SYSTEM DOCUMENTATION

1. Project Information

Project Name: Course Platform

Name: Kenner De Jesus Palma Rivadeneira

Course: Systems Engineering - Database II

Semester:2025-2 Date:20/11/2025

Instructor: Jaider Jose Quintero Mendoza

Short Project Description:

A full-stack e-learning platform for managing and delivering online courses. The system allows instructors to create and manage courses organized into modules and lessons, manage assessments (exams, quizzes, assignments), receive student submissions, and facilitate interaction through discussion forums. Students can enroll in courses, complete lessons, take assessments, submit assignments, and participate in community forums. The system includes progress tracking, grading, certificates, and a tagging system for course categorization.

2. System Architecture Overview

2.1 Architecture Description

Our system uses a three-layer architecture to keep everything organized and easy to scale. Each layer has a clear role, like a well-coordinated team. Here's a quick summary.

Presentation Layer (Frontend)

- **Angular 20.3.0**: Dynamic and user-friendly interface.
- Reusable components for entities like courses or students.
- Services to chat with the backend, using RxJS for real-time data without lags.

Business Logic Layer (Backend)

- **REST API with Django REST Framework**: Handles the smarts.
- ViewSets for quick CRUD and serializers for data validation.
- Modular apps (users, courses, etc.) to avoid any tangles.

Data Layer (Database)

- Support for MySQL, PostgreSQL, etc.

- Django ORM for easy queries, no raw SQL needed.
- Migrations to update the schema without headaches.

Comunicación:

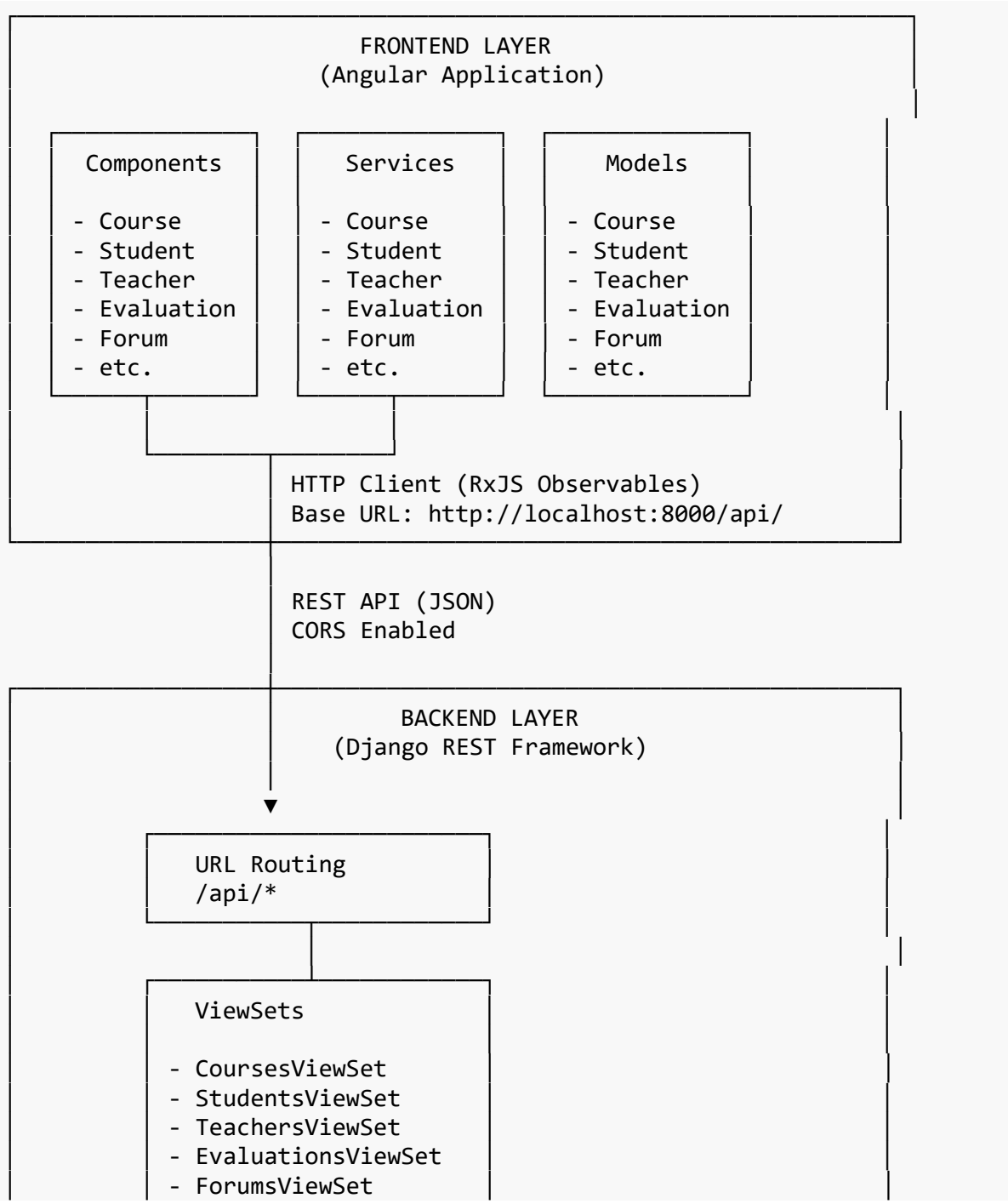
- Protocolo HTTP/HTTPS mediante API REST
- CORS habilitado para comunicación entre frontend (puerto 4200) y backend (puerto 8000).
- Formato JSON para intercambio de datos.
- Paginación implementada.

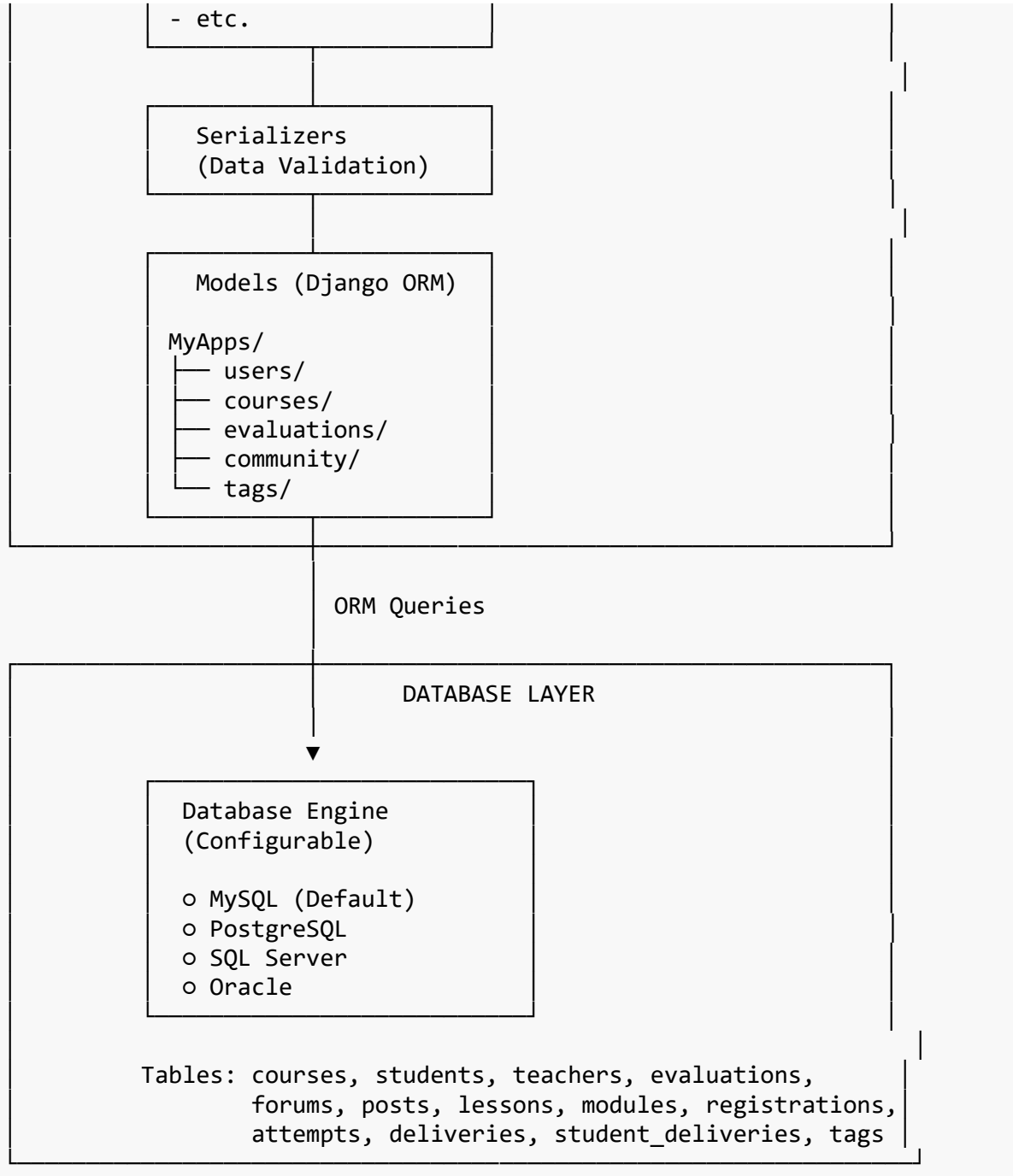
2.2 Technologies Used

- **Frontend:**
    - Angular 20.3.0 (main framework)
    - TypeScript 5.9.2
    - PrimeNG 20.3.0 (UI components)
    - TailwindCSS 4.1.17 (utility CSS framework)
    - Angular Router & Forms
    - Angular HTTP Client
- **Backend:**
    - Django 5.2.7
    - Django REST Framework 3.16.1
    - Python 3.
    - django-cors-headers 4.9.0
    - python-decouple 3.8
- **Database Engine:**
    - MySQL (default)
    - PostgreSQL
    - SQL Server
    - Oracle

- **Additional Libraries / Tools:**
    - Angular CLI 20.3.10
    - Prettier (frontend formatting)
    - Black (Python code formatting)

2.3 Visual explanation of the system's operation





3. Database Documentation (ENGLISH)

3.1 Database Description

Relational database for an e-learning platform with 13 modules in 5 areas:

User Management: teachers, students
Course Management: courses, modules, lessons, tags, registrations
Evaluations: evaluations, attempts, submissions, student_submissions

Community: forums, posts
Multi-database support (MySQL, PostgreSQL, SQL Server, Oracle)

3.2 ERD – Entity Relationship Diagram

3.3 Logical Model

Core Entities:

TEACHERS (id, name, last_name, email, phone, specialty, biography, photo_url)

STUDENTS (id, name, last_name, email, phone, state, photo_url)

COURSES (id, teacher_id, title, description, objetives, creation_date, start_date, end_date, state_course, requeriments, language, price, max_students, level, image_url, certificate_available, estimated_duration)

MODULES (id, course_id, name, description, order, estimated_duration, available_date)

LESSONS (id, module_id, name, content, content_type, available_date, order, resource_url)

EVALUATIONS (id, lesson_id, title, description, type, maximun_score, limit_date, available_from, available_until, show_results, minimum_score, attempts_allowed)

ATTEMPTS (id, evaluation_id, student_id, attempt_date, attempt_number, start_time, end_time, time_spent, state, maximun_score, score_obtained, answers)

DELIVERIES (id, lesson_id, title, description, limit_date)

STUDENT_DELIVERIES (id, delivery_id, student_id, delivery_date, archive, calification, state)

REGISTRATIONS (id, student_id, course_id, registration_date, state, final_grade, certificate_issued, progress)

