# # Timelogger, e-conomic & sproom hiring task Project Report

# Backend Architecture and Design Choices

#### Overview

The Timelogger project backend architecture is built with scalability, security, and performance in mind, facilitating a seamless user experience on the frontend while ensuring data integrity and responsiveness.

Technology Stack

- .NET Core 7.0: Chosen for its cross-platform support, performance, and robust ecosystem.

Architecture Overview

Clean Architecture

Adopted Clean Architecture principles to ensure that our application's domain and application logic are independent of the UI, database, and external frameworks. This separation of concerns allows for greater modifiability, testability, and maintainability.

- \*\*Domain Layer\*\*: Contains our core business logic, including entities like `Project` and `Time Registration`, and interfaces for repositories. It's the most inner part of our architecture.

- \*\*Application Layer\*\*: Houses use cases that execute operations on the domain entities, driven by interfaces defined in the domain layer. It communicates with the persistence layer through interfaces, adhering to the Dependency Inversion Principle.

- \*\*Infrastructure Layer\*\*: Implements interfaces defined in the application layer, particularly those related to data access. Entity Framework Core is used here to interact with SQL Server.

- \*\*Presentation Layer\*\*: Our API controllers reside in this layer, acting as the entry point for HTTP requests. They depend on the application layer to process commands and queries.

#### Design Patterns and Best Practices

- \*\*Repository Pattern\*\*: Abstracts the data layer, allowing us to swap out the database engine without impacting the application layer.

- \*\*Unit of Work\*\*: Manages transactions, ensuring that operations involving multiple repositories are completed successfully or rolled back entirely.

- \*\*Dependency Injection\*\*: Extensively used to inject dependencies at runtime, simplifying configuration and promoting loose coupling.

- \*\*CQRS (Command Query Responsibility Segregation) \*\*: Adopted for separating read and write operations, optimizing performance, and scalability. (Not Implemented)

- \*\*Domain-Driven Design (DDD)\*\*: Influenced our approach to modelling the domain, focusing on the complex business logic and rules.

#### Database

InMemory

#### Authentication and Authorization

Not implemented

#### Set up

## Back-end is set to run on port 44325

After running the API project should open link:

[localhost:44325/api/projects](https://localhost:44325/api/projects)

On the picture below can be seen several project objects instantiated from the constructor of ProjectInMemoryRepository.cs

A screenshot of a computer program

Description automatically generated

Similarly, several time registration objects are also instantiated on the picture below.

A screenshot of a computer

Description automatically generated

#### Architecture

A screenshot of a computer

Description automatically generated

#### API

Exports the crud methods to which the front-end binds and requests data.

For example, HttpGet GetAllProjects fetches all projects set in the constructor of the ProjectInMemoryRepository where also the implementation is provided

A computer code with text

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

#### ## Core

In the Core we have the Application logic where the contracts are defined IGenricRepository IProjectRepository, ITimeregistrationRepository

Also exception handling and logging contract for IappLogger.

Due to time frame EntityFramework was not implemented but I defined the features where would create the logic with commands and queries for storing and retrieving data

Application service registration class is where the services for the application layer are / can be defined and after summoned in the assembly and called in the start up

A computer screen shot of a program

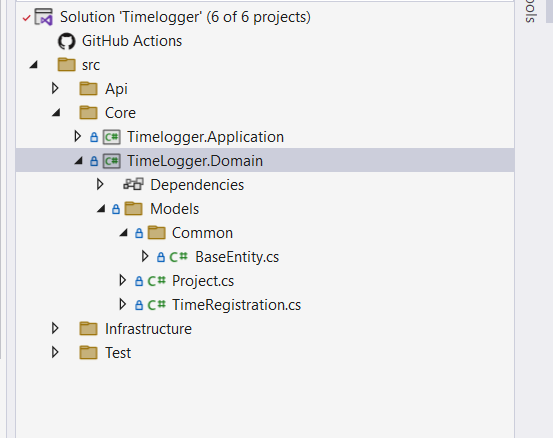
Description automatically generated

A screenshot of a computer

Description automatically generated

#### ## Domain Model

Domain model is where data is defined.



For project modularity an abstract BaseEntity was created which can be interpreted from the other entities supplying other models with the base fields and properties.

A screenshot of a computer program

Description automatically generated

Project class below defines the fields for the Project model.

A computer screen shot of a program

Description automatically generated

Time Registration class defines the fields for the time registration.

A screen shot of a computer program

Description automatically generated

#### Infrastructure Layer

In the infrastructure layer, the interfaces are implemented with the concrete method implementations.

A screenshot of a computer

Description automatically generated

#### Test

For testing was created x unit project and added moq and shuldly plugins.

A screenshot of a computer

Description automatically generated

#### Frontend

Runs on port 3001 and in the .env.develpment file the base url pointing to the backend is configured

A screenshot of a computer

Description automatically generated

Frontend is structured in folders for the different layers and components. In the following pictures below can be seen the structure of client.

The endpoints folder is where the client endpoints pointing to the API exposed methods are defined.

This can be seen in the picture below.

A screenshot of a computer

Description automatically generated

To run the React client, might have open the terminal and run command (npm install ) also is needed node.js

To start the client, execute command ( npm start ).

When the client starts shows this UI page named TimeLoggerHomePage.tsx and fetches the projects and timeregistrations from the backend.

A screenshot of a computer

Description automatically generated

On the picture bellow can be seen creation of a project. It happens when click Add Project button. Which expands and shows the field’s project name project description deadline isProjectCompleted.

If we select DateTime before the day today checkbox Completed is automatically checked and deadline is set to true which can be seen in field Completed in the Projects table. If we chose a date after today the checkbox is not checked but we always have the option to mark it to test different scenarios. When we click the button Create Project the project is created at the back end.

A screenshot of a computer

Description automatically generated

Besides React Timelogger homepage there are two other routs: Projects and Timeregistrations

If we click on Projects, we can see our newly created project 434 with Id 4 and Completed true

A screenshot of a computer

Description automatically generated

If we select date after today as deadline 15/02/2024 Completed is not checked

A screenshot of a computer

Description automatically generated

and when create the project by Create Project button click the field is set to false in the table.

There is also Toggle Sort order button which toggles the deadline between ascending and descending order.

A screenshot of a computer

Description automatically generated

# Timeregistrations

We can add a Timeregistration and connect it to a project by clicking add Timeregistration button.

It can be seen on the picture below.

There are several validations connected to a time registration. The Create Time registration button will not become clickable until all the validations required are forfeited.

First depending on the Project Id. Project with project Id which is completed, and value is true.

If you select such for example id 4 Create Time registration button stays locked.

Second, on the dates. There is a start date and end date field which calculates the Time Spent in Hours field. End Date field must be after start date, or the button stays hidden.

Lastly, time registration have to be at least 30mins between selected time from start to end then button unlocks and time registration can be created.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated