comentarios, que exporta a HTML navegable.
- **Swagger/OpenAPI**: Para documentar automáticamente las APIs REST disponibles en el sistema.

7.2 Generación de documentación automática

La documentación técnica se genera automáticamente mediante integración en el pipeline de CI/CD:

1. Los desarrolladores comentan los métodos públicos usando el formato XML.
2. DocFX compila esa información y genera una interfaz web con navegación jerárquica.
3. Swagger UI se integra directamente en la API para que los endpoints sean visibles y testeables desde el navegador.

7.3 Mejores prácticas para comentarios y documentación

- Usar siempre `/// <summary>` en clases públicas, interfaces y métodos expuestos.
- Documentar los parámetros con `/// <param name="nombre">Descripción</param>`.
- Explicar los retornos con `/// <returns>Descripción</returns>`.
- Evitar comentarios redundantes u obvios.
- Actualizar los comentarios junto con el código: el código sin documentación válida es código incompleto.

8. Descripción de módulos y funciones

8.1 Funcionalidad de cada componente

- **Relays**: Controla dispositivos de encendido/apagado, como luces, alarmas o sistemas de climatización conectados.
- **LightBeacons**: Gestiona balizas visuales (luces de colores) que indican estado o alertas.
- **ProxyDevices**: Representa dispositivos externos conectados mediante intermediarios lógicos o físicos.
- **Authentication**: Módulo que gestiona el inicio de sesión, generación de tokens, validación de permisos y caducidad de sesión.

8.2 Relación con otros módulos

- Los módulos de dispositivos se comunican con el núcleo del sistema a través de la capa de **Application**.
- `Authentication` se conecta con **Usuarios** y **Roles** en la base de datos, y es invocado desde la capa de **Presentation**.
- Todos los módulos son orquestados por `Application`, que a su vez depende de entidades y reglas en `Domain`.

8.3 Parámetros y opciones de configuración

- Cada módulo puede ser configurado desde `appsettings.json`.
- Ejemplo para `Relays`:

```
json
"relay":{
    "driver" : "RelayDriver",
    "ip" : "RelayIpAddress",
    "port" : 5678,
    "relayNumber" : 2,
    "relayName" : "RelayName",
    "total" : 10,
    "refreshTime" : 30,
    "open" : false,
    "status" : 3,
    "username" : "RelayUsername",
    "password" : "RelayPassword",
    "mode" : 0,
    "rebootDelay" : null,
    "cache" : true,
    "manualRelayStatus" : null
}
```

9. Interfaz de Usuario

La interfaz de usuario (UI) de **Epigram ADAS Health** ha sido diseñada con un enfoque intuitivo, modular y centrado en las necesidades del personal clínico y técnico. Se adapta a dispositivos táctiles, estaciones de escritorio y paneles de visualización para facilitar la interacción con el sistema en tiempo real y con mínima curva de aprendizaje.

9.1 Capturas de Pantalla y Explicaciones

Pantalla principal – Tablero de control de salas

Muestra la lista de salas en formato de tarjetas o tabla, con indicadores de:

- Estado actual (ocupada, libre, limpieza, mantenimiento, etc.)
- Paciente asignado
- Personal responsable
- Dispositivos activos (relés, balizas)

Panel de detalles – Vista de paciente

Al hacer clic sobre una sala, se despliega un panel lateral con los datos administrativos del paciente, historial reciente, historial de cambios organizativos y acciones disponibles.

Menú de navegación lateral

Incluye accesos rápidos a:

- Salas
- Pacientes
- Dispositivos
- Historial
- Configuración (visible solo para usuarios con permisos)

Pantalla de autenticación

Diseñada para entrada rápida con usuario y contraseña local o mediante validación LDAP. Permite recuperación de sesión reciente en pantallas táctiles compartidas.

9.2 Navegación a Través de la Aplicación

La navegación en **Epigram ADAS Health** se basa en tres principios clave: simplicidad, visibilidad de la información y rapidez de acceso a acciones organizativas.

Elementos clave de la navegación:

- **Barra superior**: Muestra el título del módulo activo, el nombre del usuario y un icono de cierre de sesión.
- **Panel lateral izquierdo**: Navegación persistente entre secciones principales del sistema.
- **Área central dinámica**: Cambia según la sección activa y muestra información en tarjetas, tablas o visualizaciones interactivas.
- **Botones de acción rápida**: Dispuestos de forma visible en cada vista para ejecutar tareas frecuentes (ej. marcar como “en limpieza”, activar baliza, editar datos de paciente).
- **Filtros contextuales**: Permiten ordenar o segmentar salas por estado, prioridad, ubicación o tipo.

La aplicación está optimizada tanto para dispositivos de escritorio como para pantallas táctiles, utilizando botones grandes, colores contrastados y retroalimentación visual clara (luces, íconos, animaciones breves).

9.3 Consejos para el Uso Eficiente de la Interfaz

Para aprovechar al máximo la interfaz de **Epigram ADAS Health**, se recomienda:

Utilizar pantallas táctiles en puestos de control y enfermería, lo cual permite una interacción rápida y sin necesidad de periféricos adicionales.
Personalizar vistas según el rol del usuario:
- Médicos: visualización organizativa de ocupación.
- Enfermería: gestión de camas y asignaciones.
- Técnicos: supervisión operativa de dispositivos y alertas.

Activar notificaciones visuales mediante balizas, íconos de color y mensajes contextuales para alertas críticas o cambios de estado.
Usar filtros dinámicos y búsqueda rápida para encontrar pacientes, salas o eventos en pocos segundos.
Configurar accesos directos o favoritos para vistas frecuentes en entornos de trabajo con alta rotación o emergencias.
Desconectarse tras cada sesión en dispositivos compartidos para mantener la confidencialidad y seguridad de la información.

10. APIs y Servicios Externos

El sistema **Epigram ADAS Health** expone un conjunto de APIs RESTful que permiten la integración con otros sistemas hospitalarios para intercambio exclusivamente administrativo no clínico, dispositivos IoT, servicios de autenticación y aplicaciones de terceros. Estas APIs están diseñadas conforme a las buenas prácticas de desarrollo web, garantizando seguridad, trazabilidad y facilidad de uso.

10.1 Documentación de APIs Internas y Externas

Descripción general

Las APIs de **Epigram ADAS Health** siguen el estándar **REST**, utilizan **JSON** como formato de intercambio de datos, y están documentadas de forma interactiva mediante **Swagger (OpenAPI)**, lo que permite su exploración y prueba desde el navegador.

Documentación disponible

- **Swagger UI**: acceso directo a la documentación navegable de endpoints.

- **OpenAPI JSON/YAML**: disponible para exportación e integración con herramientas externas.

Endpoints principales

Método	Endpoint	Función
Displays		
GET	/displays	Lista de displays disponibles
Patient		
GET	/persons	Lista de pacientes
POST	/persons	Crea un nuevo paciente
PUT	/persons	Actualiza la información de un paciente
PointOfCare		
GET	/point-of-cares	Lista los puntos de atención médica configurados
POST	/point-of-cares	Crea un nuevo punto de atención
PUT	/point-of-cares	Actualiza un punto de atención existente
User		
POST	/users/login	Autenticación del usuario (inicio de sesión)
GET	/users/login	Verifica estado de autenticación o sesión del usuario
GET	/users/token	Obtiene un token de autenticación de usuario
SendAlert		
GET	/send-alerts/rabbit	Verifica el estado de envío de alertas mediante RabbitMQ
GET	/send-alerts/performance	Consulta rendimiento del sistema de alertas
GET	/send-alerts/api-clients	Lista los clientes API suscritos a alertas
POST	/send-alerts/web-subscribe	Suscribe un cliente web a recibir alertas
POST	/send-alerts/web-unsubscribe	Cancela la suscripción de un cliente web

POST	/send-alerts/broadcast/{sub}	Envía una alerta a todos los suscriptores del canal {sub}

Métodos HTTP soportados

- **GET**: Lectura y consultas de datos
- **POST**: Creación de recursos (pacientes, salas, eventos)
- **PUT**: Actualización parcial o total de un recurso
- **DELETE**: Eliminación lógica (aplicado solo en casos controlados, como desactivar dispositivos)

Parámetros de entrada y salida

Los datos se intercambian en formato **JSON** y son validados automáticamente mediante atributos (`DataAnnotations`) en el backend .NET.

Ejemplo de entrada JSON para crear una sala:

```
{
  "nombre": "Sala 1",
  "estado": "ocupada",
  "pacientes": [
    {
      "id": 101,
      "nombre": "Juan Pérez"
    }
  ]
}
```

Autenticación y autorización

La API está protegida mediante autenticación **JWT (Bearer Token)**.

- Los tokens se generan tras un login exitoso (local o LDAP).
- Cada token incluye **claims** que identifican el rol del usuario.
- El acceso a cada endpoint está condicionado por el rol (enfermero, médico, técnico, admin).
- La expiración del token es configurable para adaptarse al flujo de trabajo clínico.

10.2 Ejemplos de Código y Uso

Este apartado remite al **Anexo 1a – Ejemplos de Código y Uso**, donde se detallan diversos fragmentos de código, estructuras de implementación y escenarios de aplicación del software. El anexo complementa la documentación técnica principal, proporcionando referencias prácticas que ilustran el comportamiento esperado del sistema en diferentes contextos de uso.

10.3 Directrices para la Integración con Otros Servicios

Integración con LDAP

- Permite validar las credenciales del usuario contra el **directorio activo** del hospital.
- El sistema identifica automáticamente el rol del usuario según su pertenencia a grupos de AD.
- El módulo `apiLdapLogin` debe configurarse con:
  - Dominio LDAP
  - Unidad organizativa base (OU)
  - Usuario de lectura con permisos mínimos

Integración con dispositivos IoT

- **Epigram ADAS Health** puede integrarse con dispositivos mediante **protocolo MQTT o HTTP**.
- Los dispositivos deben ser autenticados por IP o token seguro.
- Las acciones comunes incluyen:
  - Activación de relés o balizas (ej. vía POST `/relays/activar`)
  - Lectura periódica de sensores (vía polling o suscripción)

Ejemplos de Integración con Otros Servicios

Servicio Externo	Tipo de Integración	Ejemplo Funcional
Active Directory	Autenticación vía LDAP	Inicio de sesión del personal sanitario
Sensores IoT	Comunicación MQTT	Lectura automática de presencia/silencio
HL7/FHIR	Intercambio administrativo no clínico	Conexión con otros HIS o EMR (personalizado)

Limitaciones y Recomendaciones

- **Límites de petición**: Se recomienda no exceder 1000 peticiones/minuto por usuario para evitar sobrecarga.

