Proyecto Final

SICU: "Sistema de Inscripciones para la Cuna Jardín UNSA"

Nota

Estudiantes	Escuela	Asignatura
Bedregal Coaguila, Karla M. Llaique Chullunquia, Jack F. Ramos Ochochoque, Elkin E. Vilca Huarca, Laura L.	Escuela Profesional de Ingeniera de Sistemas	Programación Web 2 Semestre: III Código: 1702122

Semestre académico	Fecha de inicio	Fecha de entrega
2025 - A	22 Julio 2025	27 Julio 2025

Índice

1. Introducción

1.1. Tipo de Sistema

El presente proyecto, titulado **SICU - Sistema de Inscripciones para la Cuna Jardín UNSA**, consiste en una aplicación web desarrollada con una arquitectura de cliente-servidor. El backend ha sido implementado utilizando el framework **Django**, haciendo uso del módulo **Django REST Framework (DRF)** para la creación de APIs, mientras que el frontend ha sido construido con el framework **Vue**, lo que permite una experiencia de usuario dinámica y moderna. El sistema está dirigido al personal administrativo de la **Cuna Jardín UNSA**, con el propósito de digitalizar y automatizar el proceso de inscripción de niños entre 0 y 5 años. Mediante esta plataforma se busca reemplazar los procedimientos manuales tradicionales, reduciendo errores, mejorando la organización institucional y optimizando el tiempo dedicado al registro y seguimiento de estudiantes, apoderados, docentes y cursos. La solución planteada facilita el acceso remoto a la información, gracias a su despliegue como aplicación web, y sienta las bases para futuras extensiones del sistema, como la integración con otros servicios, la generación de reportes o la incorporación de módulos gráficos.

1.2. Justificación

La implementación de un sistema web para la gestión de inscripciones en la Cuna Jardín UNSA responde a la necesidad de modernizar procesos administrativos que, hasta el momento, se realizaban de forma manual. Este cambio representa múltiples beneficios, entre los cuales destacan:

- **Reducción de errores:** Al eliminar el registro manual, se disminuyen los errores en la carga y gestión de datos, mejorando el seguimiento individual de cada niño inscrito.
- **Ahorro de tiempo:** La automatización de procesos permite realizar inscripciones, consultas y actualizaciones de manera ágil y eficiente.
- **Mejor organización:** El sistema facilita la generación de reportes estadísticos y el control ordenado de la información, apoyando la toma de decisiones administrativas.
- **Accesibilidad:** Al tratarse de una aplicación web, los responsables del sistema pueden acceder a la plataforma desde cualquier lugar con conexión a Internet.
- **Escalabilidad:** Gracias a su arquitectura modular, el sistema permite futuras mejoras e integraciones sin comprometer su funcionamiento actual.

1.3. Objetivo General

- Desarrollar un sistema informático que permita la administración digital de inscripciones en la Cuna Jardín UNSA, con una estructura orientada a objetos para facilitar su mantenimiento y escalabilidad.

1.4. Objetivos Específicos

- Modelar entidades clave del dominio educativo: estudiantes, apoderados, docentes, cursos e inscripciones.
- Implementar operaciones para registrar, modificar, consultar y eliminar información relevante.
- Asegurar integridad y consistencia de datos mediante reglas definidas en las clases.
- Preparar el sistema para su futura integración con interfaces gráficas o web.

2. Arquitectura del Sistema

2.1. Tecnologías Utilizadas

- **Git y GitHub:** Control de versiones y repositorios remotos.
- **Python 3.13:** Lenguaje principal para el backend.
- **Django 5:** Framework web backend en Python.
- **Django REST Framework (DRF):** Creación de APIs RESTful.
- **Vue 3:** Framework frontend basado en JavaScript.
- **JavaScript:** Lenguaje tipado para el desarrollo en Vue.
- **Bootstrap 5:** Estilos e interfaces responsivas.
- **Visual Studio Code:** Editor de código fuente.
- **PostgreSQL:** Base de datos relacional para el backend.
- **Supabase:** Base de datos PostgreSQL en la nube.
- **SoapUI:** Pruebas de servicios REST.
- **Vercel (plan gratuito):** Despliegue del backend en la nube.
- **Netlify (plan gratuito):** Despliegue del frontend en la nube.
- **Navegadores Web:** Chrome, Firefox, Edge, Brave, Opera.
- **Sistemas operativos:** Windows 11 y Linux.

2.2. Estructura del Proyecto Backend

- El contenido que se entrega para la parte de Backend en este laboratorio es el siguiente:

```
Backend/
|-- AppCuna
|   |-- admin.py
|   |-- apps.py
|   |-- migrations
|   |   |-- Student.py
|   |   |-- Inscription.py
|   |   |-- ...
|   |-- models
|   |-- serializers.py
|   |-- urls.py
|   \-- views.py
|-- manage.py
|-- MyDjangoProject
|   |-- settings.py
|   \-- urls.py
|-- venvf
|   |-- bin
|   |-- include
|   |-- lib
|   |-- pyvenv.cfg
|-- requirements.txt
```

2.3. Estructura del Proyecto Frontend

- El contenido que se entrega para la parte de Frontend en este laboratorio es el siguiente:

```
Frontend/
|-- public
|   |-- index.html
|   |-- favicon.ico
|   \-- _redirects
|-- src
|   |-- App.vue
|   |-- main.js
|   |-- assets
|   |   \-- logo.png
|   |-- components
|   |   |-- HelloWorld.vue
|   |   \-- common/
|   |-- router
|   |   \-- index.js
|   |-- services
|   |   \-- api.js
|   |-- store
|   |   \-- index.js
|   \-- views
|       |-- Dashboard.vue
|       |-- Login.vue
|       |-- Profile.vue
|       |-- Students.vue
|       |-- Grades.vue
|       |-- Attendance.vue
|       |-- Announcement.vue
|       |-- Payments.vue
|       |-- Inscriptions.vue
|       |-- Teachers.vue
|       |-- Courses.vue
|       |-- Documents.vue
|       |-- Periods.vue
|       |-- Chats.vue
|       |-- Home.vue
|       |-- about-us.vue
|       |-- Workloads.vue
|       \-- user/
|-- package.json
|-- package-lock.json
|-- vue.config.js
|-- netlify.toml
\-- README.md
```

2.4. Separación Cliente - Servidor (Vue + Django)

El sistema fue desarrollado bajo una arquitectura de cliente-servidor, dividiendo claramente las responsabilidades entre el frontend (cliente) y el backend (servidor).

- El **backend**, implementado con **Django 5** y **Django REST Framework**, se encarga de la lógica del negocio, la gestión de la base de datos, las reglas de validación y la exposición de los datos mediante una **API RESTful**. Esta API ofrece múltiples endpoints que permiten consultar, registrar, modificar y eliminar información relacionada con estudiantes, apoderados, docentes, cursos, inscripciones, entre otros.

- El **frontend**, desarrollado con **Vue 3**, consume estos servicios a través de peticiones HTTP (principalmente métodos GET y POST). Vue se encarga de renderizar las vistas del sistema, gestionar la navegación del usuario, validar formularios en el cliente y mostrar respuestas o mensajes.

Esta separación permite una mayor flexibilidad, ya que ambas partes pueden desarrollarse, probarse y desplegarse de manera independiente. Además, mejora la escalabilidad del sistema, facilitando la integración futura con otros clientes, como aplicaciones móviles u otros sistemas web.

3. Modelado de Datos

3.1. Entidades y relaciones

3.1.1. Student

Este modelo representa a los estudiantes (niños) inscritos en la Cuna Jardín. Incluye información personal, de salud, y vínculos parentales. Contiene validaciones internas para asegurar la integridad en campos como limitaciones físicas o seguro médico.

3.1.2. Mother

Modelo que almacena la información detallada de las madres. Incluye datos personales, laborales, nivel de instrucción y vínculo con la universidad. Al registrar a una madre se puede crear automáticamente un usuario asociado y enviarle sus credenciales por correo.

