1.IT Gymnázium

Attendance API Documentation

Jiří Drbohlav
Praha 2024

1. Content

2. Introduction

This API is designed to help schools manage various administrative tasks digitally. It provides a range of features for managing schedules, subjects, classes, and handling student absences. This API serves as a central tool for administrators and teachers, streamlining complex school operations and ensuring easy access and management of school activities.
This documentation is meant for people working on or with this software.

3. Parts Of Software

3.1 API

The main part of this software is the API. The API is written in .NET 6 and is deployed on a Raspberry Pi. The API stores its data in a PostgreSQL database

3.2 Raspberry Pi Reader

The second part of this software is the Raspberry Pi used to log students absences using the APIs endpoint absence/write. The Raspberry Pi contains an RFID reader which is used to scan ISIC cards and send its information to the API which handles it.

3.3 Database

This software uses the PostgreSql database system. The SQL database is deployed on a Raspberry Pi as well as the API.

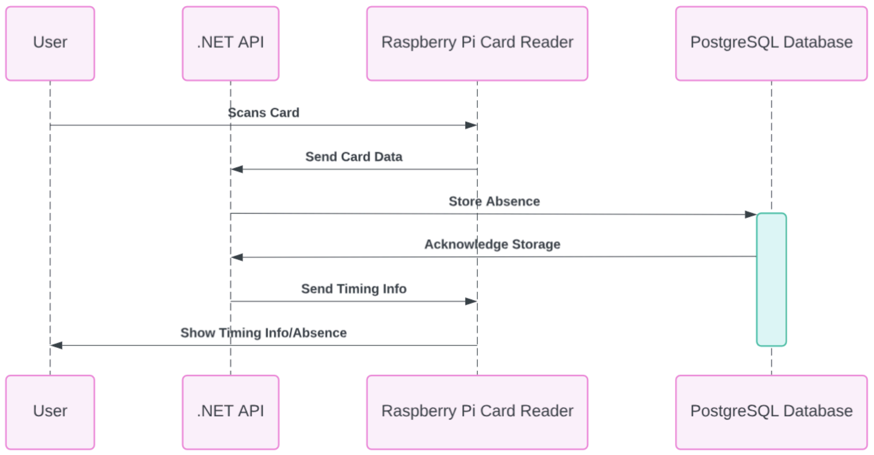
3.4 Deploy environment

The API and database is deployed on a Raspberry Pi that is using the Raspbian OS.

4. Architecture

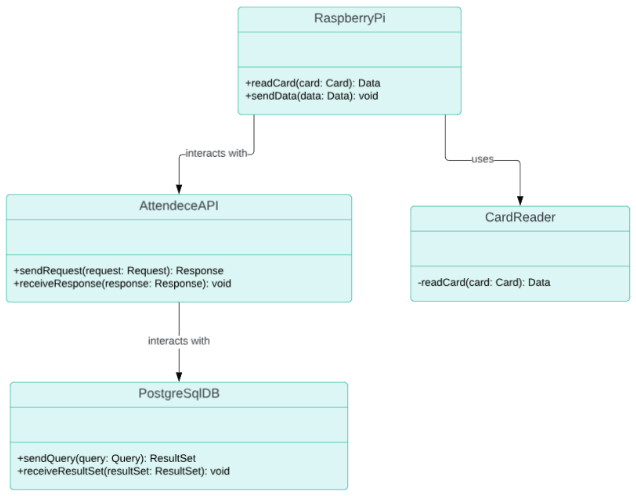
4.1 Scanning card diagram

User scans the card on the Raspberry Pi, the data gets processed by the API and is stored to the database.



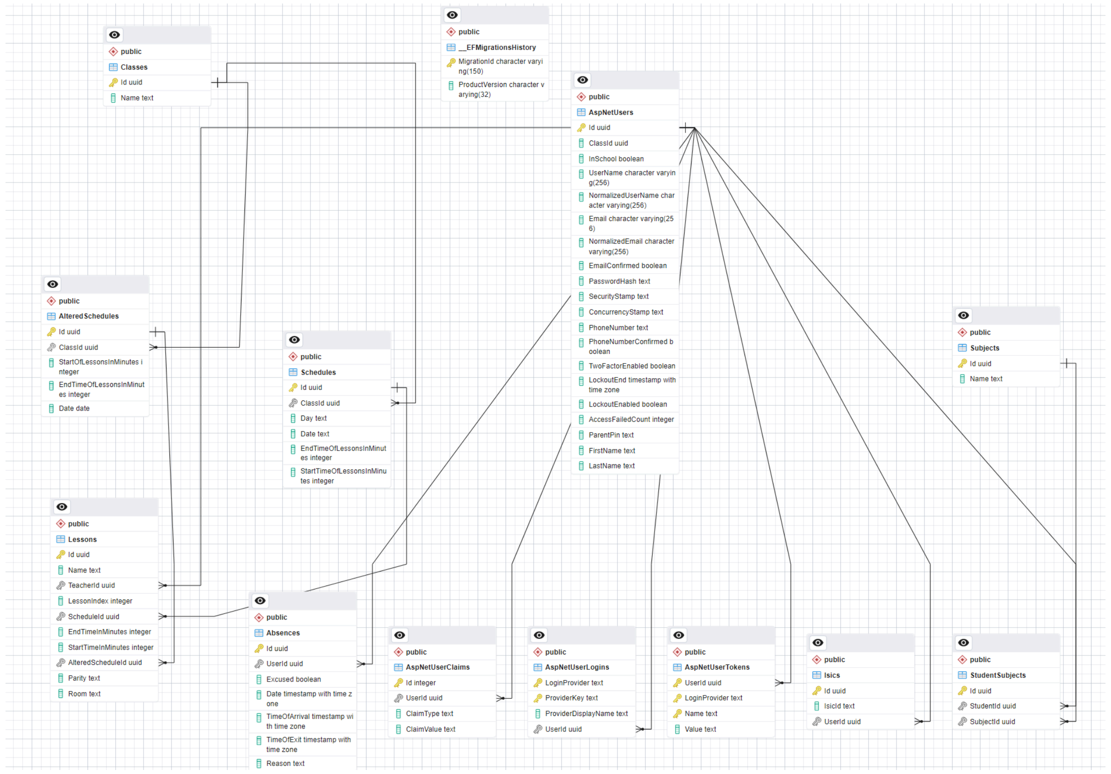
4.2 Software communication diagram

Diagram of communication between the Raspberry Pi card reader, API and the database.


RaspberryPi
+readCard(card: Card): Data
+sendData(data: Data): void
interacts with
uses
AttendeceAPI
+sendRequest(request: Request): Response
+receiveResponse(response: Response): void
CardReader
-readCard(card: Card): Data
interacts with
PostgreSqlDB
+sendQuery(query: Query): ResultSet
+receiveResultSet(resultSet: ResultSet): void

4.3 ERD

ERD diagram of the Postgre database.



5. Authentication and Authorization

5.1 Authorization

This API integrates with the school's own LDAP server to manage user authorization. When a user attempts to log in, the API sends the user's credentials (username and password) to the LDAP server. The LDAP server checks if an account with those credentials exists and if the provided password matches the stored password. If the credentials are verified, the user is granted access, and user information is retrieved and used in subsequent authorization checks.

5.2 Authentication

The API uses cookie-based authentication to manage sessions after users are successfully logged in. Here's how cookie authentication is handled:

Cookie Creation: Once the LDAP server authenticates the user, the API generates a session cookie. This cookie contains an encrypted token that securely identifies the user. The cookie is then sent to the user's browser, which stores it for subsequent requests.

Session Management: With each request to the API, the user's browser sends the cookie back to the server. The API decrypts the cookie to retrieve the user's session token, which is then used to maintain session state and user context for the duration of the session.

Expiration and Logout: Cookies have a set expiration time, after which they are automatically invalidated. In this case the log out time is 12 hours. Users can also manually log out, which triggers the API to invalidate the session cookie immediately, effectively ending the user's session.

6. Endpoint Documentation

API's endpoint documentation is hosted and managed on SwaggerHub, an integrated API design and documentation platform that ensures our API is described accurately and remains easily accessible to developers. Using SwaggerHub provides several advantages in maintaining our API documentation.
API documentation is publicly available on SwaggerHub at the following URL:
https://app.swaggerhub.com/apis/JJIRIK66_1/attendence-apiv2/1.0

7. Database

The database contains a total amount of 13 tables here is a brief description of these tables

7.1 AspNetUsers

This table contains information about the users of the system. It includes user IDs, ID of class the user is in, in-school status, personal information like names and email addresses, and security-related fields such as password hashes and security stamps.

7.2 AspNetUserClaims

Stores claims for users which are used for identity management and can store a variety of attributes relevant to authorization.

7.3 AspNetUserTokens

Used for storing tokens for authentication purposes, such as reset password tokens or email confirmation tokens. It isn't really used because the API doesn't use token based authentication.

7.4 Classes

Represents the classes within the school. Each class entry has an ID and a name.

7.5 Subjects

Contains the list of subjects taught in the school, with each subject having an ID and a name.

7.6 StudentSubjects

This is a junction table that maps students to the subjects they are attending, allowing for a many-to-many relationship between AspNetUsers (students) and Subjects.

7.7 Absences

Keeps records of student absences. Includes details like the date, times of arrival and exit, whether the absence is excused, and the reason for the absence.

7.8 Schedules

Stores information on class schedules, including class IDs, days, dates, and times. This table is essential for managing the base schedules of each class.

7.9 AlteredSchedules

Stores alternate schedules. If the schedule of the week is different from the base schedule stored in Schedules, it is stored here.

7.10 Lessons

Detailed entries of individual lessons, including lesson names, parity, associated teachers, and rooms.

8. Packages

8.1 Identity

ASP.NET Core Identity is a framework that enables applications to manage users, passwords, profile data, roles, claims, tokens, email confirmation, and more. It is used for implementing claims, authentication and authorization functionalities. This project uses the version 6.0.4.

8.2 Entity Framework Core

Entity Framework Core, is an open-source data access technology. EF Core serves as an object-relational mapper (ORM), enabling .NET to work with a database using .NET objects, thus eliminating the need for most of the data-access code that developers usually need to write. This project uses the version 6.0.4.

8.3 Swashbuckle

Swashbuckle is an open-source technology for generating Swagger documents for ASP.NET Core web applications. It seamlessly integrates with the ASP.NET Core API framework to provide automated documentation, interactive UIs, and client SDK generation.
Swashbuckle automatically generates Swagger JSON documentation for all endpoints in API, detailing available operations, their parameters, and response models.This project uses the version 6.3.0.

9. Source Code

The project's source code is available on GitHub. Access the source code at this GitHub repository: https://github.com/1-IT-Gymnazium/AttendenceAPI-BE