Informe Técnico

Gestión de Bajas Técnicas "GesHTecK"

Equipo de Desarrollo

Nombre	Grupo	Usuario
Alina María de la Noval Armenteros	C-311	@Aliali_004
Abel de la Noval Pérez	C-311	@THE_KAIS3R
Fabio Víctor Alonso Bañobre	C-311	@fabio2k3
Frank Alberto Piz Torriente	C-311	@Frankpiz04

Diccionario de Datos

En esta sección se describen las entidades que conforman la base de datos del sistema GesHTecK, con el objetivo de facilitar el mantenimiento, comprensión y evolución del sistema por parte de desarrolladores futuros.

Tablas del Diseño Inicial

Entidad: Section (Sección)

- **Id (GUID, PK):** Identificador único de la sección.
- **Name (VARCHAR(100)):** Nombre de la sección.

Entidad: Department (Departamento)

- **Id (GUID, PK):** Identificador del departamento.
- **Name (VARCHAR(100)):** Nombre del departamento.
- **SectionId (GUID, FK):** Sección a la que pertenece.
- **ResponsibleId (GUID, FK):** Responsable asignado al departamento.

Entidad: EquipmentType (Tipo de Equipo)

- **Id (GUID, PK):** Identificador del tipo de equipo.
- **Name (VARCHAR(100)):** Nombre del tipo de equipo.

Entidad: Equipment (Equipo)

- **Id (GUID, PK):** Identificador del equipo.
- **Name (VARCHAR(200)):** Nombre o código identificador del equipo.
- **AcquisitionDate (DATETIME):** Fecha de adquisición.
- **EquipmentTypeId (GUID, FK):** Tipo de equipo asociado.
- **State (ENUM):** Estado actual del equipo.
- **LocationType (ENUM):** Tipo de ubicación del equipo.
- **DepartmentId (GUID, FK, NULL):** Departamento asignado (puede ser nulo).

Entidad: User (Usuario)

- **Id (GUID, PK):** Identificador único del usuario.
- **Name (VARCHAR(100)):** Nombre completo del usuario.
- **Email (VARCHAR(255)):** Correo electrónico.
- **PasswordHash (VARCHAR(255)):** Hash de la contraseña.
- **RoleId (INT, FK):** Rol asignado al usuario.

Entidad: Technical (Técnico)

- **Id (GUID, PK, FK):** Identificador del usuario técnico.
- **Experience (INT):** Años de experiencia.
- **Specialty (VARCHAR(100)):** Especialidad técnica.

Entidad: Director (Director)

- **Id (GUID, PK, FK):** Identificador del usuario director.

Entidad: Employee (Empleado)

- **Id (GUID, PK, FK):** Identificador del usuario empleado.

Entidad: Responsible (Responsable)

- **Id (GUID, PK, FK):** Identificador del usuario responsable.

Entidad: Receptor (Receptor)

- **Id (GUID, PK, FK):** Identificador del usuario receptor.

Entidad: Role (Rol)

- **Id (INT, PK):** Identificador del rol.
- **Name (VARCHAR(50)):** Nombre del rol (Administrator, Director, Technical, Employee, Responsible, Receptor).

Entidad: Maintenance (Mantenimiento)

- **Id (GUID, PK):** Identificador del mantenimiento.
- **EquipmentId (GUID, FK):** Equipo mantenido.
- **TechnicalId (GUID, FK):** Técnico que realizó el mantenimiento.
- **MaintenanceDate (DATETIME):** Fecha del mantenimiento.
- **MaintenanceTypeId (INT, FK):** Tipo de mantenimiento.
- **Cost (DECIMAL):** Costo del mantenimiento.

Entidad: EquipmentDecommission (Baja Técnica)

- **Id (GUID, PK):** Identificador de la baja técnica.
- **EquipmentId (GUID, FK):** Equipo dado de baja.
- **TechnicalId (GUID, FK):** Técnico que realizó la baja.
- **DepartmentId (GUID, FK):** Departamento asociado.
- **DestinyTypeId (INT, FK):** Tipo de destino final.
- **RecipientId (GUID, FK):** Receptor del equipo.
- **DecommissionDate (DATETIME):** Fecha de la baja técnica.
- **Reason (VARCHAR(500)):** Razón de la baja técnica.