3.1.3. Father

(Nota: aunque no se muestra su definición, se infiere su existencia). Representa al padre del estudiante y se relaciona desde el modelo ‘Student‘.

3.1.4. Teacher

Representa a los docentes que dictan cursos en la Cuna. Al igual que las madres, se puede generar un usuario automáticamente y enviar credenciales. Se almacena su experiencia, formación y vínculo con la universidad.

3.1.5. Course

Contiene información sobre los cursos disponibles, como nombre, código, créditos, semestre y año. Es un modelo fundamental para la asignación de cargas docentes.

3.1.6. Period

Define los periodos académicos, como “2025-I”. Se puede activar un periodo para señalar cuál está vigente en el sistema.

3.1.7. Workload

Asocia a un docente con un curso y un periodo académico, incluyendo modalidad, sección, capacidad y horario. Es clave para organizar las inscripciones de los estudiantes.

3.1.8. Inscription

Representa la inscripción de un estudiante a una carga académica (workload). Incluye controles de unicidad para evitar inscripciones duplicadas y permite validaciones personalizadas.

3.1.9. Grade

Permite registrar notas de evaluaciones de los estudiantes en distintos tipos: exámenes, prácticas, tareas, etc. Soporta validaciones, ponderaciones y cálculo porcentual.

3.1.10. Attendance

Almacena la asistencia diaria del estudiante en una determinada inscripción. Registra si estuvo presente, ausente o llegó tarde. Asegura que no se duplique la asistencia por fecha.

3.1.11. Announcement

Gestiona los anuncios enviados por los docentes a los estudiantes en una carga académica. Cada anuncio tiene un título, contenido y fecha de publicación.

3.1.12. Chat

Modelo que permite la comunicación entre usuarios. Almacena el remitente, destinatario, mensaje y momento del envío. Está vinculado a una carga académica.

3.1.13. Document

Permite que los docentes adjunten archivos relevantes a un curso. Guarda el título, descripción, archivo y fecha de subida.

3.1.14. Payment

Registra los pagos realizados por los estudiantes. Almacena concepto, monto, fecha, estado del pago, número de recibo y comprobante digitalizado.

Resumen de Relaciones:
Los modelos se relacionan entre sí principalmente por claves foráneas. ‘Student‘ se relaciona con ‘Mother‘ y ‘Father‘. Los docentes (‘Teacher‘) se asocian con cursos y periodos a través de ‘Workload‘. Los estudiantes se inscriben a cargas académicas mediante ‘Inscription‘, y en base a estas se generan registros de notas (‘Grade‘), asistencias (‘Attendance‘), y pagos (‘Payment‘). El sistema también admite anuncios, documentos y mensajería interna para enriquecer la experiencia educativa.

3.2. Diagrama Entidad-Relación

3.2.1. Estructura Académica

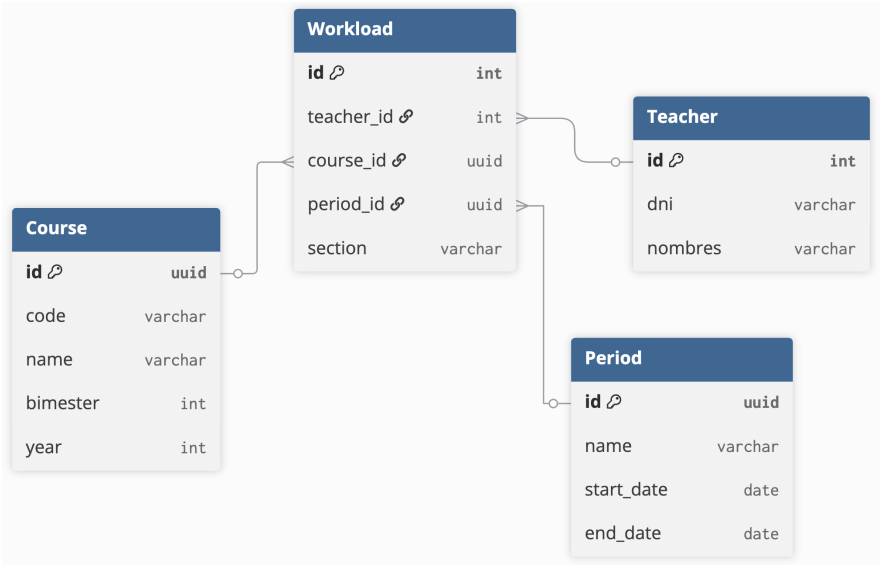

Figura 1: Relaciones entre cursos, docentes, periodos y cargas académicas.

3.2.2. Estudiantes, Apoderados e Inscripciones

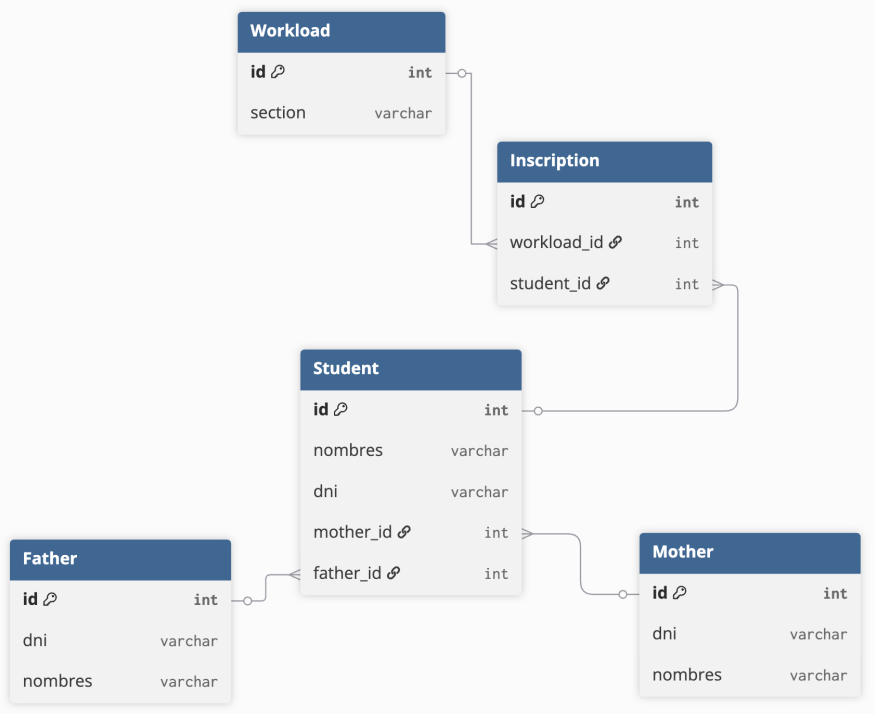

Figura 2: Relaciones entre estudiantes, apoderados e inscripciones.

