# University System (Student)

## Description:

The University System is an integrated platform designed for student use, providing essential functionalities such as enrollment in courses, access to information about subjects, generation of subjects reports and facilitating communication through email. It consists of a Student App, API backend, and databases for subjects, enrollments, and reports.

## Features:

### Subject Selection

User Story:

As a student, I want to be able to select subjects easily through the Student App.

Breakdown:

- Implement a user-friendly interface for subject selection.
- Integrate the Subject Provider component to fetch and display available subjects.
- Ensure secure communication between the Student App and the API.

Responsibilities:

- Front-end Developer: Implement the user interface.
- Back-end Developer: Integrate the Subject Provider component and handle API requests.
- Security Engineer: Ensure secure communication channels.

### Enrollment Management

User Story:

As a student, I want to manage my course enrollments efficiently.

Breakdown:

- Identification of available courses.
- Implement a feature to add or remove enrollments.
- Ensure data consistency in the database.
- Enrollment confirmation

Responsibilities:

- Front-end Developer: Implement the enrollment management interface.
- Back-end Developer: Integrate the Enrollment Handler component and manage database transactions.
- Database Administrator: Ensure data consistency and integrity.

### Report Generation

User Story:

As a student, I want to generate reports based on my academic performance.

Breakdown:

- Design a user interface for report generation.
- Implement the Report Provider component to handle report requests.
- Validate and format generated reports before persisting.

Responsibilities:

- Front-end Developer: Design the report generation interface.
- Back-end Developer: Integrate the Report Provider component and manage report generation logic.
- Quality Assurance: Validate the accuracy and format of generated reports.

## C4 Model Architecture

### Level 1: System context

In this level we will explore the 4 entities that interact with our system to define the functionality and improve the user experience.

External Entities:

- Student Users: These are the end users of the system. They interact with the system to enroll in courses, access subject information, generate reports, and submit opinions on subjects.
- Email System (Sendgrid): Facilitates email communication for various system notifications, including enrollment confirmations and report generation alerts.
- Api Services: Is a cloud platform that offers a variety of services and resources for developing and managing online applications. The system provides different services such as, hosting, static apps, app services, and Azure SQL databases. These services guarantee scalability, reliability, and efficient data management.
- Browser: Provides the user interface for students to interact with the system. Where students can perform a variety of actions, such as enrolling in courses, accessing course information, generating academic reports and submitting feedback on courses.

### Level 2: Containers

The four containers are represented in this level. These containers encapsulate different facets of the system in an efficient and organized manner. Segmenting our system this way, we logic boundaries that simplify design, enable independent scalability and facilitate maintenance.
It should be mentioned that our system uses two databases. This way we make sure that different responsibilities are separated, scalability, security and we make the system stronger facing failures.
In this section, we will explore each of these containers.

- Container: Student App (subjectContainer)

The Student App container represents the user interface layer of the system. It is responsible for providing students with a seamless and intuitive interaction platform. This container encompasses functionalities related to subject management, report generation, and the submission of opinions.

- Container: Subjects + Reports database (DB Container)

The Subjects + Reports container is the data storage layer dedicated to managing information related to subjects, academic reports, and student opinions. This container ensures the persistent storage and retrieval of data to support the dynamic aspects of the application.

- Container: Enrollment + Student database (DB Container2)

The Enrollment + Student container handles the storage and retrieval of data pertinent to student enrollments and general student information. This container is crucial for maintaining accurate records of student academic activities.

- Container: Backend API (apiContainer)

The Backend API container serves as the bridge between the user interface and the data storage layers. It processes requests from the Student App, orchestrates the necessary actions, and communicates with the database containers. This container encapsulates the application's business logic, ensuring the seamless flow of information between the frontend and backend.

## Level 3: Components

### Web Layer

- API Controller (controllerComp): This component manages the flow of HTTP requests and responses, serving as the interface between the frontend and the backend. It handles routing, serialization, and overall HTTP communication.