FORUMS (id, course_id, title, description, type, date, is_moderated, is_active)

POSTS (id, forum_id, student_id, teacher_id, post_father_id, content, publications_date, attachment_url, is_question, is_accepted_answer, upvotes, downvotes, is_moderated, published_at, edited_at)

TAGS (id, name)

Junction Tables: courses_tags (many-to-many), registrations, attempts, student_deliveries

3.4 Physical Model (Tables)

Table	Column	Type	PK/FK	Description
teachers	id	INTEGER	PK	Primary key (auto-

Table	Column	Type	PK/FK	Description
				increment)
	name	VARCHAR(50)		Teacher’s first name
	last_name	VARCHAR(50)		Teacher’s last name
	email	VARCHAR(100)		Teacher’s email address
	phone	VARCHAR(20)		Teacher’s phone number (nullable)
	specialty	VARCHAR(100)		Teacher’s specialty area
	biography	TEXT		Teacher’s biography
	photo_url	VARCHAR(250)		URL to teacher’s photo (nullable)
students	id	INTEGER	PK	Primary key (auto-increment)
	name	VARCHAR(50)		Student’s first name
	last_name	VARCHAR(50)		Student’s last name
	email	VARCHAR(30)		Student’s email address
	phone	VARCHAR(20)		Student’s phone number (nullable)
	state	VARCHAR(10)		Student state: ACTIVO/INACTIVO/SUSPENDIDO
	photo_url	VARCHAR(250)		URL to student’s photo (nullable)
courses	id	INTEGER	PK	Primary key (auto-increment)
	teacher_id	INTEGER	FK → teachers(id)	Foreign key to teachers table

Table	Column	Type	PK/FK	Description
	title	VARCHAR(200)		Course title
	description	TEXT		Course description
	objetives	TEXT		Course objectives
	creation_date	DATE		Course creation date (auto)
	start_date	DATE		Course start date
	end_date	DATE		Course end date
	state_course	VARCHAR(100)		Course state: activo/inactivo/archivado
	requeriments	VARCHAR(50)		Course requirements (nullable)
	language	VARCHAR(15)		Course language
	price	DECIMAL(10,2)		Course price (default: 0.00)
	max_students	INTEGER		Maximum number of students (nullable)
	level	VARCHAR(15)		Course level: básico/intermedio/avanzado
	image_url	VARCHAR(255)		Course image URL (nullable)
	certificate_available	BOOLEAN		Whether certificate is available (default: false)
	estimated_duration	INTEGER		Estimated duration in hours/days (nullable)
tags	id	INTEGER	PK	Primary key (auto-increment)