3.2.3. Seguimiento Académico y Recursos

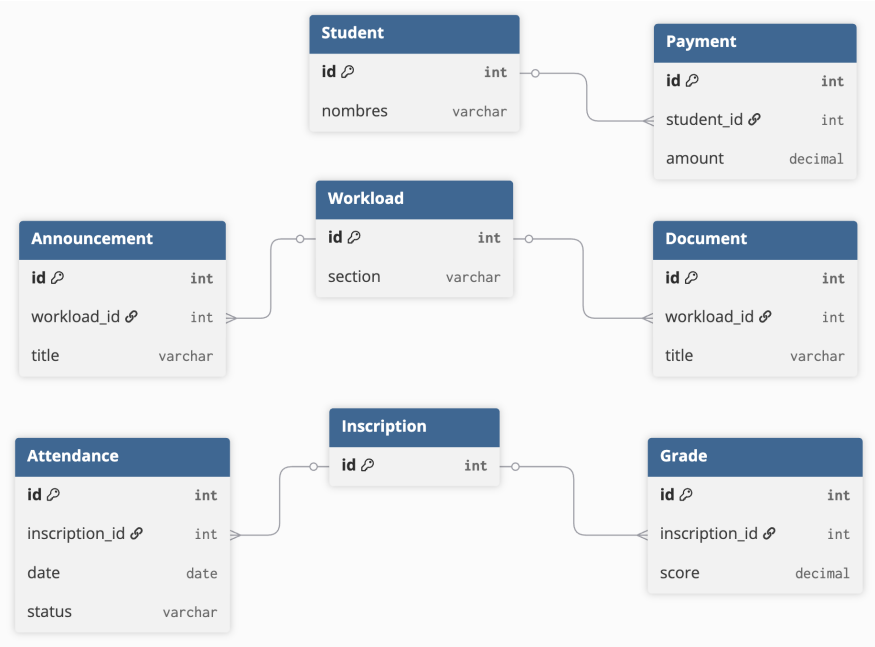

Figura 3: Relaciones para notas, asistencias, pagos y documentos.

3.3. Diccionario de Datos

En la construcción de software y en el diccionario de datos sobre todo se recomienda y se utilizará el idioma inglés para especificar objetos, atributos, etc.

Cuadro 1: Course

Atributo	Tipo	Nulo	Clave	Default	Descripción
id	UUID	No	PK	auto	Identificador único del curso
code	Cadena	Sí	Único	Ninguno	Código institucional
name	Cadena	No	No	Ninguno	Nombre del curso
credits	Decimal	Sí	No	Ninguno	Créditos académicos
bimester	Entero	No	No	1	Bimestre donde se dicta
year	Entero	No	No	1	Año en que se dicta
status	Booleano	No	No	True	Activo/Inactivo
created	FechaHora	No	No	auto	Fecha de creación
modified	FechaHora	No	No	auto	Fecha de modificación

Cuadro 2: Teacher

Atributo	Tipo	Nulo	Clave	Default	Descripción
user	FK (User)	Sí	Único	Null	Usuario vinculado
nombres	Cadena	No	No	Ninguno	Nombres del docente
apellido_paterno	Cadena	No	No	Ninguno	Apellido paterno
apellido_materno	Cadena	No	No	Ninguno	Apellido materno
dni	Cadena	No	Único	Ninguno	Documento Nacional de Identidad
fecha_nacimiento	Fecha	No	No	Ninguno	Fecha de nacimiento
edad	Entero	No	No	Ninguno	Edad actual
celular	Cadena	No	No	Ninguno	Número telefónico
correo_personal	Email	No	No	Ninguno	Correo del docente
grado_academico	Cadena	No	No	Licenciado	Nivel académico alcanzado
especialidad	Cadena	No	No	Ninguno	Especialidad docente
experiencia_docente	Entero	No	No	0	Años de experiencia
tipo_contrato	Cadena	No	No	Contratado	Tipo de contrato laboral
facultad	Cadena	No	No	Ninguno	Facultad a la que pertenece
escuela_profesional	Cadena	No	No	Ninguno	Escuela profesional
oficina	Cadena	Sí	No	Null	Oficina asignada
cv_documento	Archivo	Sí	No	Null	Archivo PDF del CV
activo	Booleano	No	No	True	Estado activo del docente

Cuadro 3: Student

Atributo	Tipo	Nulo	Clave	Default	Descripción
nombres	Cadena	No	No	Ninguno	Nombres del estudiante
apellido_paterno	Cadena	No	No	Ninguno	Apellido paterno
apellido_materno	Cadena	No	No	Ninguno	Apellido materno
fecha_nacimiento	Fecha	No	No	Ninguno	Fecha de nacimiento
dni	Cadena	No	Único	Ninguno	Documento Nacional de Identidad
sexo	Char	No	No	Ninguno	Sexo biológico (M/F)
edad	Entero	No	No	Ninguno	Edad actual
lugar_nacimiento	Cadena	No	No	Ninguno	Lugar de nacimiento
domicilio_actual	Cadena	No	No	Ninguno	Dirección actual
limitaciones_fisicas	Booleano	No	No	False	¿Tiene limitaciones físicas?

Atributo	Tipo	Nulo	Clave	Default	Descripción
especificar_limitaciones	Texto	Sí	No	Ninguno	Detalles de limitaciones (si aplica)
dificultades_sensoriales	Texto	Sí	No	Ninguno	Dificultades sensoriales
alergias	Texto	Sí	No	Ninguno	Alergias conocidas
tiene_seguro_medico	Booleano	No	No	False	¿Posee seguro médico?
nombre_seguro_medico	Cadena	Sí	No	Ninguno	Nombre del seguro
mother	FK	Sí	No	Null	Referencia a la madre
father	FK	Sí	No	Null	Referencia al padre

Cuadro 4: Father

Atributo	Tipo	Nulo	Clave	Default	Descripción
user	FK (User)	Sí	Único	Null	Usuario vinculado
nombres	Cadena	No	No	Ninguno	Nombre de la madre
apellido_paterno	Cadena	No	No	Ninguno	Apellido paterno
apellido_materno	Cadena	No	No	Ninguno	Apellido materno
dni	Cadena	No	Único	Ninguno	Documento de identidad
edad	Entero	No	No	Ninguno	Edad de la madre
fecha_nacimiento	Fecha	No	No	Ninguno	Fecha de nacimiento
celular	Cadena	No	No	Ninguno	Número de contacto
correo_electronico	Email	No	No	Ninguno	Correo institucional
estado_civil	Cadena	No	No	Soltera	Estado civil
vive_con_el_nino	Booleano	No	No	True	Vive con el niño
grado_instruccion	Cadena	No	No	secundaria	Nivel educativo
ocupacion_actual	Cadena	No	No	desocupado	Ocupación actual
tipo_dependiente	Cadena	Sí	No	Null	Dependencia laboral
rubro_actividad	Cadena	Sí	No	Null	Actividad económica
direccion_actividad	Texto	Sí	No	Null	Dirección del negocio
centro_estudios	Cadena	Sí	No	Null	Centro de estudios
tiene_vinculo_unsa	Booleano	No	No	False	Vínculo con la UNSA
area_labora	Cadena	Sí	No	Null	Área donde labora
regimen_laboral	Cadena	Sí	No	Null	Régimen laboral
tipo_docente	Cadena	Sí	No	Null	Tipo de contrato docente
tipo_administrativo	Cadena	Sí	No	Null	Tipo administrativo
facultad	Cadena	Sí	No	Null	Facultad asociada
escuela	Cadena	Sí	No	Null	Escuela profesional
oficina	Cadena	Sí	No	Null	Oficina laboral
credentials_sent	Booleano	No	No	False	¿Se enviaron credenciales?

Cuadro 5: Mother

Atributo	Tipo	Nulo	Clave	Default	Descripción
user	FK (User)	Sí	Único	Null	Usuario vinculado
nombres	Cadena	No	No	Ninguno	Nombre de la madre
apellido_paterno	Cadena	No	No	Ninguno	Apellido paterno
apellido_materno	Cadena	No	No	Ninguno	Apellido materno
dni	Cadena	No	Único	Ninguno	Documento de identidad
edad	Entero	No	No	Ninguno	Edad de la madre
fecha_nacimiento	Fecha	No	No	Ninguno	Fecha de nacimiento
celular	Cadena	No	No	Ninguno	Número de contacto
correo_electronico	Email	No	No	Ninguno	Correo institucional
estado_civil	Cadena	No	No	Soltera	Estado civil

Atributo	Tipo	Nulo	Clave	Default	Descripción
vive_con_el_nino	Booleano	No	No	True	Vive con el niño
grado_instruccion	Cadena	No	No	secundaria	Nivel educativo
ocupacion_actual	Cadena	No	No	desocupado	Ocupación actual
tipo_dependiente	Cadena	Sí	No	Null	Dependencia laboral
rubro_actividad	Cadena	Sí	No	Null	Actividad económica
direccion_actividad	Texto	Sí	No	Null	Dirección del negocio
centro_estudios	Cadena	Sí	No	Null	Centro de estudios
tiene_vinculo_unsa	Booleano	No	No	False	Vínculo con la UNSA
area_labora	Cadena	Sí	No	Null	Área donde labora
regimen_laboral	Cadena	Sí	No	Null	Régimen laboral
tipo_docente	Cadena	Sí	No	Null	Tipo de contrato docente
tipo_administrativo	Cadena	Sí	No	Null	Tipo administrativo
facultad	Cadena	Sí	No	Null	Facultad asociada
escuela	Cadena	Sí	No	Null	Escuela profesional
oficina	Cadena	Sí	No	Null	Oficina laboral
credentials_sent	Booleano	No	No	False	¿Se enviaron credenciales?

