Universidad de Magallanes

Facultad de Ingeniería
Departamento de Ingeniería en Computación

Para la obtención del título
Ingeniero en Computación

'Sistema de gestión de préstamos e inventario del laboratorio del departamento de ingeniería en computación'

Hito 2: Diseño y validación del modelo de base de datos

Por Sr. Ariel López S.
Profesor guía: Dr. Pedro Alberti V.

19 de noviembre, 2025.

Índice

Índice de figuras

Índice de cuadros

1. Introducción

El presente documento aborda el diseño e implementación del modelo de base de datos para el sistema de inventario del Laboratorio del Departamento de Ingeniería en Computación de la Universidad de Magallanes. Con el fin de garantizar una estructura clara, escalable y eficiente, el modelo se organiza en tres grandes grupos funcionales: usuarios, productos y préstamos. Cada uno de estos grupos se encuentra subdividido en tablas modulares que permiten cubrir las necesidades actuales del laboratorio y facilitan su extensión futura.

La modularidad constituye un eje central del diseño, permitiendo que el sistema evolucione sin comprometer su integridad ni su funcionamiento. Para lograrlo, la implementación se realiza en Python, utilizando SQLAlchemy como ORM para mapear las entidades y sus relaciones, junto con Alembic como herramienta de control de versiones del esquema. De este modo, cualquier modificación futura en la estructura de datos podrá ser gestionada mediante migraciones documentadas y reproducibles. Finalmente, el capítulo concluye con la presentación del diagrama Entidad–Relación que sintetiza visualmente el modelo planteado.

2. Objetivos

2.1. Objetivo general

Diseñar y mantener un modelo de base de datos relacional que soporte de forma consistente, segura y escalable la gestión de usuarios, productos y préstamos del sistema LabDIC, permitiendo la operación concurrente de múltiples administradores y clientes, así como el registro del ciclo de vida de los préstamos.

2.2. Objetivos específicos

- Registrar la información de los usuarios del sistema, sus credenciales lógicas, roles y permisos, de manera que se puedan distinguir claramente administradores y clientes, y controlar su nivel de acceso.
- Representar el inventario de productos disponibles para préstamo, incluyendo sus características relevantes y su estado dentro del laboratorio.

- Modelar el ciclo completo de los préstamos, desde la solicitud inicial, pasando por la aprobación o rechazo, hasta la entrega del producto y su posterior devolución, manteniendo un historial trazable de todas las operaciones.
- Garantizar la integridad referencial entre las entidades mediante el uso adecuado de claves primarias y claves foráneas.
- Facilitar la evolución futura del sistema mediante un esquema modular y normalizado, apoyado por el control de versiones del esquema a través de Alembic, de modo que los cambios posteriores en los requisitos sean viables a nivel de base de datos.

3. Grandes grupos funcionales

Para poder llevar a cabo los objetivos, el modelo de base de datos se organiza en torno a tres grandes grupos funcionales que reflejan el funcionamiento del sistema:

Los **usuarios** interactúan con el sistema para solicitar un **préstamo** de **productos**, y dichas solicitudes son gestionadas por los administradores hasta concretar el préstamo y registrar su devolución.

En el cuadro 1 se resumen estos grupos y las entidades principales asociadas a cada uno de ellos.

Grupo	Propósito principal	Entidades representativas
Usuarios (`users`)	Gestionar la identidad, el rol y los permisos de quienes interactúan con el sistema.	`users`, `roles`
Productos (`products`)	Representar el inventario de ítems prestables y su estado dentro del laboratorio.	`products`, `devices`, `statuses`, `device_status_log`, `ubications`, `brands`, , `categories`, `models`
Préstamos (`loans`)	Modelar el ciclo de vida de los préstamos y su historial asociado.	`loan_request`, `loan_request_item`, `statuses`, `device_status_log`

Cuadro 1: Grandes grupos funcionales del modelo de datos de LabDIC.