Entidad: Transfer (Traslado)

- **Id (GUID, PK):** Identificador del traslado.
- **EquipmentId (GUID, FK):** Equipo trasladado.
- **SourceDepartmentId (GUID, FK):** Departamento de origen.
- **TargetDepartmentId (GUID, FK):** Departamento de destino.
- **ResponsibleId (GUID, FK):** Responsable que autorizó el traslado.
- **TransferDate (DATETIME):** Fecha del traslado.
- **CreatedAt (DATETIME):** Fecha de creación del registro.

Entidad: Assessment (Evaluación)

- **Id (GUID, PK):** Identificador de la evaluación.
- **TechnicalId (GUID, FK):** Técnico evaluado.
- **DirectorId (GUID, FK):** Director evaluador.
- **Score (VALUE OBJECT):** Puntuación de desempeño (0–100).
- **Comment (VARCHAR(500)):** Comentario de la evaluación.
- **AssessmentDate (DATETIME):** Fecha de la evaluación.

Entidad: EquipmentState (Estado de Equipo)

- **Id (INT, PK):** Identificador del estado del equipo.
- **Name (VARCHAR(50)):** Nombre del estado (Operative, InMaintenance, Decommissioned, Disposed).

Entidad: LocationType (Tipo de Ubicación)

- **Id (INT, PK):** Identificador del tipo de ubicación.
- **Name (VARCHAR(50)):** Tipo de ubicación (Department, Warehouse, Disposal).

Entidad: MaintenanceType (Tipo de Mantenimiento)

- **Id (INT, PK):** Identificador del tipo de mantenimiento.
- **Name (VARCHAR(50)):** Nombre del tipo de mantenimiento (Preventive, Corrective, Predictive).

Entidad: DestinyType (Tipo de Destino)

- **Id (INT, PK):** Identificador del tipo de destino final.
- **Name (VARCHAR(50)):** Tipo de destino (Department, Warehouse, Disposal).

Esquema del Diseño de la Aplicación

Visión General de la Arquitectura

La aplicación se estructura en cinco capas principales, organizadas jerárquicamente según su nivel de responsabilidad dentro del sistema:

- Capa de Presentación (*WebUI*)
- Capa de Servicios Web (*WebAPI*)
- Capa de Infraestructura (*Infrastructure*)
- Capa de Aplicación (*Application*)
- Capa de Dominio (*Domain*)

Cada capa cumple una función específica y se comunica únicamente con las capas adyacentes, garantizando un bajo acoplamiento y una alta cohesión entre los componentes del sistema.

Descripción de las Capas del Sistema

Capa de Dominio (Domain)

La capa de dominio constituye el núcleo del sistema y contiene las reglas de negocio fundamentales. En esta capa se definen:

- Las entidades principales del sistema, tales como Usuario, Equipo, Mantenimiento, Baja, Traslado y Valoración.
- Las jerarquías de herencia entre los distintos tipos de usuario (Técnico, Director, Responsable, Empleado y Receptor).
- Las reglas de negocio independientes de cualquier tecnología o framework específico.

Esta capa no depende de ninguna otra y no contiene lógica relacionada con persistencia, interfaces gráficas ni servicios externos, lo que facilita la validación, reutilización y evolución de las reglas del negocio.

Capa de Aplicación (Application)

La capa de aplicación define y coordina los casos de uso del sistema, actuando como intermediaria entre el dominio y las capas técnicas superiores.

En esta capa se implementan:

- Casos de uso asociados a la gestión de usuarios, equipos, mantenimientos, bajas técnicas y traslados.
- Servicios de aplicación que coordinan la ejecución de la lógica de negocio definida en el dominio.
- Interfaces que establecen los contratos para el acceso a la persistencia y otros servicios externos.