Cuadro 6: Period

Atributo	Tipo	Nulo	Clave	Default	Descripción
id	UUID	No	PK	auto	Identificador único
name	Cadena	No	No	Ninguno	Nombre del periodo (Ej: 2025-I)
start_date	Fecha	No	No	Ninguno	Fecha de inicio
end_date	Fecha	No	No	Ninguno	Fecha de fin
active	Booleano	No	No	False	¿Está activo actualmente?

Cuadro 7: Workload

Atributo	Tipo	Nulo	Clave	Default	Descripción
teacher	FK (Teacher)	No	FK	Ninguno	Profesor asignado
course	FK (Course)	No	FK	Ninguno	Curso correspondiente
period	FK (Period)	No	FK	Ninguno	Periodo académico
section	Cadena	No	No	Ninguno	Sección o grupo
capacity	Entero	No	No	30	Número de vacantes
modality	Cadena	No	No	Presencial	Modalidad de enseñanza
schedule	Texto	No	No	Ninguno	Horario textual
created	FechaHora	No	No	auto	Fecha de creación
modified	FechaHora	No	No	auto	Fecha de modificación

Cuadro 8: Inscription

Atributo	Tipo	Nulo	Clave	Default	Descripción
workload	FK (Workload)	No	FK	Ninguno	Carga docente del curso
student	FK (Student)	No	FK	Ninguno	Estudiante inscrito
status	Booleano	No	No	True	Estado de la inscripción
created	FechaHora	No	No	auto	Fecha de inscripción
modified	FechaHora	No	No	auto	Fecha de modificación

Cuadro 9: Grade

Atributo	Tipo	Nulo	Clave	Default	Descripción
inscription	FK (Inscription)	No	FK	Ninguno	Inscripción del estudiante
evaluation_type	Cadena	No	No	EXAM	Tipo de evaluación
score	Decimal	No	No	Ninguno	Nota obtenida
max_score	Decimal	No	No	20.00	Máxima nota posible
description	Cadena	Sí	No	Null	Descripción adicional
evaluation_date	Fecha	No	No	Ninguno	Fecha de evaluación
weight	Decimal	No	No	1.00	Peso de la nota
status	Booleano	No	No	True	Estado de la nota
created	FechaHora	No	No	auto	Fecha de creación
modified	FechaHora	No	No	auto	Fecha de modificación

Cuadro 10: Payment

Atributo	Tipo	Nulo	Clave	Default	Descripción
student	FK (Student)	No	FK	Ninguno	Estudiante que realiza el pago
concept	Cadena	No	No	Ninguno	Motivo del pago
amount	Decimal	No	No	Ninguno	Monto del pago
date	Fecha	No	No	auto	Fecha de pago
is_paid	Booleano	No	No	False	Estado del pago
receipt_number	Cadena	Sí	No	Null	Número de recibo
voucher	Archivo	Sí	No	Null	Comprobante escaneado

Cuadro 11: Document

Atributo	Tipo	Nulo	Clave	Default	Descripción
workload	FK (Workload)	No	FK	Ninguno	Curso al que pertenece
title	Cadena	No	No	Ninguno	Título del documento
description	Texto	Sí	No	Null	Descripción adicional
file	Archivo	No	No	Ninguno	Archivo subido
uploaded_at	FechaHora	No	No	auto	Fecha de subida

Cuadro 12: Chat

Atributo	Tipo	Nulo	Clave	Default	Descripción
workload	FK (Workload)	No	FK	Ninguno	Curso al que pertenece
sender	FK (User)	No	FK	Ninguno	Usuario que envía el mensaje
receiver	FK (User)	No	FK	Ninguno	Usuario que recibe el mensaje
message	Texto	No	No	Ninguno	Contenido del mensaje
timestamp	FechaHora	No	No	auto	Fecha de envío

Cuadro 13: Announcement

Atributo	Tipo	Nulo	Clave	Default	Descripción
workload	FK (Workload)	No	FK	Ninguno	Curso correspondiente
title	Cadena	No	No	Ninguno	Título del anuncio
content	Texto	No	No	Ninguno	Cuerpo del anuncio
published_at	FechaHora	No	No	auto	Fecha de publicación

Cuadro 14: Attendance

Atributo	Tipo	Nulo	Clave	Default	Descripción
inscription	FK (Inscription)	No	FK	Ninguno	Inscripción asociada
date	Fecha	No	No	Ninguno	Fecha de asistencia
status	Cadena	No	No	Ninguno	Presente, Ausente o Tardanza

4. Implementación del Backend

4.1. Creación del Proyecto y Aplicaciones en Django

- El desarrollo del backend comenzó con la creación del proyecto principal de Django utilizando el comando:
- Posteriormente, se generó la aplicación base del sistema denominada `AppCuna`, mediante el siguiente comando:

```
$ django-admin startproject MyDjangoProject
```

Listing 1: Creación del Proyecto Django

```
$ python manage.py startapp AppCuna
```

Listing 2: Creación de la aplicación AppCuna

A continuación, se registró la aplicación en el archivo `settings.py` dentro del arreglo `INSTALLED_APPS`, junto con otros módulos requeridos como `rest_framework` para la gestión de APIs.

4.2. Modelos y Migraciones

- Los modelos fueron definidos dentro del archivo `models/` de la aplicación `AppCuna`, cada uno representando una entidad del dominio del sistema: estudiantes, docentes, cursos, periodos, inscripciones, pagos, entre otros.
- Una vez definidos los modelos, se ejecutaron los siguientes comandos para generar y aplicar las migraciones a la base de datos:

```
$ python manage.py makemigrations
$ python manage.py migrate
```

Listing 3: Migraciones de los modelos

Estas operaciones crearon las tablas correspondientes en la base de datos PostgreSQL, las cuales fueron validadas mediante inspección directa y pruebas funcionales iniciales.

4.3. Panel de Administración

- Django proporciona un panel administrativo integrado, el cual fue configurado para gestionar de manera visual las entidades del sistema.
- Para ello, se registraron los modelos relevantes en el archivo `admin.py` de `AppCuna` mediante el decorador `@admin.register`, o utilizando la función `admin.site.register()`.
- Se creó un superusuario ejecutando el siguiente comando:

```
$ python manage.py createsuperuser
```

Listing 4: Creación del superusuario

El panel de administración permite crear, editar, eliminar y buscar registros de estudiantes, docentes, cursos, inscripciones, notas, pagos y asistencia, facilitando la gestión inicial del sistema mientras se desarrollan las interfaces de usuario en el frontend.

4.4. Migración a PostgreSQL y Conexión con Supabase

- Inicialmente, el sistema utilizaba `SQLite` como base de datos por defecto para pruebas locales. Para el entorno de producción, se realizó una migración a **PostgreSQL**, aprovechando los servicios en la nube de **Supabase**, que ofrece una instancia segura, escalable y de alto rendimiento.
- Para conectar con Supabase, se implementó la siguiente lógica en el archivo `settings.py`, utilizando variables de entorno y la biblioteca `dj_database_url`:

```
if 'DATABASE_URL' in os.environ:
    DATABASES = {
        'default': dj_database_url.parse(os.getenv('DATABASE_URL'))
    }
else:
    DATABASES = {
        'default': {
            'ENGINE': 'django.db.backends.postgresql',
            'NAME': os.getenv('DB_NAME', 'postgres'),
            'USER': os.getenv('DB_USER', 'postgres.zcsblgjwreggdewwdyxf'),
            'PASSWORD': os.getenv('DB_PASSWORD', 'CunaUnsa-'),
            'HOST': os.getenv('DB_HOST', 'aws-0-us-east-2.pooler.supabase.com'),
            'PORT': os.getenv('DB_PORT', '6543'),
            'OPTIONS': {
                'sslmode': 'require',
            }
        }
    }
```

Listing 5: Configuración de la base de datos con Supabase

- Para activar esta conexión, se definieron las variables de entorno correspondientes en el archivo `.env` o mediante configuración del servidor en producción. También se instaló la dependencia necesaria:
- Una vez configurado todo, se ejecutaron las migraciones de los modelos con el comando:

```
$ pip install dj-database-url psycopg2-binary
```

Listing 6: Instalación de dependencias

```
$ python manage.py migrate
```

Listing 7: Ejecucion de migraciones

Esta configuración permite una conexión segura (SSL) y flexible entre Django y Supabase, facilitando tanto el desarrollo como el despliegue de la aplicación en la nube.

4.5. Serializers y Vistas (DRF)

Instalación y configuración

- Para construir la API REST del backend se utilizó el paquete **Django REST Framework (DRF)**. Este framework permite crear endpoints robustos para modelos, aplicar validaciones personalizadas, gestionar permisos por rol y adaptar la respuesta según el usuario autenticado. DRF fue instalado mediante el siguiente comando:

- Luego, se añadió a `INSTALLED_APPS`:

```
        $ pip install djangorestframework
```

Listing 8: Instalación de Django REST Framework

```
INSTALLED_APPS = [
    ...,
    'rest_framework',
]
```

Listing 9: Agregar DRF al proyecto

Serializers y vistas destacadas

A continuación, se describen tres serializers clave del sistema y sus vistas asociadas.

1. StudentSerializer y StudentViewSet Este serializer impone restricciones dinámicas según el rol del usuario. Por ejemplo, un padre o madre autenticado solo puede editar campos específicos de su hijo.

```
class StudentSerializer(serializers.ModelSerializer):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        request = self.context.get('request')
        if request and request.user.is_authenticated:
            is_parent = (Father.objects.filter(user=request.user).exists() or
                         Mother.objects.filter(user=request.user).exists())
            if is_parent:
                for field in ['dni', 'fecha_nacimiento', 'sexo']:
                    self.fields[field].read_only = True

    class Meta:
        model = Student
        fields = '__all__'
```

Listing 10: StudentSerializer simplificado

La vista asociada filtra los estudiantes según el rol:

```
class StudentViewSet(viewsets.ModelViewSet):
    queryset = Student.objects.all()
    serializer_class = StudentSerializer

    def get_queryset(self):
        user = self.request.user
        if user.is_superuser:
            return Student.objects.all()
        try:
            teacher = Teacher.objects.get(user=user)
            workloads = Workload.objects.filter(teacher=teacher)
            inscriptions = Inscription.objects.filter(workload__in=workloads)
            return Student.objects.filter(inscription__in=inscriptions).distinct()
        except:
            ...
```

Listing 11: StudentViewSet resumido

2. GradeSerializer y GradeViewSet Este serializer filtra las inscripciones disponibles según el docente autenticado:

```
class GradeSerializer(serializers.ModelSerializer):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        request = self.context.get('request')
```

```
        if request and request.user.is_authenticated:
            try:
                teacher = Teacher.objects.get(user=request.user)
                teacher_workloads = Workload.objects.filter(teacher=teacher)
                self.fields['inscription'].queryset = Inscription.objects.filter(
                    workload__in=teacher_workloads
                )
            except:
                ...
    class Meta:
        model = Grade
        fields = '__all__'
```

Listing 12: GradeSerializer filtrando inscripciones

3. AnnouncementSerializer y AnnouncementViewSet Este serializer limita los workloads al docente correspondiente:

```
class AnnouncementSerializer(serializers.ModelSerializer):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        request = self.context.get('request')
        if request and request.user.is_authenticated:
            try:
                teacher = Teacher.objects.get(user=request.user)
                self.fields['workload'].queryset = Workload.objects.filter(teacher=teacher)
            except:
                ...
    class Meta:
        model = Announcement
        fields = '__all__'
```

Listing 13: AnnouncementSerializer con filtrado

4.6. URLs y Rutas

La configuración de rutas del proyecto permite acceder a todos los endpoints del backend mediante un sistema jerárquico organizado en dos niveles:

- El archivo `urls.py` principal del proyecto redirige desde la raíz hacia el prefijo `/api/`.
- El archivo `AppCuna/urls.py` registra todas las rutas de la aplicación utilizando el router de Django REST Framework y también define rutas personalizadas para autenticación.

A continuación se muestra el contenido de cada archivo clave.

```
from django.contrib import admin
from django.urls import path, include
from django.shortcuts import redirect

def redirect_to_api(request):
    return redirect('/api/')

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/', include('AppCuna.urls')),
    path('', redirect_to_api),
]
```

Listing 14: Archivo MyDjangoProject/urls.py

Este archivo configura dos rutas principales:

- `/admin/`: Panel administrativo de Django.

- `/api/`: Redirecciona al módulo de rutas de la aplicación `AppCuna`.

```python
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from .views import *

router = DefaultRouter()
router.register(r'students', StudentViewSet)
router.register(r'padres', FatherViewSet)
router.register(r'madres', MotherViewSet)
router.register(r'teachers', TeacherViewSet)
router.register(r'workloads', WorkloadViewSet)
router.register(r'courses', CourseViewSet)
router.register(r'inscriptions', InscriptionViewSet)
router.register(r'grades', GradeViewSet)
router.register(r'payments', PaymentViewSet)
router.register(r'announcements', AnnouncementViewSet)
router.register(r'attendance', AttendanceViewSet)
router.register(r'periods', PeriodViewSet)
router.register(r'chats', ChatViewSet)
router.register(r'documents', DocumentViewSet)

urlpatterns = [
    path('', include(router.urls)),

    path('auth/register/father/', RegisterFatherAPIView.as_view(), name='register-father'),
    path('auth/register/mother/', RegisterMotherAPIView.as_view(), name='register-mother'),
    path('auth/register/teachers/', RegisterTeacherAPIView.as_view(), name='register-teacher'),
    path('auth/login/', LoginAPIView.as_view(), name='login'),
    path('auth/logout/', LogoutAPIView.as_view(), name='logout'),
    path('auth/profile/', ProfileAPIView.as_view(), name='profile'),
    path('auth/credentials/', GetCredentialsAPIView.as_view(), name='get-credentials'),

    # Admin tokens
    path('auth/get-token/', GetUserTokenAPIView.as_view(), name='get-token'),
    path('auth/all-tokens/', GetAllTokensAPIView.as_view(), name='all-tokens'),
]
```

Listing 15: Archivo AppCuna/urls.py

Estructura General de Endpoints

Los endpoints generados automáticamente por el router se resumen en la siguiente tabla:

4.7. Autenticación con Tokens

El sistema implementa autenticación dual: mediante sesiones de Django (para navegación en navegador) y mediante tokens JWT (para consumo desde herramientas como Postman o aplicaciones frontend como Angular). Esta combinación permite flexibilidad tanto en pruebas como en producción.

Configuración en `settings.py`

Se activan ambas clases de autenticación en el backend mediante:

```python
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework_simplejwt.authentication.JWTAuthentication',
        'rest_framework.authentication.SessionAuthentication',
    ],
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.IsAuthenticated',
    ],
    ...
```

Endpoint	Descripción
`/api/students/`	Gestión de estudiantes
`/api/padres/`	Gestión de padres
`/api/madres/`	Gestión de madres
`/api/teachers/`	Gestión de docentes
`/api/workloads/`	Gestión de asignaciones docentes
`/api/courses/`	Gestión de cursos
`/api/inscriptions/`	Inscripciones de estudiantes a cursos
`/api/grades/`	Notas por inscripción
`/api/payments/`	Pagos de matrícula u otros
`/api/announcements/`	Anuncios por parte de docentes
`/api/attendance/`	Asistencia de estudiantes
`/api/periods/`	Periodos académicos
`/api/chats/`	Mensajes en el sistema
`/api/documents/`	Documentos institucionales

Cuadro 15: Principales endpoints generados por DRF Router

```
}
```

Listing 16: Configuración de autenticación en settings.py

Login con Tokens

El login se realiza vía el endpoint:

- `POST /api/auth/login/`

Este endpoint procesa credenciales (DNI y contraseña) mediante el `LoginAPIView`. Si son válidas, se activa automáticamente una sesión del lado del navegador y además se generan tokens JWT:

```
{
  "success": true,
  "tokens": {
    "access": "eyJ0eXAiOiJKV1QiLCJh...",
    "refresh": "eyJ0eXAiOiJKV1QiLCJhbGci..."
  },
  "user_type": "father",
  "user_data": { ... }
}
```

Listing 17: Respuesta JSON al hacer login

Logout y Perfil

El sistema permite cerrar la sesión y consultar los datos del usuario autenticado:

- `POST /api/auth/logout/` – cierra la sesión activa.
- `GET /api/auth/profile/` – devuelve información útil y enlaces según el rol (padre, madre, docente o admin).

Rutas para desarrollo (tokens de prueba)

Para facilitar pruebas, se habilitan rutas protegidas para desarrollo:

- `POST /api/auth/get-token/`: permite obtener un token JWT ingresando solo el DNI.

- `GET /api/auth/all-tokens/`: lista todos los usuarios registrados con sus respectivos tokens.

Estas rutas están marcadas en la documentación con advertencias y no deben usarse en entornos de producción.

Autenticación desde herramientas externas

Una vez obtenido el `access token`, puede utilizarse en herramientas como Postman, Insomnia o desde Vue, enviándolo en el encabezado HTTP:

```
curl -H "Authorization: Bearer <ACCESS_TOKEN>" http://localhost:8000/api/grades/
```

Listing 18: Ejemplo de uso con curl

Resumen de Endpoints de Autenticación

Ruta	Descripción
`POST /api/auth/login/`	Inicia sesión y retorna tokens JWT + sesión
`POST /api/auth/logout/`	Cierra la sesión activa
`GET /api/auth/profile/`	Devuelve datos del usuario autenticado y opciones según rol
`POST /api/auth/get-token/`	(Dev) Genera tokens ingresando el DNI
`GET /api/auth/all-tokens/`	(Dev) Lista todos los usuarios con sus tokens

Cuadro 16: Rutas principales del sistema de autenticación

5. Implementación desl Frontend

5.1. Creación del Proyecto Vue

Para desarrollar la interfaz de usuario del sistema, se utilizó Vue 3, un framework progresivo para la construcción de interfaces web. El proyecto fue creado utilizando la herramienta oficial de línea de comandos `@vue/cli`. A continuación, se detallan los archivos clave relacionados con la creación y configuración del proyecto.

```
{
"name": "cuna-frontend",
"version": "1.0.0",
"private": true,
"scripts": {
    "serve": "vue-cli-service serve",
    "build": "vue-cli-service build",
    "lint": "vue-cli-service lint"
    },
"dependencies": {
    "@fortawesome/fontawesome-free": "^7.0.0",
    "axios": "^1.6.0",
    "bootstrap": "^5.3.0",
    "core-js": "^3.8.3",
    "vue": "^3.2.13",
    "vue-router": "^4.0.3",
    "vuex": "^4.0.0"
    }
}
```

Listing 19: package.json - Configuración del Proyecto

El archivo `package.json` define las dependencias principales del proyecto, como Vue, Vuex, Vue Router, Bootstrap y Axios. También incluye scripts útiles como `serve` para iniciar el servidor de desarrollo local y `build` para generar la versión optimizada del sitio.

```
module.exports = {
    publicPath: '/',
    outputDir: 'dist',
    assetsDir: 'assets',
        devServer: {
            proxy: {
            '/api': {
                target: 'https://cuna-unsa-production.up.railway.app',
                changeOrigin: true,
                secure: true,
                }
            }
        }
}
```

Listing 20: vue.config.js - Proxy de Desarrollo

El archivo `vue.config.js` permite definir opciones de compilación y desarrollo. En este caso, se configura un proxy que redirige automáticamente las solicitudes a `/api` hacia el backend desplegado en Railway, lo cual facilita las pruebas sin problemas de CORS durante el desarrollo local.

5.2. Componentes y Servicios

En el desarrollo del frontend con Vue.js, se estructuraron los distintos elementos visuales y de interaccion del sistema a traves de componentes reutilizables y servicios de conexion con la API. A continuacion, se describe la logica de algunos componentes clave.

Componente de Login

El archivo `Login.vue` contiene el formulario de autenticacion que permite a los usuarios ingresar al sistema. Utiliza `v-model` para enlazar datos con los campos del formulario y acciones de Vuex para despachar la operacion de login.

```
<form @submit.prevent="handleLogin">
<input v-model="form.username" required />
<input v-model="form.password" type="password" required />
<button type="submit" :disabled="loading">
    {{ loading ? 'Iniciando...' : 'Iniciar Sesion' }}
</button>
</form>
```

Listing 21: Formulario de Login (Login.vue)

En la seccion de metodos se utiliza una accion del store para realizar la peticion de autenticacion al backend:

```
methods: {
...mapActions(['login']),
async handleLogin() {
    try {
    const res = await this.login(this.form)
    this.$router.push('/dashboard')
    } catch (error) {
    console.error('Error en login:', error)
    }
}
}
```

Listing 22: Logica de Login

Vista Principal: Dashboard

El componente `Dashboard.vue` representa el panel principal al que se redirige luego del login. En esta vista se integran otros componentes secundarios y datos obtenidos desde la API. Aunque su logica es sencilla,

sirve como punto de partida para acceder a distintas secciones como cursos, estudiantes o calificaciones.

Componente Comun: Barra de Navegacion

En la carpeta `components/common/` se implemento una barra de navegacion dinamica que muestra diferentes enlaces segun el estado de autenticacion del usuario. Utiliza `router-link` para la navegacion y datos provenientes del `store` para personalizar la interfaz.

```
<router-link class="nav-link" to="/dashboard">Pagina Principal</router-link>
<router-link class="nav-link" to="/grades">Calificaciones</router-link>
<a class="dropdown-item text-danger" href="#" @click="handleLogout">
Cerrar Sesion
</a>
```

Listing 23: Fragmento de Navbar.vue

Servicios de Conexion

El archivo `services/api.js` contiene la instancia de Axios configurada para conectarse con el backend desplegado, permitiendo que todos los componentes puedan acceder a los datos protegidos mediante un proxy local durante el desarrollo.

```
import axios from 'axios'

const api = axios.create({
baseURL: '/api',
timeout: 5000
})

export default api
```

Listing 24: Instancia de Axios

5.3. Conexión a la API REST

Para establecer la comunicación entre el cliente (Vue.js) y el servidor backend (Django + DRF), se creó un archivo `api.js` dentro del directorio `src/services/`. Este módulo configura la librería `Axios` para realizar solicitudes HTTP con autenticación mediante tokens.