- **Tiempos de respuesta esperados**: 200–300 ms en consultas estándar.
- **Reintentos automáticos**: Aplicar políticas de reintento con backoff exponencial en servicios externos.
- **Seguridad**: No exponer APIs sin HTTPS. Asegurar logs y monitoreo activo de accesos a través de herramientas como Serilog o Prometheus + Grafana.

11. Seguridad

11.1 Medidas de seguridad implementadas

- **Autenticación basada en tokens (JWT)**: Permite sesiones seguras con expiración y validación del usuario.
- **Roles y permisos**: Cada usuario accede solo a la funcionalidad correspondiente a su rol (médico, enfermero, técnico, admin).
- **HTTPS obligatorio**: Toda comunicación entre clientes y servidores debe estar cifrada mediante certificados TLS válidos.
- **Auditoría de eventos**: Registro automático de accesos, cambios críticos y operaciones sensibles.

11.2 Guía de buenas prácticas de seguridad. Recomendaciones para mantener la seguridad

- No compartir credenciales entre usuarios.
- Cambiar contraseñas periódicamente (recomendado cada 90 días).
- Evitar el acceso a la aplicación desde redes no seguras.
- Bloquear cuentas tras múltiples intentos fallidos de login.
- No exponer endpoints sin autenticación (usar middlewares de protección).

11.3 Pautas para la gestión de usuarios y permisos. Procedimientos en caso de vulnerabilidades detectadas

11.3.1 Gestión de claves y certificados

- Las claves privadas deben almacenarse en un sistema seguro (como Azure Key Vault, AWS Secrets Manager o archivos fuera del control del código fuente).
- Los certificados TLS deben renovarse automáticamente (por ejemplo, con Let's Encrypt).
- Se recomienda rotación de claves cada 6 meses.

11.3.2 Control de acceso y roles

- **Admin**: Acceso total a configuración, usuarios, módulos y logs.
- **Médico**: Puede visualizar, registrar y actualizar información organizativa relacionada con la opcupación de salas.
- **Enfermero**: Acceso a control de pacientes y salas.
- **Técnico/Mantenimiento**: Acceso limitado a visualización del estado y equipos.

11.3.3 Manejo de datos sensibles

- Datos personales administrativos se almacenan cifrados.
- El acceso a información confidencial requiere doble validación en entornos sensibles.
- Se cumple con la normativa europea de protección de datos (**GDPR**).

11.3.4 Procedimientos en caso de vulnerabilidades

- Notificar al equipo de seguridad (correo electrónico Epigram).
- Activar el protocolo de contención (bloqueo de usuario, desconexión de red).
- Documentar el incidente.
- Aplicar parche o mitigación en el menor tiempo posible.
- Informar a los responsables y registrar el seguimiento.

12. Ciclo de vida y desarrollo del software

12.1 Desarrollo del software

Esta sección describe el proceso seguido para el desarrollo, mantenimiento, verificación, validación y liberación del software **Epigram ADAS Health**.

El proceso de desarrollo se aplica a:

- desarrollo inicial del software
- modificaciones funcionales
- corrección de errores
- actualizaciones y parches
- liberación de nuevas versiones

El objetivo es garantizar la trazabilidad, calidad y control del software conforme a buenas prácticas de desarrollo de software médico y a los requisitos regulatorios aplicables.

12.2 Modelo de desarrollo

El desarrollo de **Epigram ADAS Health** sigue un modelo **incremental con prácticas ágiles**, permitiendo la evolución progresiva del sistema mediante versiones controladas.

Este modelo permite:

- incorporación progresiva de funcionalidades
- mejora continua del sistema
- validación frecuente de cambios
- reducción del impacto de modificaciones

El desarrollo se realiza siguiendo una arquitectura modular basada en **Clean Architecture**, que facilita la mantenibilidad, la escalabilidad y la independencia entre componentes.

12.3 Roles y responsabilidades

Durante el ciclo de vida del software participan los siguientes roles:

Responsable de desarrollo

- analizar requisitos técnicos y funcionales
- diseñar la solución técnica
- implementar el software
- documentar el desarrollo realizado
- garantizar el cumplimiento de la arquitectura y convenciones de codificación

Responsable de verificación / pruebas

- definir y ejecutar casos de prueba
- documentar resultados de pruebas
- verificar que el software cumple los requisitos definidos
- registrar incidencias detectadas

Responsable de aprobación

- revisar resultados de verificación y validación
- confirmar cumplimiento de criterios de aceptación
- autorizar la liberación de versiones del software

Responsable de calidad

- supervisar el cumplimiento del proceso de desarrollo

- garantizar la trazabilidad entre requisitos, desarrollo y pruebas
- asegurar el cumplimiento regulatorio

12.4 Gestión de requisitos

Los requisitos del software se documentan en el documento **DOC 05 – Especificación de Requisitos de Software (SRS)**.