La existencia de esta capa permite modificar o ampliar los flujos funcionales del sistema sin afectar directamente al dominio ni a las capas de presentación.

Capa de Infraestructura (Infrastructure)

La capa de infraestructura contiene las implementaciones técnicas concretas requeridas para el funcionamiento del sistema, dando soporte a los contratos definidos en la capa de aplicación.

Entre sus responsabilidades se incluyen:

- Implementaciones de repositorios para el acceso y manipulación de la base de datos.
- Configuración del mapeo objeto-relacional y del contexto de persistencia.
- Integración con servicios externos o componentes técnicos adicionales, en caso de ser necesarios.

Esta capa depende de la capa de aplicación, respetando el principio de inversión de dependencias, y encapsula todos los detalles tecnológicos del sistema.

Capa de Servicios Web (WebAPI)

La capa WebAPI expone la funcionalidad del sistema a través de servicios REST, permitiendo la comunicación entre la lógica interna y la interfaz de usuario.

Sus principales funciones son:

- Definir controladores y endpoints para cada caso de uso.
- Realizar validaciones básicas de las solicitudes recibidas.
- Gestionar las respuestas HTTP y los códigos de estado correspondientes.

Esta capa no contiene lógica de negocio, delegando dicha responsabilidad completamente a la capa de aplicación.

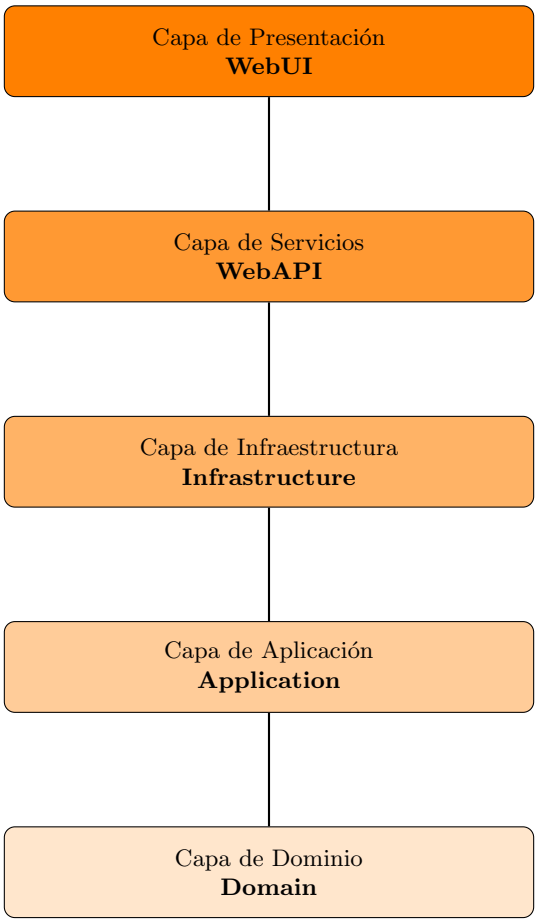
Capa de Presentación (WebUI)

La capa de presentación es la encargada de la interacción directa con los usuarios finales del sistema. En ella se implementan:

- Interfaces gráficas para la gestión de equipos, usuarios y procesos técnicos.
- Flujos de navegación, formularios y validaciones de entrada.

Esta capa consume exclusivamente los servicios expuestos por la WebAPI y no accede de forma directa ni a la base de datos ni a la lógica del dominio.

Esquema Visual de la Arquitectura



Esquema de las Clases Definidas y su Relación

En esta sección se describe la estructura de clases del sistema GesHTecK, así como las relaciones existentes entre ellas, con el objetivo de facilitar la comprensión del modelo de dominio y su mantenimiento futuro.

El diseño se basa en principios de orientación a objetos, utilizando herencia, composición y asociaciones claramente definidas, y se encuentra alineado con la arquitectura en capas descrita en el punto anterior.

Visión General del Modelo de Clases

El modelo de clases del sistema se organiza alrededor de las siguientes categorías principales:

- Usuarios y jerarquía de roles.
- Gestión de equipos y su ciclo de vida.
- Procesos técnicos asociados a los equipos.
- Estructura organizativa (departamentos y secciones).