4. División en tablas modulares

Los cambios futuros en la estructura de esta tabla, como la adición de nuevos campos o modificaciones en los existentes, se gestionarán fácilmente mediante el uso de migraciones controladas por Alembic, garantizando que el modelo de datos se mantenga escalable y flexible frente a futuros requerimientos.

4.1. Módulo de Usuarios

4.1.1. Tabla usuarios

El módulo de usuarios está diseñado para gestionar la autenticación, control de acceso y roles dentro del sistema. La tabla `users` almacena la información básica de cada usuario, incluyendo sus datos personales, credenciales y el estado de su cuenta. Este modelo permite identificar de manera única a los usuarios mediante su RUT, nombre de usuario o correo electrónico, y controlar el acceso a las funcionalidades del sistema a través de campos como `is_active` e `is_admin`.

Campo	Descripción
`id`	Identificador único del usuario. Es la clave primaria de la tabla.
`rut`	RUT del usuario, utilizado como identificador único. Este campo es obligatorio y único para cada usuario.
`name`	Nombre completo del usuario. Se utiliza para mostrar al usuario su nombre en el sistema.
`username`	Nombre de usuario único para acceder al sistema. Este campo también debe ser único.
`email`	Correo electrónico del usuario, utilizado para la autenticación y la comunicación. Este campo es único.
`password`	Contraseña cifrada del usuario. Es el campo crítico para la seguridad del acceso al sistema.
`phone`	Número de teléfono del usuario. Es un campo opcional.
`address`	Dirección física del usuario. También es un campo opcional.
`created_at`	Fecha y hora en que el usuario fue creado en el sistema. Se asigna automáticamente al momento de la creación.
`is_active`	Indica si la cuenta del usuario está activa. Permite habilitar o deshabilitar el acceso al sistema.
`is_admin`	Define si el usuario tiene permisos de administrador. Es un campo booleano que indica si el usuario tiene privilegios adicionales dentro del sistema.

Cuadro 2: Estructura de la tabla `users`.

4.1.2. Tabla de roles

La tabla `roles` se encarga de almacenar los diferentes roles que los usuarios pueden tener dentro del sistema. Cada rol tiene un nombre único y una descripción opcional que explica sus permisos o funciones dentro del sistema. Esta tabla permite gestionar y asignar roles a los usuarios, facilitando el control de acceso y la administración de permisos.

Campo	Descripción
`id`	Identificador único del rol. Es la clave primaria de la tabla.
`name`	Nombre del rol, utilizado para identificar el rol de manera única dentro del sistema (por ejemplo, 'admin', 'usuario'). Este campo es obligatorio y único.
`description`	Descripción del rol. Proporciona detalles sobre las funciones y permisos asociados con el rol. Es un campo opcional.

Cuadro 3: Estructura de la tabla `roles`.

La gestión de roles es fundamental para la administración de permisos en el sistema. A través de esta tabla, se pueden asignar diferentes capacidades a los usuarios según el rol que se les asigne, asegurando un control adecuado sobre el acceso y las funcionalidades disponibles para cada tipo de usuario.

4.2. Módulo de Productos

El módulo de productos es responsable de gestionar el inventario de ítems disponibles para préstamo dentro del sistema. La tabla `products` almacena la información de cada producto, como su nombre, descripción, cantidad disponible y otras propiedades relevantes para gestionar los préstamos de los productos. También se mantiene la relación con otras tablas como marcas, modelos y categorías de productos.

4.2.1. Tabla de productos

La tabla `products` es fundamental para el control del inventario, ya que permite gestionar y actualizar las existencias de los productos disponibles para préstamo. Además, almacena información relacionada con la descripción del producto y su relación con otras entidades, como las marcas y los modelos.

Campo	Descripción
`id`	Identificador único del producto. Es la clave primaria de la tabla.
`name`	Nombre del producto (por ejemplo, 'Laptop Dell XPS 13'). Este campo es único y obligatorio.
`brand_id`	ID de la marca asociada al producto. Es una clave foránea que referencia a la tabla `brands`. Este campo es opcional.
`model_id`	ID del modelo asociado al producto. Es una clave foránea que referencia a la tabla `models`. Este campo es opcional.
`category_id`	ID de la categoría del producto. Es una clave foránea que referencia a la tabla `categories`. Este campo es opcional.
`description`	Descripción detallada del producto. Este campo es opcional y proporciona información adicional sobre el producto.
`available_stock`	Cantidad de unidades disponibles del producto para préstamo. Este campo es obligatorio y se actualiza conforme se prestan los productos.
`is_active`	Indica si el producto está activo y disponible para préstamo. Este campo es booleano y tiene un valor predeterminado de `True`.
`created_at`	Fecha y hora de creación del producto en el sistema. Este campo se asigna automáticamente al momento de la creación del producto.

Cuadro 4: Estructura de la tabla `products`.

La tabla `products` es clave para el manejo eficiente del inventario y la gestión de los préstamos. Permite mantener un registro detallado de todos los productos, su disponibilidad y las características asociadas, y está diseñada para ser fácilmente ampliable en caso de que sea necesario agregar más atributos o relaciones con otras tablas.

4.2.2. Tabla de marcas

La tabla `brands` almacena las marcas asociadas a los productos. Cada marca tiene un nombre único, lo que permite categorizar los productos según su fabricante. Esta tabla está relacionada con la tabla `products`, de modo que cada producto puede tener una marca específica.

Campo	Descripción
`id`	Identificador único de la marca. Es la clave primaria de la tabla.
`name`	Nombre de la marca, utilizado para identificarla de manera única dentro del sistema (por ejemplo, 'Dell', 'HP', 'Apple'). Este campo es obligatorio y único.

Cuadro 5: Estructura de la tabla `brands`.

4.2.3. Tabla de modelos

La tabla `models` almacena los modelos de los productos. Cada modelo tiene un nombre único y está relacionado con los productos disponibles en el inventario. Al igual que la tabla de marcas, los productos pueden estar asociados a un modelo específico.

Campo	Descripción
`id`	Identificador único del modelo. Es la clave primaria de la tabla.
`name`	Nombre del modelo (por ejemplo, 'XPS 13', 'Spectre x360', 'MacBook Pro'). Este campo es obligatorio y único.

Cuadro 6: Estructura de la tabla `models`.

4.2.4. Tabla de categorías

La tabla `categories` agrupa los productos en categorías específicas, como 'Laptops', 'Tablets' o 'Smartphones'. Esto facilita la organización y búsqueda de productos dentro del inventario.

Campo	Descripción
`id`	Identificador único de la categoría. Es la clave primaria de la tabla.
`name`	Nombre de la categoría (por ejemplo, 'Laptops', 'Tablets', 'Smartphones'). Este campo es obligatorio y único.

Cuadro 7: Estructura de la tabla `categories`.

4.2.5. Tabla de dispositivos

La tabla `devices` almacena la información detallada de cada elemento de un producto existente en el inventario.

El motivo del nombre **dispositivo** surge a partir de que en el entorno de la tecnología, dentro del inventario del laboratorio, una gran cantidad y variedad de elementos dentro del mismo corresponde a *dispositivos*. Es por ello que, cada dispositivo está asociado con un producto específico y tiene un código interno, número de serie y un estado asociado. Además, idealmente, se puede rastrear su ubicación dentro del sistema.

Campo	Descripción
`id`	Identificador único del dispositivo. Es la clave primaria de la tabla.
`product_id`	ID del producto asociado al dispositivo. Es una clave foránea que referencia a la tabla `products`. Este campo es obligatorio.
`internal_code`	Código interno único para identificar el dispositivo. Este campo es opcional.
`serial_number`	Número de serie único del dispositivo. Este campo es opcional y único.
`status_id`	ID del estado del dispositivo. Es una clave foránea que referencia a la tabla `statuses`. Este campo es obligatorio.
`created_at`	Fecha y hora de creación del dispositivo en el sistema. Este campo se asigna automáticamente al momento de la creación.
`ubication_id`	ID de la ubicación del dispositivo. Es una clave foránea que referencia a la tabla `ubications`. Este campo es opcional.