Cada requisito dispone de:

- identificador único
- descripción
- tipo (funcional, no funcional, seguridad o compatibilidad)
- riesgos asociados
- pruebas asociadas

La trazabilidad entre requisitos, riesgos y pruebas se mantiene mediante:

- DOC 13 – Matriz de trazabilidad
- Registro de riesgos
- Plan de pruebas

12.5 Gestión del desarrollo

El desarrollo del software se realiza siguiendo las siguientes fases:

Análisis

Identificación del requisito o cambio solicitado.

Evaluación del impacto en:

- arquitectura
- módulos
- base de datos
- seguridad
- integraciones externas

Diseño

Definición de la solución técnica, incluyendo:

- componentes afectados

- interfaces
- estructuras de datos
- lógica de negocio

Cuando es necesario, se actualiza la documentación técnica.

Implementación

La implementación se realiza siguiendo:

- arquitectura definida
- principios SOLID
- convenciones de codificación del proyecto

El control de versiones se gestiona mediante repositorio Git.

12.6 Gestión de versiones

Cada versión del software **Epigram ADAS Health** se identifica mediante **versionado semántico**:

```
MAJOR.MINOR.PATCH
```

Ejemplo:

```
1.0.0
1.1.0
1.1.1
```

Cada versión liberada incluye:

- número de versión
- fecha de liberación
- descripción de cambios
- pruebas ejecutadas
- aprobación de liberación

12.7 Gestión de cambios

Todos los cambios en el software se registran en el documento:

Registro de Desarrollo y Cambios

Cada registro incluye:

- identificador del cambio
- descripción
- tipo de cambio (corrección, mejora, nueva funcionalidad)
- versión afectada
- requisitos asociados
- pruebas realizadas
- responsable del cambio
- aprobación

Este registro garantiza la trazabilidad completa del ciclo de vida del software.

12.8 Liberación del software

Una versión del software puede ser liberada únicamente cuando se cumplen los siguientes criterios:

- los requisitos asociados han sido implementados
- las pruebas han sido ejecutadas con resultado satisfactorio
- no existen defectos críticos abiertos
- la documentación técnica ha sido actualizada
- la liberación ha sido aprobada formalmente

Para cada versión liberada se generan **notas de versión (Release Notes)** que incluyen:

- versión
- fecha
- cambios implementados
- correcciones
- pruebas realizadas

13. Pruebas

El proceso de aseguramiento de calidad del software **Epigram ADAS Health** incluye actividades de verificación y pruebas destinadas a confirmar que el sistema cumple los requisitos definidos y funciona de forma segura, estable y conforme a su uso previsto.

Las actividades de prueba se realizan durante el desarrollo, antes de la liberación de nuevas versiones y cuando se aplican cambios derivados de actividades de mantenimiento.

13.1 Objetivo de la verificación

El proceso de verificación tiene como objetivo comprobar que el software **Epigram ADAS Health** cumple los requisitos funcionales y no funcionales definidos en la **Especificación de Requisitos de Software (TF_05)**.

La verificación permite asegurar que el sistema implementa correctamente las funcionalidades especificadas y que su comportamiento es coherente con el diseño definido.

13.2 Alcance de la verificación

Las actividades de verificación se aplican a todos los requisitos definidos para el software Epigram ADAS Health, incluyendo:

- requisitos funcionales
- requisitos no funcionales
- requisitos de seguridad
- requisitos de compatibilidad y entorno

Cada requisito debe estar asociado a al menos una actividad de verificación que permita comprobar su correcta implementación.

13.3 Métodos de verificación

La verificación del software puede realizarse mediante diferentes métodos, según la naturaleza del requisito evaluado.

Los métodos utilizados incluyen:

Revisión documental: Revisión de los requisitos, arquitectura y documentación técnica para verificar su coherencia y completitud.

Pruebas funcionales: Ejecución de pruebas destinadas a comprobar el correcto funcionamiento de las funcionalidades del sistema.

Pruebas de integración: Verificación del correcto funcionamiento de los diferentes módulos del sistema cuando interactúan entre sí.

Pruebas de usuario: Validación del comportamiento del sistema desde la perspectiva del usuario en escenarios representativos de uso.

13.4 Tipos de Pruebas Realizadas

Pruebas unitarias

- Validan el correcto funcionamiento de funciones y métodos individuales en aislamiento.
- Aplicadas principalmente a la lógica de negocio contenida en las capas `Domain` y `Application`.

Pruebas de integración

- Verifican la comunicación y correcta interacción entre distintos módulos del sistema:
  - API ↔ Base de datos
  - API ↔ Módulos funcionales (`Relays`, `LightBeacons`, etc.)
  - API ↔ Servicios externos (LDAP, dispositivos IoT)

Pruebas de sistema

- Evalúan el comportamiento del sistema completo mediante flujos de uso reales.
- Validan escenarios operativos como:
  - Admisión de pacientes
  - Cambio de estado de salas
  - Activación de dispositivos y alertas

Pruebas de rendimiento

- Miden el tiempo de respuesta, consumo de CPU y RAM bajo carga simulada.
- Se realizan en entorno de staging antes de pasar a producción.

Pruebas de seguridad

- Evalúan la resistencia del sistema frente a ataques conocidos:
  - Inyección SQL (SQLi)
  - Cross-Site Scripting (XSS)
  - Cross-Site Request Forgery (CSRF)

13.5 Criterios de aceptación

Para considerar que la verificación del software es satisfactoria deben cumplirse los siguientes criterios:

- todos los requisitos definidos deben haber sido verificados
- no deben existir errores críticos abiertos antes de la liberación de una versión del software
- las pruebas deben haber sido documentadas y revisadas

13.6 Plan de pruebas del software

Las pruebas detalladas del software **Epigram ADAS Health** se definen en el documento **Plan de Pruebas y Casos de Prueba (DOC 08)**.

Este documento describe:

- la estrategia de pruebas
- los casos de prueba definidos
- los requisitos asociados
- los riesgos asociados
- los resultados esperados

Cada caso de prueba se encuentra trazado con los requisitos definidos en **TF_05** y con los riesgos documentados en el **Registro Maestro de Riesgos (REG-61-01)** mediante la **Matriz de Trazabilidad (DOC 13)**.

13.7 Ejecución y registro de pruebas

La ejecución de las pruebas se documenta en el **Registro de Pruebas del software**, donde se registra información como:

- identificador de la prueba
- versión del software evaluada
- entorno de prueba
- fecha de ejecución
- técnico responsable
- resultado obtenido

Las evidencias de las pruebas realizadas se conservan en el **Registro de Evidencias TC**.

13.8 Responsables de verificación

Las actividades de verificación se realizan bajo las siguientes responsabilidades:

Responsable técnico

- ejecutar las pruebas definidas
- documentar los resultados obtenidos

Responsable de calidad

- revisar los resultados de verificación

- confirmar que las pruebas han sido ejecutadas correctamente
- autorizar el cierre de las actividades de verificación

14. Mantenimiento y actualizaciones

14.1 Objetivo y alcance del mantenimiento

El mantenimiento del software **Epigram ADAS Health** tiene como objetivo garantizar que el sistema continúe funcionando de forma segura, estable y conforme a su uso previsto después de su liberación.

Las actividades de mantenimiento incluyen la gestión de incidencias, correcciones de errores, actualizaciones del sistema y mejoras evolutivas del software.

Este proceso se realiza de forma controlada y documentada conforme a:

- el proceso de desarrollo definido para el software
- el sistema de gestión de calidad de la organización
- las buenas prácticas aplicables al ciclo de vida del software médico.

Las actividades de mantenimiento aplican a todas las versiones liberadas del software **Epigram ADAS Health** y pueden incluir correcciones, adaptaciones al entorno técnico, mejoras funcionales o acciones preventivas destinadas a mejorar la estabilidad y seguridad del sistema.

14.2 Tipos de mantenimiento

Las actividades de mantenimiento del software **Epigram ADAS Health** se clasifican en los siguientes tipos:

Mantenimiento correctivo: Corrección de errores o defectos detectados tras la liberación del software.

Mantenimiento adaptativo: Adaptación del software a cambios en su entorno operativo, como actualizaciones del sistema operativo, dependencias o infraestructura.

Mantenimiento evolutivo: Incorporación de mejoras o nuevas funcionalidades en el software.

Mantenimiento preventivo: Cambios destinados a mejorar la estabilidad, seguridad o mantenibilidad del sistema, reduciendo la probabilidad de fallos futuros.

14.3 Gestión de incidencias y errores

Los errores o incidencias del software pueden detectarse durante el uso del sistema, a través de actividades de soporte técnico, monitorización de logs o durante revisiones internas.

Cada incidencia se registra y analiza para determinar su causa raíz y evaluar su impacto en el funcionamiento del sistema.

El registro de incidencias incluye, al menos:

- descripción del error
- versión del software afectada
- módulo o componente afectado
- fecha de detección
- origen de la incidencia
- evaluación preliminar del impacto

Cuando corresponde, se evalúa el impacto del error sobre la seguridad del software o sobre riesgos previamente identificados.

Las incidencias y las acciones correctivas aplicadas se documentan en el **Registro de desarrollo y cambios del software**, garantizando la trazabilidad completa del proceso de mantenimiento.

14.4 Implementación de cambios

Cuando una incidencia o necesidad de mejora requiere modificar el software **Epigram ADAS Health**, el cambio se gestiona de forma controlada conforme al proceso de desarrollo del software.

Cada cambio implementado debe incluir, como mínimo:

- registro del cambio en el **Registro de desarrollo y cambios del software**
- referencia a la incidencia o necesidad que origina el cambio
- identificación de los requisitos afectados, cuando corresponda
- evaluación del impacto técnico del cambio
- evaluación del impacto en riesgos, cuando aplique
- implementación del cambio en el código fuente
- actualización de la documentación técnica, si es necesario

El objetivo de este proceso es garantizar que todos los cambios realizados en el software sean trazables desde su origen hasta su liberación.

14.5 Verificación y pruebas tras mantenimiento

Todo cambio implementado como resultado de actividades de mantenimiento debe verificarse antes de su liberación.

El objetivo de esta verificación es confirmar que:

- el error ha sido corregido correctamente
- el sistema continúa cumpliendo sus requisitos funcionales
- no se han introducido nuevos errores o efectos no deseados

Las actividades de verificación pueden incluir:

- pruebas unitarias
- pruebas funcionales
- pruebas de integración
- pruebas de regresión

Las pruebas realizadas se documentan en el **registro de pruebas del software**, incluyendo:

- versión del software evaluada
- pruebas ejecutadas
- fecha de ejecución
- responsable de la prueba
- resultado obtenido
- evidencias asociadas cuando corresponda

14.6 Liberación de versiones tras mantenimiento

Cuando una actividad de mantenimiento implica modificaciones en el software, se genera una nueva versión del sistema.

Las versiones del software **Epigram ADAS Health** se identifican mediante un sistema de versionado del tipo:

MAJOR.MINOR.PATCH

donde:

- **MAJOR** indica cambios importantes o modificaciones estructurales
- **MINOR** corresponde a nuevas funcionalidades o mejoras
- **PATCH** corresponde a correcciones de errores o ajustes menores

Antes de liberar una nueva versión deben completarse las siguientes actividades:

- finalización de las pruebas de verificación
- actualización de la documentación técnica cuando sea necesario
- registro del cambio en el **Registro de desarrollo y cambios**
- documentación de la versión mediante **nota de versión (release notes)**

Cada versión liberada se conserva para garantizar la trazabilidad completa del software.

14.7 Procedimientos para el mantenimiento rutinario

Un mantenimiento regular es fundamental para garantizar la estabilidad, rendimiento y seguridad del sistema **Epigram ADAS Health**. Se recomienda seguir los siguientes procedimientos:

- **Revisión semanal de logs de errores**

  Revisar los registros generados por el sistema (`logs/`, visor de eventos, etc.) para identificar errores recurrentes o inesperados.

- **Verificación diaria del funcionamiento de los dispositivos conectados**
  Validar que relés, balizas, sensores y otros dispositivos estén activos y sincronizados correctamente.

- **Comprobación de backups automáticos**

  Confirmar que las copias de seguridad (base de datos y configuración) se están generando según el cronograma y son recuperables.

- **Actualización mensual del sistema operativo y dependencias críticas**

  Aplicar actualizaciones de seguridad del sistema operativo y bibliotecas (por ejemplo, .NET, drivers, frameworks) en un entorno controlado antes de pasar a producción.

- **Supervisión de uso de recursos**

  Monitorizar el uso de CPU, memoria y espacio en disco del servidor de **Epigram ADAS Health** para evitar cuellos de botella o caídas inesperadas.

14.8 Aplicación de actualizaciones y parches

Para garantizar la continuidad del servicio y minimizar los riesgos, se recomienda seguir los siguientes pasos:

1. **Verificar compatibilidad de versión.**

- Consultar el changelog y la documentación del proveedor para identificar cambios de configuración o estructura de datos.

2. **Preparar un entorno de pruebas (staging)**
   - Replicar el entorno de producción con datos anonimizados o simulados para realizar la validación previa.

3. **Ejecutar migraciones de base de datos (si aplica)**
   - Aplicar scripts de migración con supervisión técnica.
   - Registrar la ejecución para trazabilidad.

4. **Validar el sistema mediante pruebas automatizadas y manuales**
   - Verificar que todos los módulos funcionen correctamente.
   - Confirmar que no se han roto integraciones o funcionalidades existentes.

5. **Desplegar en producción durante una ventana de mantenimiento**
   - Notificar con antelación a los usuarios.
   - Realizar el despliegue fuera del horario pico.
   - Monitorear el sistema durante las primeras horas posteriores.

14.9 Migración de datos entre versiones

Cuando una nueva versión modifica la estructura de datos, se deben seguir los siguientes pasos para una migración segura:

- **Exportar backup completo**
  Realizar una copia de seguridad de la base de datos, configuración y archivos críticos antes de iniciar la migración.
- **Usar scripts de migración versionados**
  Ejemplo:

```
pgsql

V1__init.sql
V2__add_estado_column.sql
V3__migracion_pacientes.sql
```

- **Aplicar las migraciones en orden y con logs de auditoría**
  Registrar el tiempo, usuario y resultado de cada script ejecutado.
- **Validar integridad de los datos migrados**
  Comparar conteo de registros, relaciones clave y campos obligatorios antes y después de la migración.

- **Plan de reversión (rollback)**
  Tener disponibles scripts o snapshots para volver a la versión anterior si la migración falla.

14.10 Buenas prácticas para el mantenimiento

- **Automatizar tareas recurrentes** con scripts o herramientas CI/CD.
- **Mantener documentación técnica actualizada** con cada cambio aplicado.
- **Registrar en un changelog interno** todas las intervenciones realizadas (actualizaciones, incidentes, reinicios).
- **Capacitar al personal técnico regularmente** sobre los procedimientos de actualización y recuperación.
- **Evitar modificaciones directas en producción** sin revisión previa en entorno de pruebas.

14.11 Puntos de contacto para soporte técnico

- **Correo electrónico de soporte:** soporte@smacs.com
- **Teléfono de emergencia:** 963 531 206
- **Portal web de soporte:** https://soporte.smacs.es/
- **Centro de actualizaciones:** consultas soporte@smacs.com

14.12 Procedimiento de actualizaciones y parches

1. **Recepción del aviso de nueva versión o parche**
   a. A través del portal de soporte o boletín de seguridad.
2. **Evaluación del impacto**
   a. Determinar si el parche es crítico, funcional o de seguridad.
3. **Planificación del despliegue**
   a. Asignar fecha, recursos técnicos y comunicación interna.
4. **Descarga y validación del paquete**
   a. Verificar integridad mediante hash/checksum.
5. **Aplicación del parche**
   a. Mediante script, instalador o actualización CI/CD.
6. **Pruebas post-implementación**
   a. Verificar logs, disponibilidad del sistema y funcionalidad clave.
7. **Cierre del proceso**
   a. Registrar el parche aplicado y actualizar el estado del sistema.

15. Resolución de problemas

15.1 Gestión de incidencias

Cuando se detecta un problema en el funcionamiento del software **Epigram ADAS Health**, la incidencia debe registrarse para su análisis y resolución.

Las incidencias pueden ser detectadas por:

- usuarios del sistema
- personal técnico
- resultados de pruebas o monitorización

Cada incidencia registrada debe incluir, al menos:

- descripción del problema
- módulo o componente afectado
- fecha de detección
- persona que reporta la incidencia

Análisis y resolución

El equipo técnico analiza la causa del problema y determina la acción correctiva necesaria.

La corrección puede consistir en:

- modificación del software
- ajuste de configuración
- actualización del sistema

Cuando la corrección implica cambios en el software, estos se gestionan conforme al proceso de desarrollo y control de cambios descrito en el apartado **13 Ciclo de vida y desarrollo del software**.

Verificación

Una vez aplicada la corrección, se verifica que el problema ha sido resuelto y que el sistema continúa funcionando correctamente.

Registro

Las incidencias y las acciones correctivas aplicadas se documentan en el **Registro de Desarrollo y Cambios del software**, con el fin de mantener la trazabilidad del ciclo de vida del sistema.

15.2 Diagnóstico y solución de errores comunes

Problema	Causa probable	Solución sugerida
No se inicia la API	Cadena de conexión incorrecta	Revisar `appsettings.json`
Usuario no puede iniciar sesión	Token inválido o expirado	Solicitar nuevo login

Datos no se actualizan en pantalla	Error en WebSocket o conexión backend	Reiniciar servicio o revisar logs
Dispositivo no responde	Red desconectada o IP incorrecta	Verificar configuración de red

15.3 Preguntas frecuentes y sus respuestas

- **Qué hago si pierdo mi contraseña?**
  Contactar con el administrador del sistema para restablecerla.
- **¿Puedo cambiar el idioma del sistema?**
  Actualmente disponible solo en español. Versión multilingüe en desarrollo.
- **¿Dónde encuentro los logs?**
  Carpeta `logs/` dentro del directorio de la aplicación.

15.4 Canales de soporte técnico

- Correo electrónico de soporte
- Manual de usuario disponible
- Centro de ayuda en línea

16. Glosario

- **API**: Interfaz de programación de aplicaciones. Conjunto de definiciones y protocolos que permiten la comunicación entre diferentes sistemas o componentes.
- **Backoff exponencial**: Estrategia para repetir operaciones fallidas aumentando progresivamente el tiempo de espera entre intentos.
- **Baliza (Beacon)**: Dispositivo luminoso o sonoro utilizado para indicar estados o alertas dentro del sistema Epigram ADAS Health.
- **CI/CD**: Integración y entrega continuas. Prácticas de desarrollo que permiten automatizar pruebas, integración y despliegue de software. Prácticas de desarrollo que permiten automatizar pruebas, integración y despliegue de software.
- **CRUD**: Create, Read, Update, Delete. Operaciones básicas para el manejo de datos en bases de datos y sistemas informáticos.
- **Docker**: Plataforma que permite crear, desplegar y ejecutar aplicaciones en contenedores, aislando entornos de ejecución.
- **EF Core**: Entity Framework Core. ORM (Object Relational Mapper) para .NET que permite el acceso a bases de datos de manera orientada a objetos.
- **JWT**: Token web JSON (usado para autenticación).
- **LDAP**: Lightweight Directory Access Protocol. Protocolo utilizado para acceder y mantener servicios de directorio distribuidos.