Table	Column	Type	PK/FK	Description
	name	VARCHAR(30)		Tag name
courses_tags	id	INTEGER	PK	Primary key (auto-increment)
	course_id	INTEGER	FK → courses(id)	Foreign key to courses table
	tag_id	INTEGER	FK → tags(id)	Foreign key to tags table
modules	id	INTEGER	PK	Primary key (auto-increment)
	course_id	INTEGER	FK → courses(id)	Foreign key to courses table (nullable)
	name	VARCHAR(50)		Module name (nullable)
	description	TEXT		Module description (nullable)
	order	VARCHAR(100)		Module order/sequence
	estimated_duration	INTEGER		Estimated duration (nullable)
	available_date	DATE		Module availability date (nullable)
lessons	id	INTEGER	PK	Primary key (auto-increment)
	module_id	INTEGER	FK → modules(id)	Foreign key to modules table
	name	VARCHAR(100)		Lesson name
	content	TEXT		Lesson content
	content_type	VARCHAR(14)		Content type: Texto/Video/PDF/Enlace externo/Quiz/Documento

Table	Column	Type	PK/FK	Description
	available_date	DATE		Lesson availability date (nullable)
	order	VARCHAR(50)		Lesson order within module
	resource_url	VARCHAR(255)		Resource URL (nullable)
registrations	id	INTEGER	PK	Primary key (auto-increment)
	student_id	INTEGER	FK → students(id)	Foreign key to students table
	course_id	INTEGER	FK → courses(id)	Foreign key to courses table
	registration_date	DATETIME		Registration date and time
	state	VARCHAR(10)		Registration state: active/completed/cancelled
	final_grade	DECIMAL(5,2)		Final grade (nullable)
	certificate_issued	BOOLEAN		Whether certificate was issued (nullable)
	progress	DECIMAL(5,2)		Student progress percentage (nullable)
evaluations	id	INTEGER	PK	Primary key (auto-increment)
	lesson_id	INTEGER	FK → lessons(id)	Foreign key to lessons table
	title	VARCHAR(30)		Evaluation title
	description	TEXT		Evaluation description
	type	VARCHAR(10)		Evaluation type: Exam/Quiz/Assignment

Table	Column	Type	PK/FK	Description
	maximun_score	INTEGER		Maximum possible score (nullable)
	limit_date	DATETIME		Submission deadline (nullable)
	available_from	DATETIME		Available from date (nullable)
	available_until	DATETIME		Available until date (nullable)
	show_results	BOOLEAN		Show results immediately (nullable)
	minimum_score	DECIMAL(5,2)		Minimum passing score (nullable)
	attempts_allowed	INTEGER		Maximum attempts allowed (nullable)
attempts	id	INTEGER	PK	Primary key (auto-increment)
	evaluation_id	INTEGER	FK → evaluations(id)	Foreign key to evaluations table (nullable)
	student_id	INTEGER	FK → students(id)	Foreign key to students table (nullable)
	attempt_date	DATETIME		Attempt date and time
	attempt_number	INTEGER		Attempt number (nullable)
	start_time	DATETIME		Start time (nullable)
	end_time	DATETIME		End time (nullable)
	time_spent	INTEGER		Time spent in minutes

Table	Column	Type	PK/FK	Description
				(nullable)
	state	VARCHAR(11)		Attempt state: EN PROGRESO/COMPLETADO/CALIFICADO
	maximun_score	INTEGER		Maximum score for attempt (nullable)
	score_obtained	DECIMAL(5,0)		Score obtained (nullable)
	answers	JSON		Student answers in JSON format (nullable)
deliveries	id	INTEGER	PK	Primary key (auto-increment)
	lesson_id	INTEGER	FK → lessons(id)	Foreign key to lessons table
	title	TEXT		Delivery title
	description	TEXT		Delivery description
	limit_date	DATETIME		Submission deadline (nullable)
student_deliveries	id	INTEGER	PK	Primary key (auto-increment)
	delivery_id	INTEGER	FK → deliveries(id)	Foreign key to deliveries table (nullable)
	student_id	INTEGER	FK → students(id)	Foreign key to students table (nullable)
	delivery_date	DATETIME		Delivery submission date
	archive	VARCHAR(255)		File URL or path
	calification	FLOAT		Grade received
	state	VARCHAR(20)		Delivery state:

Table	Column	Type	PK/FK	Description
				CALIFICADO/PENDIENTE/DEVUELTA
forums	id	INTEGER	PK	Primary key (auto-increment)
	course_id	INTEGER	FK → courses(id)	Foreign key to courses table (nullable)
	title	VARCHAR(200)		Forum title
	description	TEXT		Forum description
	type	VARCHAR(9)		Forum type: GENERAL/PREGUNTA/PROJECTOS/SOCIAL (nullable)
	date	DATETIME		Forum creation date (auto)
	is_moderated	BOOLEAN		Whether posts need moderation (default: false)
	is_active	BOOLEAN		Whether forum is active (default: false)
posts	id	INTEGER	PK	Primary key (auto-increment)
	forum_id	INTEGER	FK → forums(id)	Foreign key to forums table (nullable)
	student_id	INTEGER	FK → students(id)	Foreign key to students table (nullable)
	teacher_id	INTEGER	FK → teachers(id)	Foreign key to teachers table (nullable)
	post_father_id	INTEGER	FK →	Self-referential

Table	Column	Type	PK/FK	Description
			posts(id)	FK for threaded replies (nullable)
	content	TEXT		Post content
	publications_date	DATETIME		Publication date (auto)
	attachment_url	VARCHAR(255)		Attachment file URL (nullable)
	is_question	BOOLEAN		Whether post is a question (default: false)
	is_accepted_answer	BOOLEAN		Whether answer is accepted (default: false)
	upvotes	INTEGER		Number of upvotes (default: 0)
	downvotes	INTEGER		Number of downvotes (default: 0)
	is_moderated	BOOLEAN		Whether post needs approval (default: false)
	published_at	DATETIME		Publication timestamp (auto, nullable)
	edited_at	DATETIME		Last edit timestamp (auto, nullable)

4. Use Cases – CRUD

4.1 Use Case: Create Course

Actor: Teacher

Description: Create a new course with basic information, schedule, pricing, and associate it with tags

Preconditions: Teacher exists in the system

Postconditions: New course is created and stored in database

Main Flow:

1. Teacher accesses course creation form
1. System displays form with required fields
1. Teacher enters course data (title, description, dates, price, etc.)
1. Teacher selects associated tags (optional)
1. System validates data
1. System creates course record
1. System returns confirmation

4.2 Use Case: Read Course

Actor: Student, Teacher

Description: Retrieve and view course information

Preconditions: Course exists in the system

Postconditions: Course details are displayed

Main Flow:

1. User requests course list or specific course
1. System queries database
1. System returns course data with related information
1. User views course details

4.3 Use Case: Update Course

Actor: Teacher

Description: Modify existing course information

Preconditions: Course exists and user is the assigned teacher

Postconditions: Course information is updated in database

Main Flow:

1. Teacher accesses course update form
1. System loads current course data
1. Teacher modifies fields
1. System validates changes
1. System updates course record
1. System returns confirmation

4.4 Use Case: Delete Course

Actor: Teacher

Description: Remove a course from the system

Preconditions: Course exists and user is the assigned teacher

Postconditions: Course and related records (modules, lessons, etc.) are deleted (CASCADE)

Main Flow:

1. Teacher selects course to delete
1. System confirms deletion request
1. Teacher confirms
1. System deletes course (CASCADE delete removes related records)
1. System returns confirmation

5. Backend Documentation

5.1 Backend Architecture

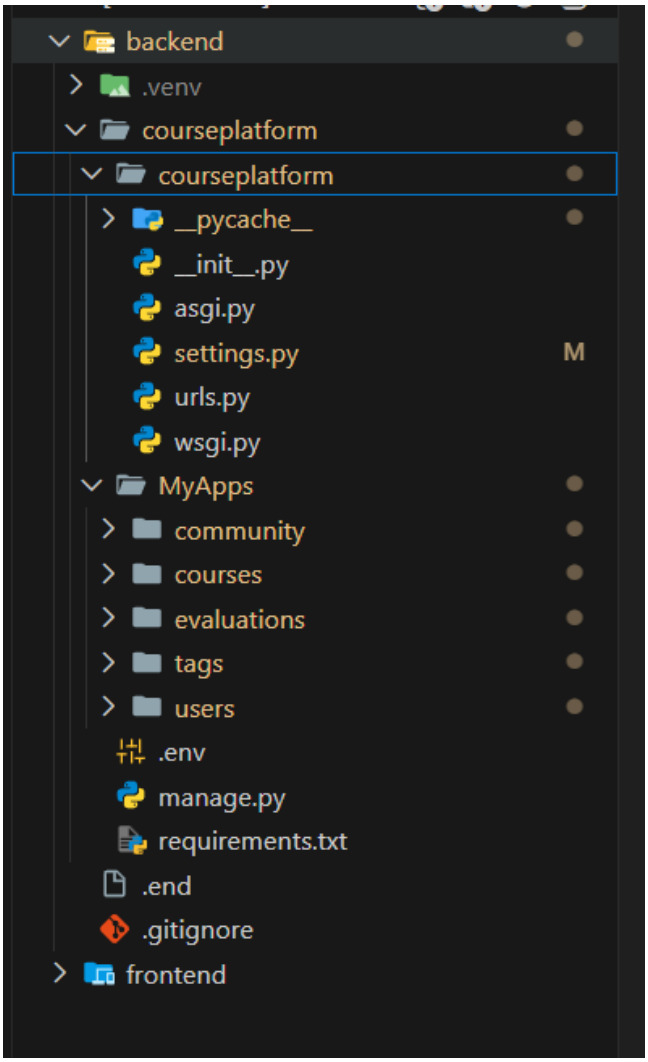
The backend follows Django REST Framework architecture with a modular design:

- Framework: Django 5.2.7 with Django REST Framework 3.16.1
- Architecture pattern: Model-View-Serializer (MVS) pattern
- API style: RESTful API using ViewSets
- Routing: DefaultRouter for automatic CRUD endpoint generation
- Database abstraction: Django ORM with multi-database support
- CORS: Enabled for cross-origin requests from Angular frontend

Components:

- Models: Django ORM models representing database entities
- Serializers: Data validation and serialization layer
- ViewSets: RESTful view logic using ModelViewSet
- Routers: URL routing configuration for API endpoints
- Settings: Centralized configuration with environment variables

5.2 Folder Structure



5.3 API Documentation (REST)

Courses API

Method Path: GET /api/courses/

Purpose: Retrieve all courses with pagination

Request Body Example: None

Responses:

- 200 OK - Returns paginated list of courses
- 400 Bad Request - Invalid request parameters

Request Body Example:

```
{
  "count": 10,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": 1,
      "title": "Introduction to Python",
      "description": "Learn Python basics",
      "teacher": 1,
      "price": "99.99",
      "state_course": "activo"
    }
  ]
}
```


5.4 REST Client

The system was tested using:

Tool:

- vsCode extension called REST CLIENT
- The following was tested:
- CRUD endpoints for each module.
- Error handling.
- Serialization and relationships between models.

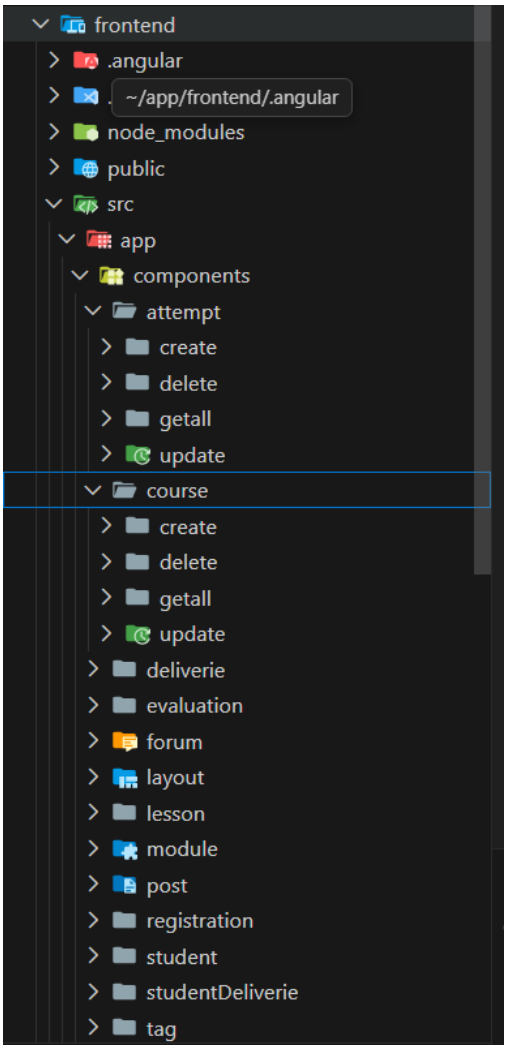
This is important because:

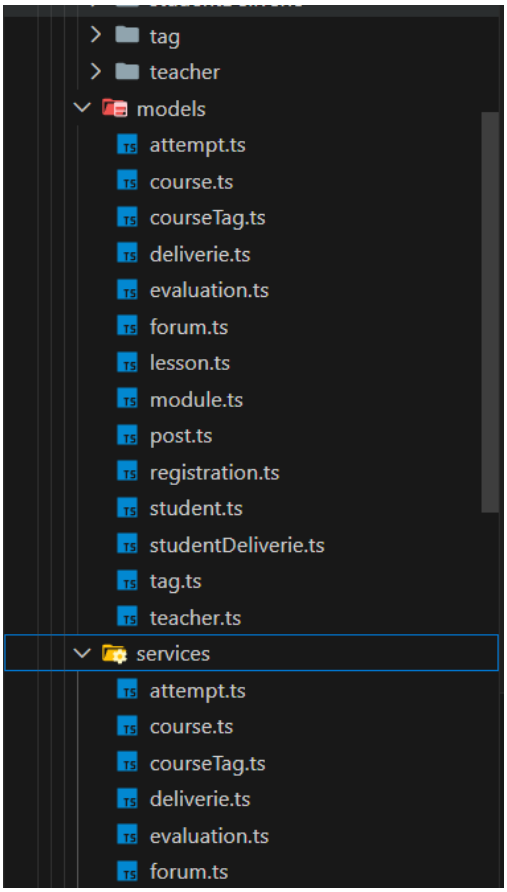
- It validates backend responses before connecting Angular.
- It ensures that request formats match expectations.

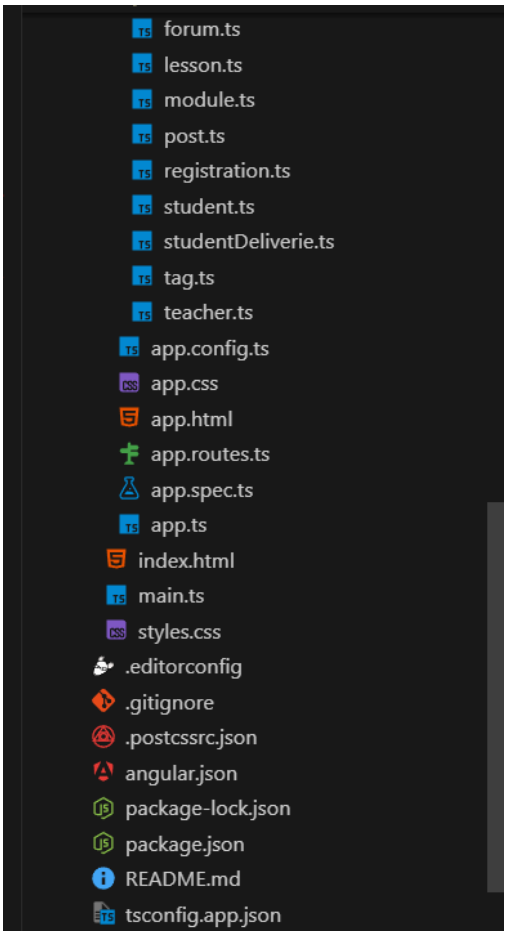
6. Frontend Documentation

6.1 Technical Frontend Documentation

Framework Used: Angular 20.3
Folder Structure:

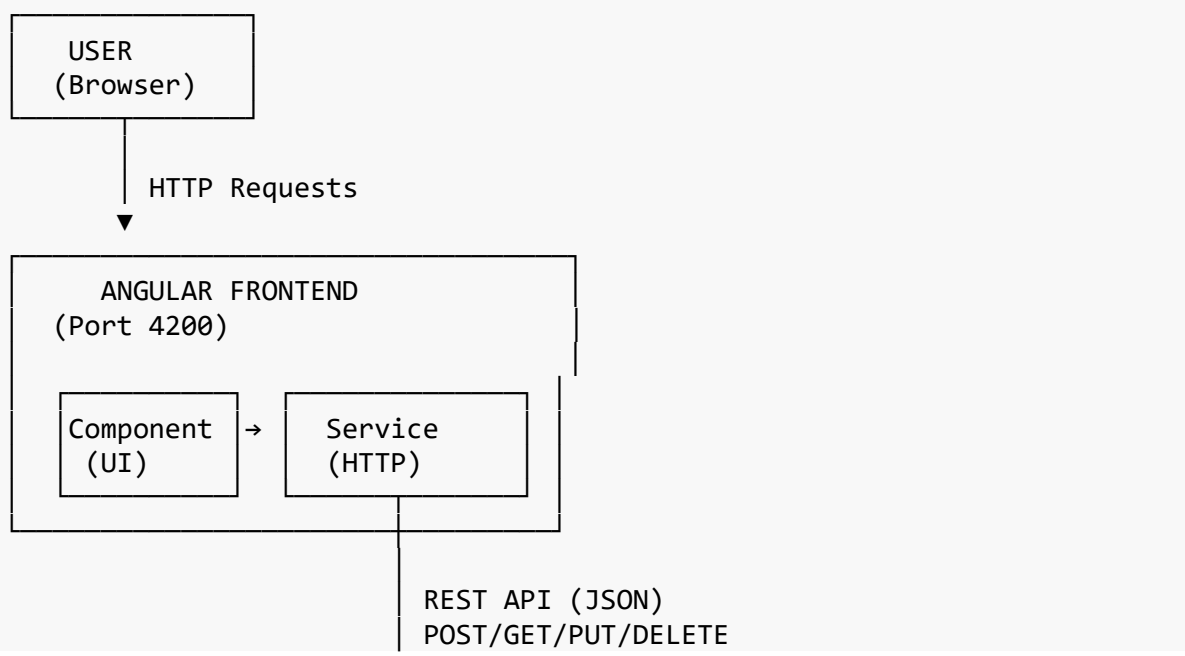

frontend
.angular
~/app/frontend/.angular
node_modules
public
src
app
components
attempt
create
delete
getall
update
course
create
delete
getall
update
deliverie
evaluation
forum
layout
lesson
module
post
registration
student
studentDeliverie
tag

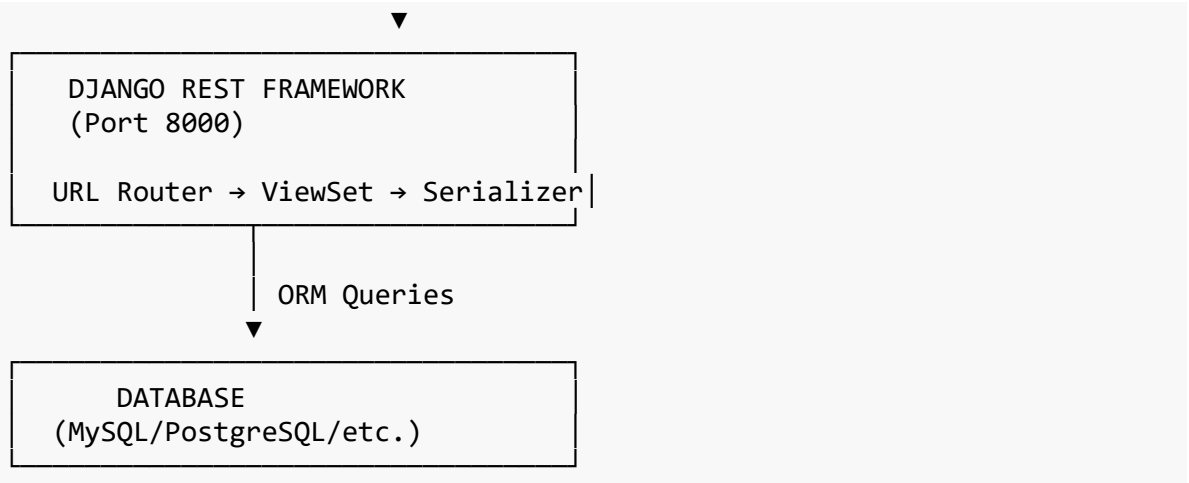

tag
teacher
models
attempt.ts
course.ts
courseTag.ts
deliverie.ts
evaluation.ts
forum.ts
lesson.ts
module.ts
post.ts
registration.ts
student.ts
studentDeliverie.ts
tag.ts
teacher.ts
services
attempt.ts
course.ts
courseTag.ts
deliverie.ts
evaluation.ts
forum.ts



Models, services and Components

6.2 Visual explanation of the system's operation





Data Flow:

User interaction in Angular component

Component calls service method

Service sends HTTP request to Django API

Django ViewSet processes request

Serializer validates/transforms data

ORM queries database

Response returns through layers

Component updates UI

7. Frontend–Backend Integration

Integration Architecture

- Communication protocol: HTTP REST API
- Data format: JSON
- CORS: Enabled for cross-origin requests
- Frontend base URL: http://localhost:4200
- Backend base URL: http://localhost:8000/api/

Integration Pattern

Angular services encapsulate API communication:

Service Layer → HTTP Client → Django REST API

Example Integration Flow:

Angular Component calls service method

this.courseService.getAllCourses()

Service makes HTTP request

http.get('http://localhost:8000/api/courses/')

Django REST Framework processes request

- URL Router → ViewSet → Serializer → Model

Response returns to Angular

- Observable stream → Component updates UI

8. Conclusions & Recommendations

Conclusions

- Full-stack e-learning platform implemented with Angular frontend and Django REST backend
- REST API provides complete CRUD operations for 13 entities
- Modular architecture allows independent development and maintenance
- Multi-database support enables flexibility in deployment
- Clear separation between frontend and backend facilitates scalability

9. Annexes (Optional)