Cuadro 8: Estructura de la tabla `devices`.

4.2.6. Tabla de estados

La tabla `statuses` permite gestionar los diferentes estados de los dispositivos y las solicitudes asociadas, como 'available', 'borrowed', 'under maintenance', entre otros. Los estados se utilizan para rastrear el ciclo de vida de cada dispositivo y solicitud dentro del sistema.

Campo	Descripción
`id`	Identificador único del estado. Es la clave primaria de la tabla.
`name`	Nombre del estado (por ejemplo, 'available', 'borrowed', 'under maintenance'). Este campo es obligatorio y único.

Cuadro 9: Estructura de la tabla `statuses`.

4.2.7. Tabla de ubicaciones

La tabla `ubications` gestiona las ubicaciones físicas de los dispositivos dentro del sistema. Cada ubicación tiene un nombre y una descripción opcional, lo que permite identificar fácilmente el lugar donde se encuentra cada dispositivo.

Campo	**Descripción**
`id`	Identificador único de la ubicación. Es la clave primaria de la tabla.
`name`	Nombre de la ubicación (por ejemplo, 'TI-1', 'Sala de servidores', 'Administración'). Este campo es obligatorio y único.
`description`	Descripción detallada de la ubicación. Este campo es opcional y proporciona información adicional sobre la ubicación.

Cuadro 10: Estructura de la tabla `ubications`.

4.2.8. Tabla de registros de cambios de estado de dispositivos

La tabla `device_status_logs` se utiliza para registrar cada cambio de estado de los dispositivos en el sistema. Cada vez que se cambia el estado de un dispositivo, se crea una entrada en esta tabla que incluye el estado anterior, el nuevo estado, la fecha y hora del cambio, el usuario que realizó la modificación y un comentario opcional sobre el cambio.

Campo	Descripción
`id`	Identificador único del registro de cambio de estado. Es la clave primaria de la tabla.
`device_id`	ID del dispositivo cuyo estado fue cambiado. Es una clave foránea que referencia a la tabla `devices`. Este campo es obligatorio.
`status_id`	ID del nuevo estado del dispositivo. Es una clave foránea que referencia a la tabla `statuses`. Este campo es obligatorio.
`timestamp`	Fecha y hora en que se registró el cambio de estado. Este campo se asigna automáticamente al momento de la creación del registro.
`user_id`	ID del usuario que realizó el cambio de estado. Es una clave foránea que referencia a la tabla `users`. Este campo es obligatorio.
`comment`	Comentario opcional sobre el cambio de estado, para dar contexto sobre el motivo del cambio. Este campo es opcional.

Cuadro 11: Estructura de la tabla `device_status_logs`.

4.3. Módulo de Prestamos

En este módulo, se gestionan las solicitudes de préstamo de dispositivos. Las tablas `device_status_logs` y `statuses`, que ya han sido descritas en las secciones 4.2.8 y 4.2.6, también son utilizadas en este módulo para registrar y gestionar el estado de las solicitudes y los dispositivos a lo largo de su ciclo de vida.

4.3.1. Tabla de solicitudes de préstamo

La tabla `loan_requests` gestiona las solicitudes de préstamo de los usuarios. Cada solicitud incluye información como el usuario que la realizó, el estado de la solicitud, las fechas relacionadas con la entrega y devolución del dispositivo, así como una razón opcional para el préstamo.

Campo	Descripción
`id`	Identificador único de la solicitud de préstamo. Es la clave primaria de la tabla.
`user_id`	ID del usuario que realiza la solicitud. Es una clave foránea que referencia a la tabla `users`. Este campo es obligatorio.
`request_date`	Fecha y hora en que se realizó la solicitud. Este campo se asigna automáticamente al momento de la creación de la solicitud.
`status_id`	ID del estado de la solicitud. Es una clave foránea que referencia a la tabla `statuses`. Este campo es obligatorio.
`reason`	Motivo opcional de la solicitud de préstamo (por ejemplo, "Necesito el dispositivo para un proyecto").
`delivery_date`	Fecha en que se entrega el dispositivo al usuario. Este campo es opcional.
`estimated_return_date`	Fecha estimada de devolución del dispositivo. Este campo es opcional.
`actual_return_date`	Fecha real de devolución del dispositivo. Este campo es opcional y se completa cuando se retorna el dispositivo.

Cuadro 12: Estructura de la tabla `loan_requests`.

4.3.2. Tabla de elementos de solicitudes de préstamo

La tabla `loan_request_items` gestiona los dispositivos individuales solicitados en una solicitud de préstamo. Cada elemento en esta tabla está asociado con una solicitud específica de préstamo y un dispositivo específico. Esto permite llevar un control detallado de los productos solicitados.

Campo	Descripción
`id`	Identificador único del elemento de la solicitud de préstamo. Es la clave primaria de la tabla.
`loan_request_id`	ID de la solicitud de préstamo asociada. Es una clave foránea que referencia a la tabla `loan_requests`. Este campo es obligatorio.
`device_id`	ID del dispositivo solicitado. Es una clave foránea que referencia a la tabla `devices`. Este campo es obligatorio.

Cuadro 13: Estructura de la tabla `loan_request_items`.

5. Modelo de base de datos

En la sección 4, se describieron de manera detallada las tablas del sistema, incluyendo sus campos y relaciones, con el objetivo de gestionar de forma eficiente el inventario y el ciclo de vida de los dispositivos. A partir de estas descripciones, el modelo de base de datos se implementa utilizando SQLAlchemy, que actúa como el ORM (Object-Relational Mapping) para facilitar la interacción entre el código Python y la base de datos relacional.

Además, para asegurar la escalabilidad y facilitar futuras modificaciones del esquema, la gestión de las migraciones de la base de datos se realiza con Alembic, una herramienta que permite aplicar cambios incrementales de manera controlada. Esta solución asegura que cualquier modificación en el modelo de datos, ya sea la adición de nuevas tablas o campos, se maneje de manera eficiente y sin afectar los datos existentes.

Todo el proyecto backend está configurado y gestionado dentro de un entorno virtual `.venv`, creado utilizando `uv from astral`. Esto garantiza la separación y aislamiento de dependencias, proporcionando un entorno controlado para el desarrollo y la ejecución del proyecto. La utilización de este entorno virtual facilita la instalación de las bibliotecas necesarias, como SQLAlchemy y Alembic, sin interferir con el sistema global.

