**ENERGETIKAI TECHNIKUM ÉS KOLLÉGIUM**

**Final Exam Project**

**Consumption Management System**

**2025.**

|  |  |
| --- | --- |
| **Profession:** | **Prepared By:** |
| **SOFTWARE DEVELOPER AND -TESTER** | **Dobosi Gábor** |
| **5 0613 12 03** | **Mák Luca** |
|  | **Orbán Barnabás** |

# ****Overview****

The purpose of our system is to simplify and enhance the efficiency of administrative processes in spa complexes. The program manages and stores guest data and consumption in real time, enabling fast and accurate record-keeping. The application supports guest registration using a unique identifier, tracking of arrival and departure times, as well as detailed logging of the services and consumptions used by the guests.

The system is capable of recording and invoicing the fees of various services (e.g., ticket purchases, wellness treatments, dining, and other supplementary services) in real time. It provides real-time feedback on the user’s entitlements and consumption status.

For the complex’s staff, the system offers a simple and transparent user interface that allows for the quick and accurate completion of administrative tasks.

# ****Requirements****

As a first step, we reviewed the exam requirements to identify the mandatory elements. The application or website was required to include a frontend component responsible for displaying and loading data. Additionally, a backend system was necessary to handle data storage, permission management, the precise definition of database tables and relationships, and the implementation of API endpoints, enabling communication between the frontend and backend.

For efficiency, we used the React framework on the frontend side. For backend development, we chose the ASP.NET Core Web API technology, which is based on the C# programming language.

The software uses an MSSQL-based database, which is easily manageable through the Entity Framework. Our application supports basic data management operations (CRUD), including data creation (Create), retrieval (Read), updating (Update), and deletion (Delete). These operations were implemented on the backend using the HTTP methods GET, POST, PUT, and DELETE.

We aimed to follow clean code principles to enhance development efficiency, maintain code clarity, and simplify future modifications. To support this, for example, we defined clear variable names following the PascalCase naming convention and created a logical, well-structured folder hierarchy.