



Sistema de Gestión Universitaria

Documentación del proyecto

Autor

Josué Ribero Duarte

✉ Email: jribero95@ucatolica.edu.co

👤 GitHub: [Josue-Ribero](#)

🎓 Código: 67001295

| | |
|----------|---------|
| version | 1.0.0 |
| FastAPI | 0.118.3 |
| SQLModel | 0.0.24 |
| Python | 3.8+ |
| SQLite | 3.51.0 |



Tabla de Contenidos

- Descripción
 - Características principales
 - Arquitectura
 - Reglas de negocio
 - Tecnologías
 - Estructura del proyecto
 - Características técnicas
 - Próximas mejoras
-



Descripción

Sistema REST API robusto desarrollado con **FastAPI** y **SQLModel** para la gestión integral de procesos académicos universitarios. Permite administrar estudiantes, cursos y matrículas con operaciones CRUD completas, manteniendo integridad referencial y un historial completo de todas las operaciones.

Problema que Resuelve

Las universidades necesitan un sistema eficiente para:

- **Gestionar cursos** con horarios, créditos y disponibilidad
 - **Administrar estudiantes** con su información académica
 - **Controlar matrículas** evitando conflictos de horarios y sobrecupo
 - **Mantener historial** de todas las transacciones para auditoría
-

Características Principales

Gestión de Cursos

- ☒ Registro con código único de 7 caracteres
- ☒ Asignación flexible de créditos (1-4)
- ☒ Múltiples franjas horarias disponibles
- ☒ Consulta de estudiantes matriculados en tiempo real
- ☒ Actualización de horarios sin perder información
- ☒ Preservación de histórico al eliminar






Gestión de Estudiantes

- ☒ Identificación única por cédula (7-10 dígitos)
- ☒ Email único para comunicaciones
- ☒ Seguimiento de progreso por semestre (1-12)
- ☒ Consulta rápida de cursos activos y finalizados
- ☒ Actualización de información académica
- ☒ Histórico completo de actividad estudiantil

Gestión de Matrículas

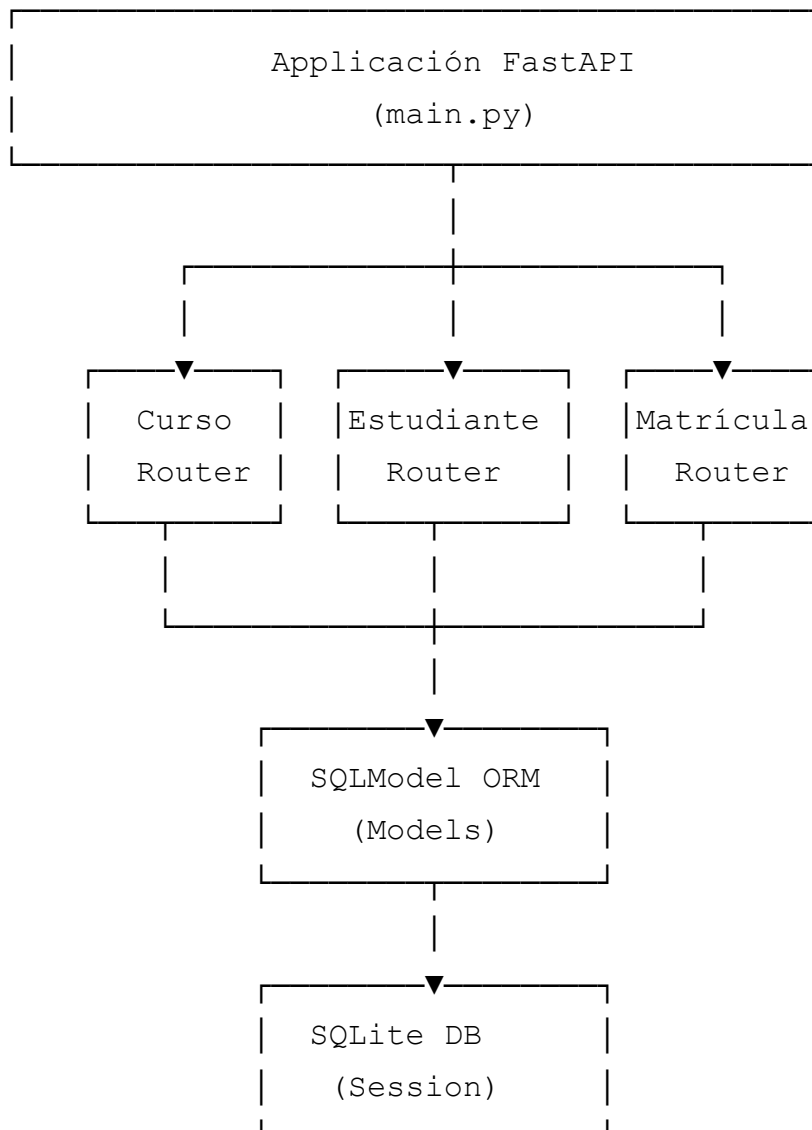
- ☒ Sistema de estados: **MATRICULADO** | **DESMATRICULADO** | **FINALIZADO**
- ☒ **Validación inteligente:** Un estudiante solo puede estar matriculado en un curso a la vez
- ☒ Rematriculación automática para cursos desmatriculados
- ☒ Prevención de repetición de cursos finalizados
- ☒ Timestamps automáticos de todas las operaciones
- ☒ Razones detalladas en histórico de eliminaciones