5.1. Diagrama Entidad Relación

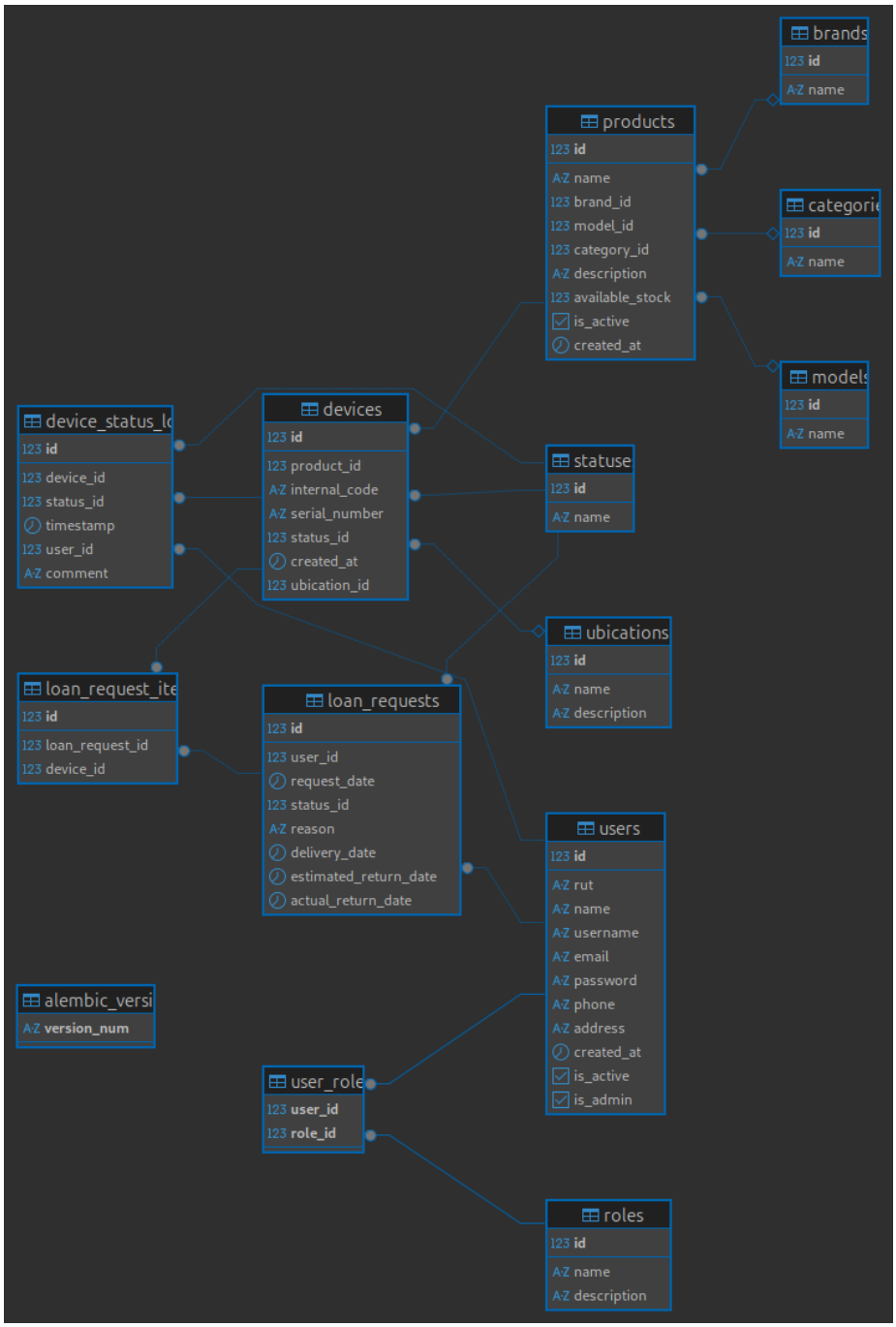

Figura 1: Diagrama ER del modelo para Inventario LabDIC

6. Conclusión

El modelo de base de datos desarrollado cumple con los principios fundamentales de modularidad, coherencia estructural y buenas prácticas de diseño. Las tablas fueron definidas de manera clara y consistente, reflejando adecuadamente los procesos del sistema de inventario y permitiendo una gestión precisa tanto de usuarios como de productos y solicitudes de préstamo. El diagrama Entidad–Relación confirma la solidez del esquema y la correcta articulación entre sus componentes.

Asimismo, la integración de SQLAlchemy y Alembic provee un marco robusto para la evolución continua del sistema, asegurando que cualquier cambio futuro quede documentado y pueda aplicarse mediante migraciones controladas. Con esta base, queda establecido el esqueleto técnico necesario para avanzar hacia el desarrollo completo del proyecto, integrando el backend y posteriormente el frontend, garantizando que ambos interactúen de forma adecuada con un modelo de datos bien estructurado y preparado para crecer.