Jerarquía de Usuarios

La clase **User** actúa como clase base dentro del sistema y representa a cualquier usuario autenticado. A partir de esta clase se define una jerarquía de herencia implementada mediante el patrón *Table Per Type (TPT)*.

- **User**: contiene los atributos comunes a todos los usuarios, como nombre, correo electrónico, contraseña y rol.
- **Director**: especialización de User con funciones de supervisión general.
- **Employee**: hereda de User e incorpora la pertenencia a un departamento.
- **Responsible**: especialización de Employee encargada de la gestión de un departamento o sección.
- **Technical**: hereda de User e incorpora información específica como especialidad y años de experiencia.

Esta jerarquía permite extender el sistema con nuevos tipos de usuarios sin modificar la clase base, respetando el principio de abierto/cerrado.

Gestión de Equipos

La clase **Equipment** representa los activos técnicos gestionados por el sistema. Cada equipo se asocia a un tipo específico definido por la clase **EquipmentType** y pertenece a una unidad organizativa.

- **Equipment** mantiene relaciones uno-a-muchos con:
  - **Maintenance**
  - **Transfer**
  - **EquipmentDecommission**
- **EquipmentType** define la categoría del equipo y se mantiene como una entidad independiente para evitar redundancia.

El estado del equipo y su tipo de ubicación se gestionan mediante enumeraciones inteligentes, garantizando coherencia y validación a nivel de dominio.

Procesos Técnicos

Los procesos que afectan al ciclo de vida de los equipos se modelan mediante clases específicas:

- **Maintenance**: representa las intervenciones técnicas realizadas sobre un equipo, incluyendo tipo de mantenimiento, fecha y costo.
- **Transfer**: modela los traslados de equipos entre ubicaciones o responsables.
- **EquipmentDecommission**: registra la baja definitiva de un equipo, indicando la causa y el destino final.

Estas clases se relacionan directamente con Equipment mediante asociaciones uno-a-muchos, reflejando el historial completo de cada activo.

Valoraciones Técnicas

La clase **Assessment** representa la evaluación del desempeño de los técnicos. Cada valoración se asocia a:

- Un técnico evaluado.
- Un evaluador.
- Una puntuación representada mediante un *Value Object*.

El uso de objetos de valor para la puntuación permite encapsular reglas de validación y evita inconsistencias en los datos.

Estructura Organizativa

La estructura organizativa del sistema se modela mediante las clases:

- **Section**: representa una unidad organizativa principal.
- **Department**: subdivisión de un departamento, con un responsable asignado.Department

Estas entidades permiten asociar usuarios y equipos a contextos organizativos concretos.

Consideraciones de Diseño

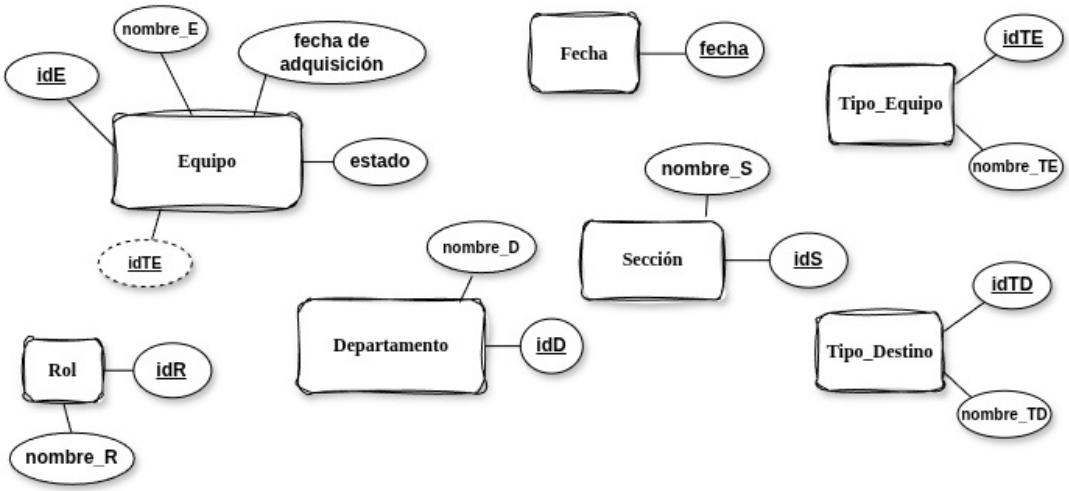
El modelo de clases presenta las siguientes características relevantes para el mantenimiento del sistema:

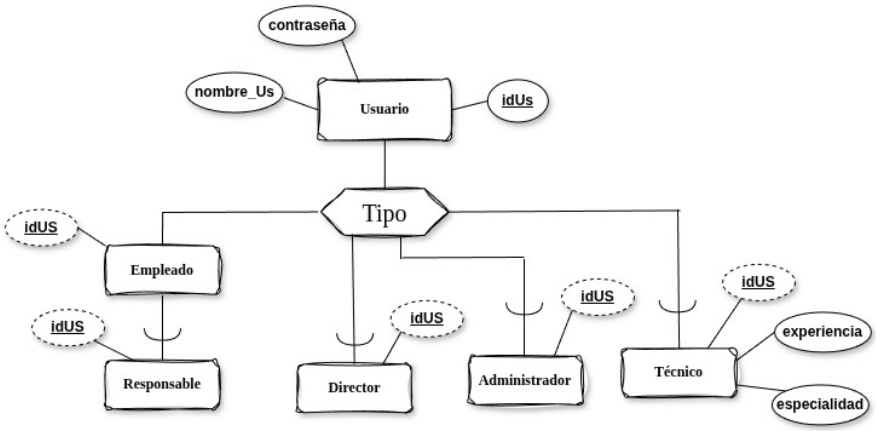
- Separación clara entre entidades, procesos y estructura organizativa.
- Uso controlado de herencia para evitar duplicación de atributos.
- Relaciones explícitas y bien definidas mediante asociaciones.
- Encapsulamiento de reglas de negocio en el dominio.

Este diseño facilita la evolución del sistema, la incorporación de nuevas funcionalidades y la comprensión del código por desarrolladores externos.

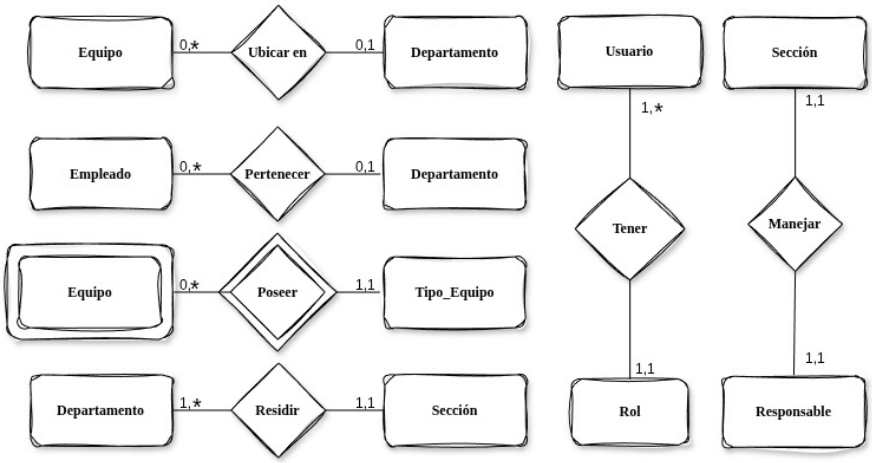
MODELO DE ENTIDAD RELACIONAL EXTENDIDO (MERX)