- Se define una instancia de `Axios` con la URL base tomada desde una variable de entorno (`VUE_APP_API_URL`).
- Se usan interceptores para incluir el token de autenticación almacenado en `localStorage` en cada solicitud.
- Ante errores de autenticación (`401 Unauthorized`), el sistema elimina el token y redirige automáticamente al login.
- Las rutas implementadas permiten operaciones completas (GET, POST, PUT, DELETE) sobre entidades como estudiantes, docentes y cursos.

```
const apiInstance = axios.create({
  baseURL: process.env.VUE_APP_API_URL,
  headers: {
    'Content-Type': 'application/json',
  },
  timeout: 10000,
  withCredentials: true,
});

apiInstance.interceptors.request.use((config) => {
  const token = localStorage.getItem('token');
```

```
  if (token) {
    config.headers.Authorization = `Bearer ${token}`;
  }
  return config;
});
```

Listing 25: Ejemplo de configuración de Axios en api.js

Ejemplo de consumo en un componente: el método `login` desde la vista `Login.vue` invoca `api.login()` usando datos del formulario.

```
methods: {
  async handleLogin() {
    try {
      const res = await api.login(this.form);
      this.$router.push('/dashboard');
    } catch (error) {
      console.error('Error en login:', error);
    }
  }
}
```

Listing 26: Invocación del login

Pendiente: configurar correctamente el archivo `vue.config.js` para habilitar el proxy en desarrollo local:

```
devServer: {
  proxy: {
    '/api': {
      target: 'http://localhost:8000',
      changeOrigin: true,
      secure: false,
    }
  }
}
```

Listing 27: vue.config.js – proxy en desarrollo

6. Interfaces de Usuario

6.1. Flujo del Usuario Final (CRUD - Core Business)

El sistema CUNA UNSA ofrece una interfaz amigable donde los distintos tipos de usuarios (principalmente estudiantes y administradores) interactúan con los datos del sistema a través de operaciones CRUD (Crear, Leer, Actualizar y Eliminar). A continuación, se detalla el flujo general desde el punto de vista del usuario final:

- **Inicio de Sesión:** El usuario accede a la página de login (`/login`), ingresa sus credenciales y, si son válidas, se redirige al `Dashboard`. El token de autenticación se almacena en el navegador y se utiliza en cada petición.
- **Visualización de Información (Read):** Desde el `Dashboard`, el usuario puede acceder a distintas vistas mediante el menú de navegación:
  - `/students` para ver la lista de estudiantes.
  - `/teachers` para docentes registrados.
  - `/courses`, `/grades`, `/workloads`, entre otros.

  Estas vistas consumen datos directamente desde la API mediante llamadas GET.

- **Registro y Edición (Create / Update):** En vistas como `/students` o `/teachers`, los formularios permiten al administrador registrar nuevos datos (POST) o editar información existente (PUT). Los formularios están conectados con el archivo `api.js`, que centraliza todas las operaciones de escritura hacia el backend.
- **Eliminación de Registros (Delete):** Desde las vistas de lista, se puede eliminar un registro específico (estudiante, curso, etc.) usando botones o iconos asociados. Esto genera una llamada DELETE a la API, actualizando la vista inmediatamente.
- **Módulos adicionales:** Además del núcleo CRUD, el sistema permite:
  - **Ver calificaciones** asignadas por curso.
  - **Administrar inscripciones** por periodo académico.
  - **Gestionar pagos, documentos y anuncios** del sistema.
- **Flujo de sesión:** En todo momento, se verifica la validez del token. Si caduca o es inválido, el usuario es redirigido automáticamente al login mediante interceptores configurados.
- **Experiencia centrada en el usuario:** Toda la navegación está protegida por rutas autenticadas y los estados globales se gestionan con Vuex para mantener la sesión y los datos consistentes a lo largo de la aplicación.

Resumen: El sistema implementa un flujo completo de negocio basado en operaciones CRUD, asegurando que los usuarios puedan administrar los datos relevantes con facilidad, eficiencia y seguridad. La estructura modular permite expandir funcionalidades futuras sin afectar la experiencia del usuario final.

6.2. Pantallas Clave (Inicio, Inscripción, Consulta, etc.)

Figura 4: Vista de inicio de sesión del sistema

Figura 5: Pantalla principal de bienvenida al ingresar al sistema

Figura 6: Sección de gestión de cursos para el usuario

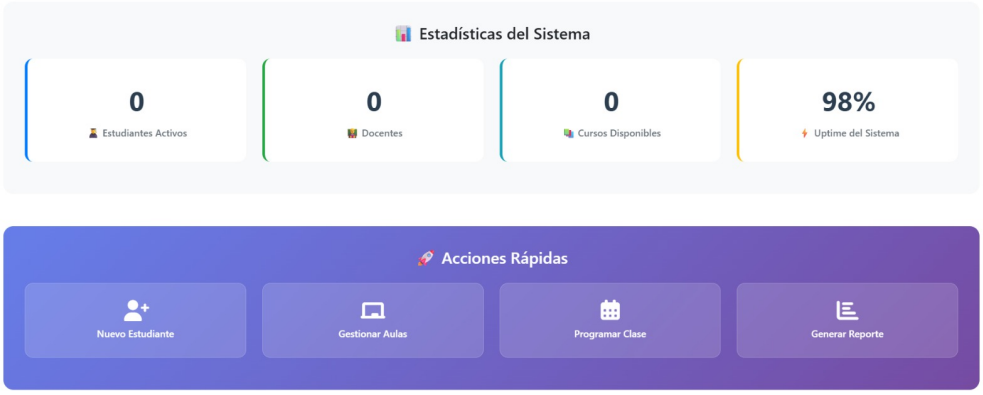

Figura 7: Panel administrativo para usuarios con privilegios

7. Despliegue del Proyecto

7.1. Despliegue del Backend en Render

Para hacer accesible el backend desarrollado con Django, se utilizó la plataforma Render, la cual permite el despliegue continuo mediante la integración con GitHub. A continuación, se detallan los aspectos clave del

despliegue:

- **Repositorio Git conectado:** Se enlazó el repositorio público del backend alojado en GitHub, disponible en la URL: `https://github.com/KarlaBedregal/CunaUnsa\_`.
- **Framework utilizado:** El proyecto fue desarrollado con Django y configurado para funcionar con una base de datos PostgreSQL remota, empleando variables de entorno para una mayor seguridad y portabilidad.
- **Archivo de entrada (entry point):** Render utiliza el archivo `manage.py` como punto de entrada para ejecutar las tareas administrativas del servidor.
- **Base de datos:** Se usó un servicio externo proporcionado por Supabase, el cual hospeda una instancia de PostgreSQL. La conexión se establece mediante variables de entorno definidas en la sección de configuración del servicio Render.
- **Variables de entorno configuradas en Render:**
  - `DB_NAME` = `postgres`
  - `DB_USER` = `postgres.zcsblgjwreggdewwdyxf`
  - `DB_PASSWORD` = `CunaUnsa-`
  - `DB_HOST` = `aws-0-us-east-2.pooler.supabase.com`
  - `DB_PORT` = `6543`
  - `SECRET_KEY` = `[clave secreta de Django]`
  - `DEBUG` = `False`
  - `ALLOWED_HOSTS` = `[çuna-unsa-backend.onrender.com"]`
- **Comandos de despliegue configurados:**
  - `Build command: pip install -r requirements.txt`
  - `Start command: gunicorn MyDjangoProject.wsgi`
- **Estrategia de despliegue:** Se configuró el servicio como `Web Service`, con entorno `Python 3`, usando despliegue automático desde la rama principal (`main`) del repositorio.
- **URL pública del backend:** `https://cuna-unsa-backend.onrender.com`
  Esta URL permite a los clientes frontend y usuarios externos consumir la API REST del sistema desde cualquier ubicación.