- **Microservicio**: Estilo de arquitectura de software que estructura una aplicación como un conjunto de servicios pequeños e independientes.
- **Pipeline**: Flujo automatizado de tareas en el proceso de desarrollo (compilación, pruebas, despliegue, etc.).
- **Proxy Device**: Dispositivo intermedio que facilita la comunicación entre el sistema central y otros dispositivos no integrables directamente.
- **Relé (Relay)**: Dispositivo físico o virtual que permite el control de circuitos eléctricos a través de señales digitales.
- **Role-based Access Control (RBAC)**: Control de acceso basado en roles. Cada usuario tiene permisos según su función.
- **Swagger / OpenAPI**: Herramientas para documentar y probar APIs REST. Permiten una exploración visual de los endpoints disponibles.
- **TDE**: Transparent Data Encryption. Técnica de cifrado de bases de datos en reposo para proteger los datos sensibles.
- **Token**: Cadena segura que representa la identidad o autorización de un usuario durante una sesión en el sistema.

17. Bibliografía

- OWASP Secure Coding Guidelines
- MDR 2017/745 (Reglamento Europeo de Productos Sanitarios)

18. Licencias y créditos de componentes de terceros

- Dependencias bajo licencia MIT, Apache 2.0 o GNU
- Frameworks utilizados: .NET 6, Swagger, xUnit

19. Comunidad de foros y soporte

- Canal de soporte interno

20. Anexos

- Anexo 1a – Ejemplos de Código y Uso

21. Historial de cambios

Versión	Fecha	Descripción del cambio
		Edición inicial

Anexo 1a – Ejemplos de Código y Uso

A continuación, se describe el funcionamiento y uso de dos de los endpoints pertenecientes a la API, /point-of-cares y /displays/compact. Esta documentación está diseñada para proporcionar una comprensión clara de su propósito, cómo interactuar con ellos y qué esperar de sus respuestas.

Ejemplo 1. Obtener Listado Compacto de Displays

Este endpoint se utiliza para recuperar una lista con todos los displays registrados en el sistema. La petición no devuelve toda la información de cada display, sino únicamente aquella necesaria para visualizar el listado, minimizando así el tiempo de ejecución y el consumo de recursos.

- **Método HTTP**: GET
- **Descripción**: Recupera una lista abreviada de todos los displays.
- **URL**: `/displays/compact`
- **Parámetros de la solicitud**: Al ser una petición GET simple, no requiere de ningún parámetro en la URL ni en el cuerpo de la solicitud.
- **Tipo de contenido de la solicitud**: No aplica.
- **Ejemplo de Solicitud (Request)**:

```
GET /displays/compact
```

- **Ejemplo de Respuesta (Response):** Si la solicitud se realiza correctamente, se recibirá un código de estado **`200 OK`** y un array JSON con el listado de displays. Cada objeto en el array contendrá el `id`, `name` y `type` del display.


```
[
  {
    "id": "65c114566d5d1ac02b4f2478",
    "name": "VdH UCI D1",
    "type": "DisplayNurse"
  },
  {
    "id": "66191fc591b99fb5ac42e715",
    "name": "Trueta UCI A",
    "type": "DisplayNurse"
  },
  {
    "id": "661ff0deb38e54a8b21011fd",
    "name": "VdH UCI D2",
    "type": "DisplayNurse"
  },
]
```


- **Posibles Códigos de Estado:**
  - **`200 OK`:** La solicitud fue exitosa y devolvió el listado de displays.
  - **`500 Internal Server Error`:** Ocurrió un error inesperado en el servidor.

Ejemplo 2. Creación de un nuevo punto de cuidado

Este endpoint permite la creación de nuevos puntos de cuidado en el sistema, los cuales son esenciales para organizar y gestionar la información del paciente. Permite configurar detalles como la ubicación física (habitación y cama), la unidad a la que pertenece y las configuraciones específicas de dispositivos asociados, como cámaras y balizas.

- **Método HTTP**: POST
- **Descripción**: Crea un nuevo punto de cuidado con los detalles proporcionados.
- **URL**: `/point-of-cares`
- **Tipo de Contenido de la Solicitud**: `application/json`.
- **Ejemplo de Solicitud (Request):** El siguiente bloque de código JSON representa el cuerpo de la solicitud para crear un punto de cuidado. Este ejemplo crearía un punto de cuidado en la "Room 301", "Bed A" de la unidad con el `unitId` especificado, e incluiría la configuración para una baliza.

```
POST /point-of-cares

Content-Type: application/json
{
  "room": "Room 301",
  "bed": "Bed A",
  "hall": ",
  "unitId": "65c0964a295c815955366c23",
  "configuration": {
    "beacons": [
      {
        "type": "BeaconType",
        "options": {
          "url": "http://beacon.example.com/data",
          "port": 8888,
          "password": "secure_password",
          "emulate": true
        }
      }
  ],
```

```
}
```

- **Ejemplo de Respuesta (Response):** Si la petición se completa de forma satisfactoria, se recibirá un código 200 OK, junto con el punto de cuidado creado en formato JSON. En esta respuesta, el punto de cuidado ya dispondría de su identificador asignado por el sistema.
- **Posibles Códigos de Estado:**
  - **`200 OK`:** El punto fue creado correctamente.
  - **`400 Bad Request`:** La solicitud no pudo ser procesada debido a datos inválidos o faltantes en el cuerpo de la solicitud.
  - **`500 Internal Server Error`:** Ocurrió un error inesperado en el servidor.