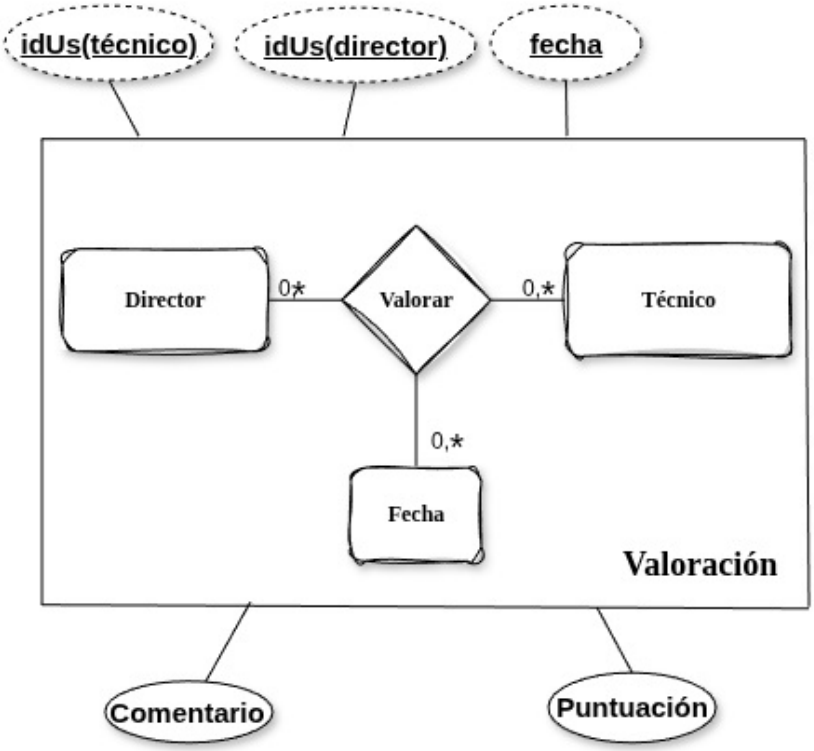
Entidades

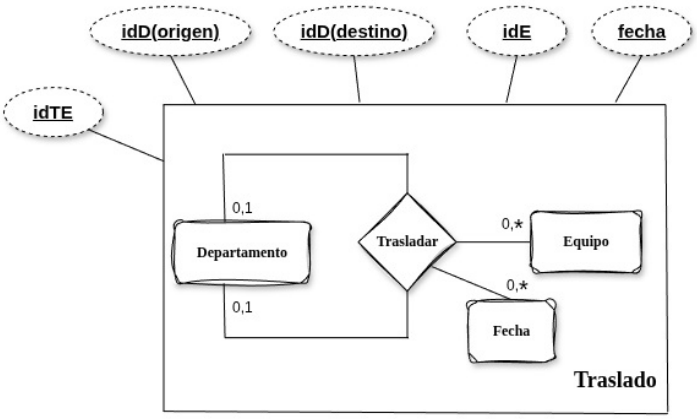


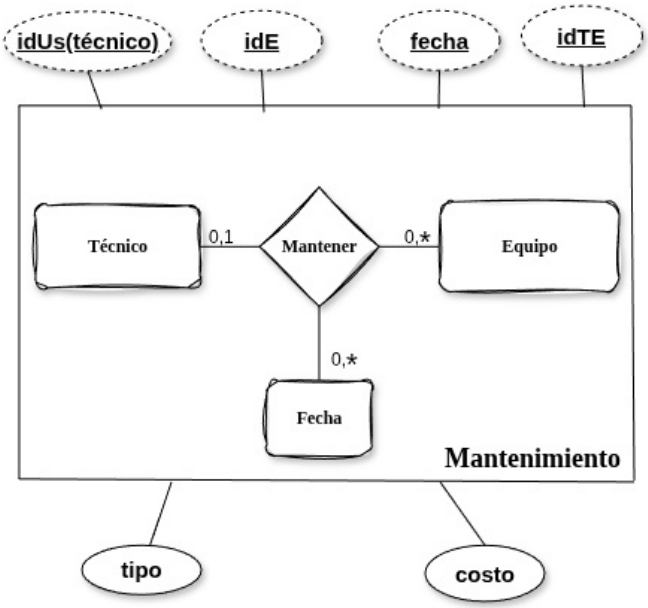


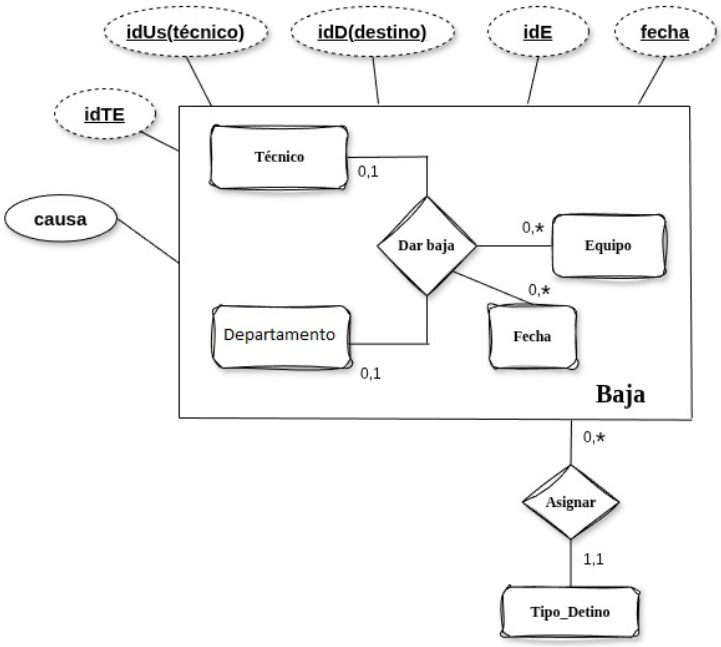
Relaciones


Equipo
0,*
Ubicar en
0,1
Departamento
Empleado
0,*
Pertenecer
0,1
Departamento
Equipo
0,*
Poseer
1,1
Tipo_Equipo
Departamento
1,*
Residir
1,1
Sección
Usuario
1,*
Tener
1,1
Rol
Sección
1,1
Manejar
1,1
Responsable


idUs(técnico)
idUs(director)
fecha
Director
0*
Valorar
0,*
Técnico
0,*
Fecha
Valoración
Comentario
Puntuación


idD(origen)
idD(destino)
idE
fecha
idTE
0,1
Departamento
Trasladar
0,*
Equipo
0,*
Fecha
0,1
Traslado


idUs(técnico)
idE
fecha
idTE
Técnico
0,1
Mantener
0,*
Equipo
0,*
Fecha
Mantenimiento
tipo
costo



Solución Propuesta

La solución propuesta para el sistema **GesHTecK** se fundamenta en una arquitectura moderna, modular y desacoplada, diseñada para garantizar la mantenibilidad, escalabilidad y claridad estructural del sistema a lo largo de su ciclo de vida. Esta propuesta responde a los requerimientos funcionales y no funcionales del sistema, así como a las buenas prácticas de la ingeniería de software.

El diseño adoptado permite que el sistema pueda ser comprendido, mantenido y extendido por desarrolladores que no hayan participado en su implementación inicial, reduciendo la dependencia del conocimiento implícito y facilitando su evolución futura.

Arquitectura Empleada

El sistema GesHTecK adopta una **arquitectura en capas basada en los principios de Clean Architecture**. Este enfoque promueve la separación clara de responsabilidades y establece un flujo de dependencias dirigido hacia el núcleo del dominio, garantizando que las reglas de negocio permanezcan independientes de tecnologías externas.

Las principales características de esta arquitectura son:

- Separación estricta entre lógica de negocio, aplicación, infraestructura y presentación.
- Independencia del dominio respecto a frameworks, bases de datos e interfaces de usuario.
- Bajo acoplamiento entre componentes y alta cohesión interna.
- Facilidad para pruebas, mantenimiento y extensión del sistema.

Este enfoque resulta especialmente adecuado para un sistema de gestión técnica como GesHTecK, donde la evolución de reglas de negocio y procesos operativos es frecuente.