7.2. Despliegue del Frontend en Netlify

Para facilitar el acceso público a la interfaz del sistema, se utilizó la plataforma Netlify para el despliegue del frontend desarrollado en Vue 3.

- **Repositorio del Proyecto**: El código fuente del frontend se encuentra disponible en GitHub: `https://github.com/lau230595/CunaFrontend`https://github.com/lau230595/CunaFrontend
- **Configuración de Build**: Se configuró el archivo `netlify.toml` en la raíz del proyecto con las siguientes instrucciones:

```
[build]
command = "npm run build"
publish = "dist"

[[redirects]]
from = "/*"
to = "/index.html"
status = 200
```

Listing 28: Archivo netlify.toml

- **Comando de Build**: El comando de compilación para producción utilizado fue:

```
$ npm run build
```

- **Ruta de Publicación**: La carpeta generada tras el build es `dist`, la cual se especifica como carpeta a publicar en la configuración de Netlify.
- **Variables de Entorno**: Se requiere configurar una variable de entorno llamada `VUE_APP_API_URL` para que el frontend se comunique con la API REST del backend desplegado.

7.3. Conexión entre Frontend y Backend

La comunicación entre el frontend (Vue 3) y el backend (Django + Django REST Framework) se realiza mediante peticiones HTTP a través de Axios, una librería JavaScript incluida como dependencia del proyecto.

- **Archivo central de conexión:** En `src/services/api.js` se encuentra centralizada toda la lógica de consumo de la API. Allí se configura:
  - La URL base de la API, leída desde una variable de entorno: `VUE_APP_API_URL`.
  - Interceptores de solicitud, que automáticamente añaden el token JWT a las cabeceras.
  - Interceptores de respuesta, que manejan errores como expiración de sesión (401) y redireccionan al login.
  - El envío del token CSRF en caso de ser necesario.
- **Seguridad:** El backend entrega un `token JWT` tras el inicio de sesión. Este se almacena en el `localStorage` del navegador y se incluye en cada petición como cabecera `Authorization: Bearer <token>`. Esto asegura que el acceso a recursos protegidos sea autenticado.
- **Rutas consumidas:** Algunas rutas importantes del backend que el frontend utiliza son:
  - `/api/auth/login/`, `/api/auth/logout/`, `/api/auth/profile/`
  - `/api/students/`, `/api/teachers/`, `/api/courses/`, entre otras.
- **Gestión desde componentes:** Los componentes Vue utilizan el archivo `api.js` para realizar las acciones necesarias. Por ejemplo, el componente `Login.vue` utiliza `api.login()` al enviar el formulario de inicio de sesión.
- **Configuración del entorno local:** En `vue.config.js` se configura un proxy local que redirige peticiones `/api` al dominio del backend en producción:

```
devServer: {
    proxy: {
        '/api': {
            target: 'https://cuna-unsa-production.up.railway.app',
            changeOrigin: true,
            secure: true,
        }
    }
}
```

Resumen: La conexión entre ambas capas es robusta, segura y centralizada. Se asegura una integración fluida mediante uso de tokens, configuración modular y control automático de sesiones expiradas. Todo esto facilita la escalabilidad y el mantenimiento futuro del sistema.

7.4. URLs Públicas del Sistema

A continuación se detallan los enlaces públicos relacionados al desarrollo, despliegue y documentación del sistema:

- **Repositorio del Frontend (Vue):** `https://github.com/lau230595/CunaFrontend`
- **Repositorio del Backend (Django):** `https://github.com/KarlaBedregal/CunaUnsa_`
- **Sitio Web Desplegado (Netlify):** `https://cunaunsa.netlify.app/`
- **API REST Pública (Render):** `https://cunaunsa.onrender.com/api/`
- **Playlist de YouTube del Proyecto:** `https://www.youtube.com/playlist?list=PLxcXnsVAKkkOeRiS--BGBqqsqF3`

8. Trabajo en Equipo

8.1. Distribución de Tareas

- **Bedregal Coaguila, Karla M.** —Desarrollo principal del backend en Django y API REST.
- **Llaique Chullunquia, Jack F.** — Desarrollo principal del frontend en Vue.
- **Ramos Ochochoque, Elkin E.** — Apoyo en documentación y backend.
- **Vilca Huarca, Laura L.** — Participación activa en backend, frontend y deploy de ambos.

8.2. Porcentaje de Participación

Integrante	**Porcentaje**
Bedregal Coaguila, Karla M.	100 %
Llaique Chullunquia, Jack F.	100 %
Ramos Ochochoque, Elkin E.	100 %
Vilca Huarca, Laura L.	100 %

8.3. Rubrica

Cuadro 17: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

Cuadro 18: Evaluación por Criterios

Criterio	**Excelente (3)**	**Bueno (2)**	**Regular (1.5)**	**Deficiente (0.7)**	**Auto**	**Docente**
Producción de Videos	Muy claro, estructurado. Alta calidad.	Claro, buena calidad. Casi completo.	Algo claro. Calidad media.	Confuso o baja calidad. No comunica.	2	
Backend en Producción	Funciona sin errores. Seguridad sólida.	Funciona con leves fallos.	Parcialmente funcional.	No funciona ni desplegado.	2	
Frontend en Producción	Interfaz funcional y atractiva.	Usable con detalles menores.	Interfaz básica, falta funcionalidad.	Confuso o no funcional.	2	
Backup de BD	Completo, verificado y documentado.	Completo pero sin prueba.	Incompleto, manual.	No hay backup.	2	
GitHub Backend	Commits claros. Buen uso de ramas.	Organizado. Mensajes adecuados.	Desordenado. Mensajes vagos.	Caótico. Todo en main.	2	
GitHub Frontend	Estructura clara y ramas activas.	Constante, sin mucho detalle.	Básico y directo a main.	Sin estructura ni control.	2	
Informe Final	Completo, análisis crítico.	Buen análisis, pocos errores.	Cumple, con algunos errores.	Incompleto o con errores graves.	2	
Total (/21)					14	

9. Conclusiones

Referencias

[1] Vue.js, "Vue.js Guide," [Online]. Available: `https://vuejs.org/guide/introduction.html`. [Accessed: 27-Jul-2025].

[2] Vuex, "State Management for Vue.js," [Online]. Available: `https://vuex.vuejs.org/`. [Accessed: 27-Jul-2025].

[3] Axios, "Axios HTTP Client," [Online]. Available: `https://axios-http.com/docs/intro`. [Accessed: 27-Jul-2025].

[4] Vue Router, "Vue Router Official Docs," [Online]. Available: `https://router.vuejs.org/`. [Accessed: 27-Jul-2025].

[5] T. Christie, "Django REST framework," [Online]. Available: `https://www.django-rest-framework.org/`. [Accessed: 27-Jul-2025].

[6] Django Software Foundation, "Django Documentation," [Online]. Available: `https://docs.djangoproject.com/en/stable/`. [Accessed: 27-Jul-2025].

[7] Netlify, "Netlify Documentation," [Online]. Available: `https://docs.netlify.com/`. [Accessed: 27-Jul-2025].

[8] Render, "Render Docs," [Online]. Available: `https://docs.render.com/`. [Accessed: 27-Jul-2025].

[9] Bootstrap, "Bootstrap 5 Documentation," [Online]. Available: `https://getbootstrap.com/docs/5.3/getting-started/introduction/`. [Accessed: 27-Jul-2025].

[10] Auth0, "Introduction to JSON Web Tokens," [Online]. Available: `https://auth0.com/docs/secure/tokens/json-web-tokens`. [Accessed: 27-Jul-2025].

[11] ESLint, "ESLint Documentation," [Online]. Available: `https://eslint.org/docs/latest/`. [Accessed: 27-Jul-2025].

[12] Vue.js Team, "Vue CLI - Standard Tooling for Vue.js Development," [Online]. Available: `https://cli.vuejs.org/`. [Accessed: 27-Jul-2025].