### Application Layer

- Subject Provider (subjectProvider): Responsible for handling business logic related to subject management. It processes requests for subject information, filters subjects based on characteristics, and ensures a smooth flow of data between the frontend and the database.
- Enrollment Handler (enrollmentHandler): Manages business logic for student enrollments. It processes requests for enrolling in courses, updates the database with enrollment information, and handles related actions.
- Report Handler (reportProvider): Manages the creation and validation of academic reports. It processes requests for generating reports, interacts with the database to retrieve relevant data, and ensures the accurate representation of academic information.
- Opinion Handler (opinionProvider): This component is responsible for handling the submission and retrieval of opinions or reports on subjects. It ensures that students can express their feedback on each subject they have attended.

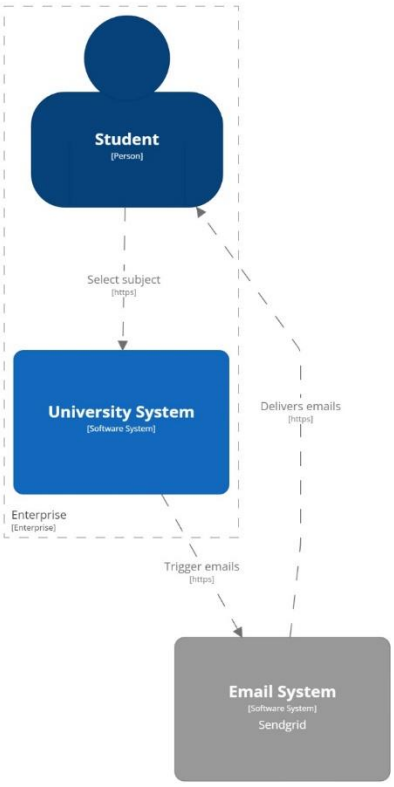
### Infrastructure Layer

- DBContext (dbContextComp): Acts as the Object-Relational Mapping (ORM) component, translating Linq queries to interactions with the database. It plays a crucial role in data storage, retrieval, and modification using Entity Framework Core.

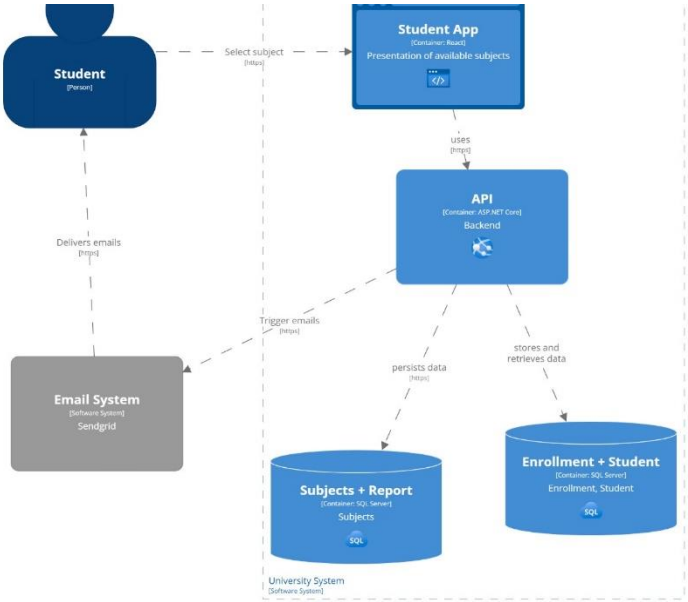
### Domain Layer

- Report (reportDomain): Represents the domain model for academic reports. It defines the structure and properties of reports, ensuring consistency in data representation.
- Subject (subjectDomain): The domain model for subjects, outlining their characteristics, prerequisites, and other relevant information.
- Enroll (enrollDomain): Represents the domain model for student enrollments, capturing essential details about course enrollments.
- Opinion (opinionDomain): Domain model for student opinions on subjects. It defines the structure for storing and retrieving feedback.

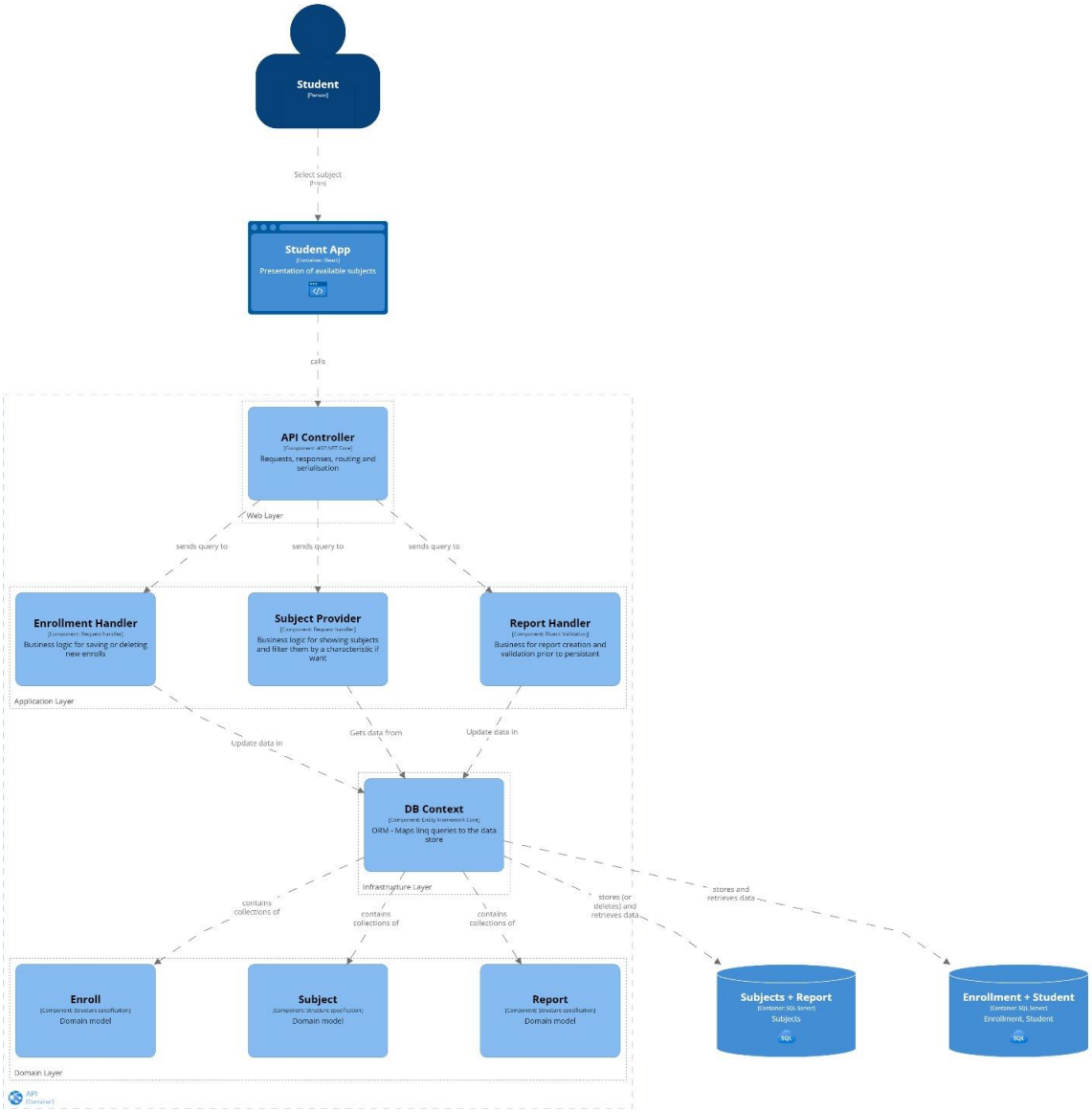
## System Architecture




[System Context] University System
jueves, 23 de noviembre de 2023, 11:51 hora estándar de Europa central




[Container] University System
jueves, 23 de noviembre de 2023, 11:50 hora estándar de Europa central




[Component] University System - API

jueves, 23 de noviembre de 2025, 17:31 hora estándar de Europa central