Sistema de Histórico

-  Tablas separadas para auditoría
 -  Preservación automática en cascada
 -  Registro de fechas de eliminación
 -  Razones detalladas de cada eliminación
 -  Trazabilidad completa de operaciones
-

Arquitectura

El sistema sigue una arquitectura de capas con separación de responsabilidades.
Esta hecho en capas de: **Negocio**, **Persistencia** y **Base de datos**:



Entidades Principales

Curso

```
{
  "id": 1,
  "codigo": "CH01001", # 7 caracteres, único
  "nombre": "Cultura Catolica",
  "creditos": "DOS", # Enum: UNO, DOS, TRES, CUATRO
  "horario": "SIETE_A_NUEVE"
}
```

Estudiante

```
{
  "id": 1,
  "cedula": "1234567890", # 7-10 dígitos, único
  "nombre": "Juana Torres",
  "email": "jtorres23@ucatolica.edu.co",
  "semestre": "QUINTO" # Enum: PRIMERO...DUODECIMO
}
```

Matrícula

```
{
  "id": 1,
  "codigo": "CH01001",
  "cedula": "1234567890",
  "fecha": "2025-10-24 15:28:56.337990",
  "matriculado": "MATRICULADO" # MATRICULADO | DESMATRICULADO | FINALIZA
}
```

Enumeraciones

CreditosCurso

- UNO → "1"
- DOS → "2"
- TRES → "3"
- CUATRO → "4"

HorarioCurso

- SIETE_A_NUEVE → 7:00 AM - 9:00 AM
- NUEVE_A_ONCE → 9:00 AM - 11:00 AM
- ONCE_A_UNA → 11:00 AM - 1:00 PM
- DOS_A_CUATRO → 2:00 PM - 4:00 PM
- CUATRO_A_SEIS → 4:00 PM - 6:00 PM
- SEIS_A_OCHO → 6:00 PM - 8:00 PM

Semestre

- PRIMERO ... DUODECIMO → "1"... "12"

EstadoMatricula

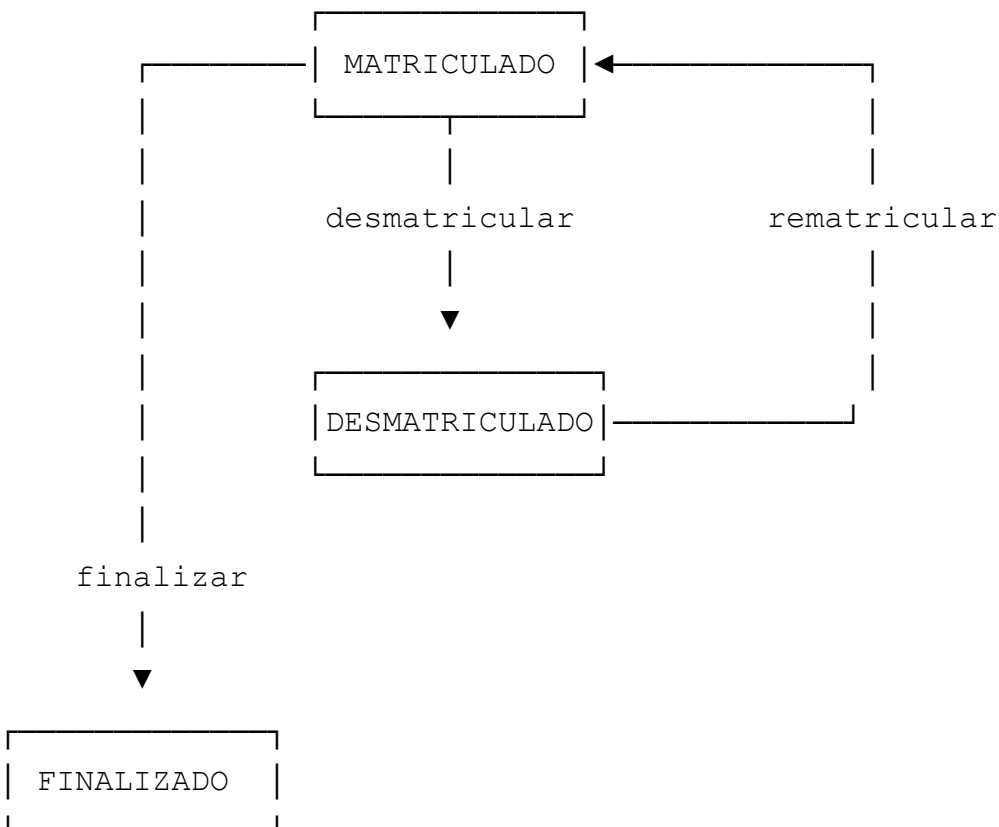
- MATRICULADO → Estudiante actualmente cursando
- DESMATRICULADO → Estudiante retirado del curso
- FINALIZADO → Curso completado exitosamente

Reglas de Negocio





Validaciones de Integridad

| Regla | Descripción | Validación |
|-------------------------------|---|--|
| Código Único | No pueden existir dos cursos con el mismo código | Código único a nivel de base de datos |
| Cédula Única | Cada estudiante debe tener una cédula diferente | Cédula única a nivel de base de datos |
| Email Único | Los emails no pueden duplicarse | Constraint de unicidad |
| Matrícula Activa Única | Un estudiante solo puede estar matriculado en un curso a la vez | Validación antes de inserción |
| Código 7 Caracteres | Los códigos de curso deben tener exactamente 7 caracteres | Validación con Field |
| Cédula Numérica | Las cédulas solo pueden contener dígitos | Validación con <code>.isdigit()</code> |

Lógica de Estados de Matrícula







Estados:

-  `MATRICULADO` → `DESMATRICULADO` (permitido)
-  `DESMATRICULADO` → `MATRICULADO` (rematriculación)
-  `MATRICULADO` → `FINALIZADO` (completar curso)
-  `FINALIZADO` → `cualquier estado` (bloqueado)



Cascada de Eliminación

Al eliminar un **Curso** o **Estudiante**, el sistema:

1.  Busca todas las matrículas relacionadas
2.  Guarda copias en tablas de histórico con:
 - Fecha de eliminación
 - Razón detallada
 - Estado original
3.  Preserva la entidad principal en tabla histórica
4.  Elimina registros de tablas activas

Razones de eliminación automáticas:

- `"Curso eliminado"` → Cuando se elimina un curso
- `"Curso finalizado - estudiante eliminado"` → Curso completado
- `"Curso desmatriculado - estudiante eliminado"` → Curso retirado
- `"Curso en progreso - estudiante eliminado"` → Curso activo

Backend

| Tecnología | Versión | Uso |
|------------|---------|-----------------------------------|
| FastAPI | 0.118.3 | Framework web moderno y rápido |
| SQLModel | 0.0.24 | ORM con validación Pydantic |
| Pydantic | 2.9.2 | Validación de datos |
| Uvicorn | latest | Servidor ASGI de alto rendimiento |

Base de Datos

| Tecnología | Versión | Uso |
|------------|---------|---|
| SQLite | 3.51.0 | Base de datos relacional ligera |
| SQLAlchemy | 2.0+ | Motor de base de datos (incluido en SQLModel) |

Lenguaje

| Tecnología | Versión | Características Usadas |
|------------|---------|----------------------------------|
| Python | 3.13+ | Type hints, async, enums, clases |

Descripción de Componentes

`main.py`

Punto de entrada de la aplicación. Configura FastAPI, registra routers y crea las tablas de la base de datos.

`db/db.py`

Gestiona la conexión a SQLite y proporciona sesiones de base de datos mediante inyección de dependencias.

models/

Define los modelos de datos con SQLAlchemy:

- **Base:** Campos compartidos para crear/actualizar
- **Tabla:** Modelo con tabla en DB y relaciones
- **Histórico:** Registros eliminados con timestamps

routers/

Implementa los endpoints REST con validaciones de negocio:

- Operaciones CRUD completas
- Validaciones de integridad
- Manejo de errores HTTP

utils/enum.py






Enumeraciones para valores predefinidos:

- Créditos, horarios, semestres, estados
-



Características Técnicas

Validaciones Automáticas

-  **Type Safety:** Validación de tipos con Pydantic
-  **Campos únicos:** Códigos, cédulas, emails
-  **Longitud:** Cédulas 7-10 dígitos, códigos 7 caracteres
-  **Formato:** Cédulas numéricas, emails válidos
-  **Normalización:** Mayúsculas en códigos/nombres, minúsculas en emails

Manejo de Errores

| Código | Significado | Cuándo ocurre |
|--------|-----------------------|--|
| 200 | OK | Operación exitosa |
| 201 | Created | Recurso creado |
| 400 | Bad Request | Datos inválidos o regla de negocio violada |
| 404 | Not Found | Recurso no encontrado |
| 500 | Internal Server Error | Error del servidor |



¿Cómo probar el API?

Swagger UI

1. Accede a <http://127.0.0.1:8000/docs>
2. Selecciona un endpoint
3. Click en "Try it out"
4. Completa los parámetros
5. Click en "Execute"



Contribuciones

Este proyecto fue desarrollado como **Segundo Parcial** de la asignatura **Desarrollo de Software**. Si deseas mejorar algo, tu contribución es bienvenida.



Próximas Mejoras

- ☐ Implementar autenticación JWT.
- ☐ Crear dashboard web de la app.
- ☐ Agregar tests unitarios con pytest.



Si te gusta este proyecto, dale una estrella en GitHub