Descripción de los Módulos del Sistema

La solución se organiza en los siguientes módulos principales, cada uno con una responsabilidad bien definida:

Módulo Domain

Constituye el núcleo del sistema y contiene las reglas de negocio fundamentales. En este módulo se definen:

- Entidades del dominio como Usuario, Equipo, Mantenimiento, Traslado y Baja Técnica.
- Jerarquías de herencia entre tipos de usuario, implementadas mediante el patrón TPT (Table Per Type).
- Value Objects para representar conceptos del dominio como correo electrónico, contraseñas y puntuaciones de evaluación.
- Enumeraciones enriquecidas (Smart Enums) para representar estados y tipos del sistema.

Este módulo no depende de ningún otro y no contiene código relacionado con persistencia, interfaces gráficas o servicios externos.

Módulo Application

El módulo de aplicación actúa como intermediario entre el dominio y las capas externas. Su función principal es coordinar los casos de uso del sistema.

Incluye:

- Servicios de aplicación que orquestan las operaciones del dominio.
- Definición de interfaces de repositorios y contratos de persistencia.
- Lógica de flujo para los procesos de negocio sin implementar detalles técnicos.

Esta capa permite modificar o extender los procesos del sistema sin afectar directamente al dominio ni a la infraestructura.

Módulo Infrastructure

Este módulo contiene las implementaciones técnicas concretas necesarias para el funcionamiento del sistema.

Entre sus responsabilidades se encuentran:

- Implementación de repositorios utilizando Entity Framework Core.
- Configuración del contexto de base de datos y mapeo objeto-relacional.
- Persistencia de datos en el sistema gestor de base de datos.

La infraestructura depende del módulo de aplicación, respetando el principio de inversión de dependencias.

Módulo WebAPI

El módulo WebAPI expone la funcionalidad del sistema mediante servicios REST, permitiendo la comunicación entre la lógica interna y la interfaz de usuario.

Sus responsabilidades incluyen:

- Definición de controladores y endpoints.
- Recepción y validación básica de solicitudes HTTP.
- Delegación de la lógica de negocio al módulo de aplicación.

Este módulo no contiene reglas de negocio, limitándose a actuar como capa de transporte.

Módulo WebUI

El módulo WebUI representa la capa de presentación del sistema y es el punto de interacción con los usuarios finales.

Incluye:

- Interfaces gráficas para la gestión de equipos, usuarios, mantenimientos y bajas técnicas.
- Validaciones de entrada y flujos de navegación.
- Consumo de los servicios expuestos por la WebAPI.

Este módulo no accede directamente a la base de datos ni a la lógica del dominio.

Flujo General de Funcionamiento

El flujo de funcionamiento del sistema sigue un esquema claramente definido:

- El usuario interactúa con la interfaz WebUI.
- La WebUI envía solicitudes a la WebAPI.
- La WebAPI delega la ejecución al módulo Application.
- El módulo Application coordina la lógica del dominio.
- La Infrastructure gestiona la persistencia de los datos.

Este flujo garantiza que las dependencias siempre se dirijan hacia el núcleo del dominio.

Decisiones Técnicas Relevantes

Entre las principales decisiones técnicas adoptadas en la solución se destacan:

- Uso de herencia TPT para modelar roles de usuario, facilitando la extensibilidad.
- Empleo de Value Objects para encapsular validaciones y asegurar consistencia del dominio.
- Utilización de Smart Enums para representar estados y tipos de forma segura.
- Uso de Entity Framework Core como ORM para simplificar el acceso a datos.

Estas decisiones contribuyen a la robustez y claridad del sistema.

Consideraciones de Mantenimiento y Evolución

La solución propuesta permite:

- Incorporar nuevos tipos de usuarios sin modificar el núcleo del dominio.
- Añadir nuevas funcionalidades mediante nuevos casos de uso en la capa de aplicación.
- Cambiar la tecnología de persistencia sin afectar la lógica de negocio.
- Facilitar la incorporación de nuevos desarrolladores al proyecto.

En conjunto, esta solución proporciona una base sólida y profesional para la gestión eficiente de bajas técnicas y procesos asociados, asegurando la evolución sostenible del sistema GesHTecK.