Yair Zederman – 208991711
Noga Zax – 316490259

High Level Project Description:

Task-Manager Project

Topics in Information Systems and Programming Languages - Mini Project

Table Of Contents:

Yair Zederman – 208991711
Noga Zax – 316490259

How to run this project:
1. Clone the git repository:

Git clone https://github.com/nogazax/miniproj-zederman.git

2. Run the project

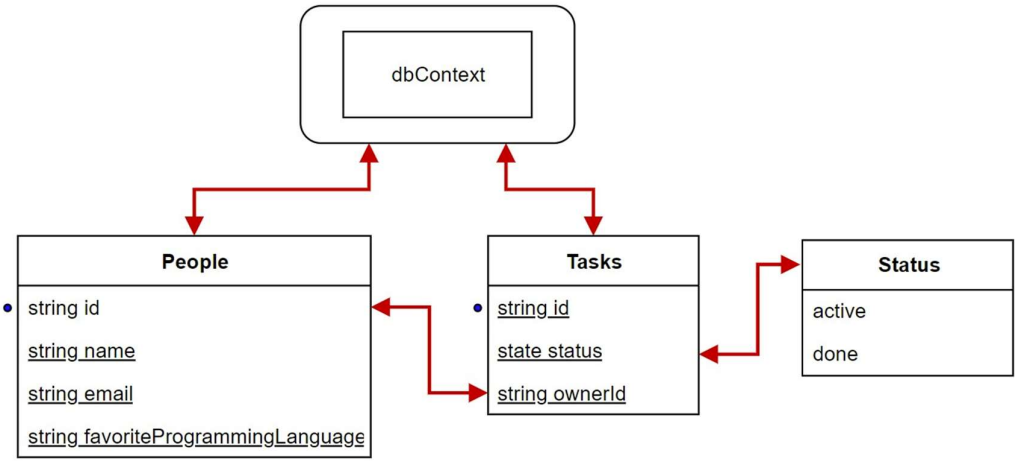
dotnet run --project miniproj-zederman/dotnetProj/dotnetProj.csproj

NOTE: web server runs on port 8080 by default, works with your Swagger 😊

General Design Notes:

The server was implemented with abstraction of Data – Model – API in a way that keeps the pattern of designs like MVC, and ORM to use and save different objects in the database.

A diagram of modules in the project and their connections:



-Fields with underline are mandatory when a client uses a POST endpoint.

-Fields with blue dot are unique fields (therefore it would be impossible to add another field from the same class with the same value).

-We chose this model because the framework we are working with is using the structure of MVC- 'Model-View-Controller'.

-The classes above are abstractions of the database tables we are working with (ORM).

Yair Zederman – 208991711
Noga Zax – 316490259

Entities:

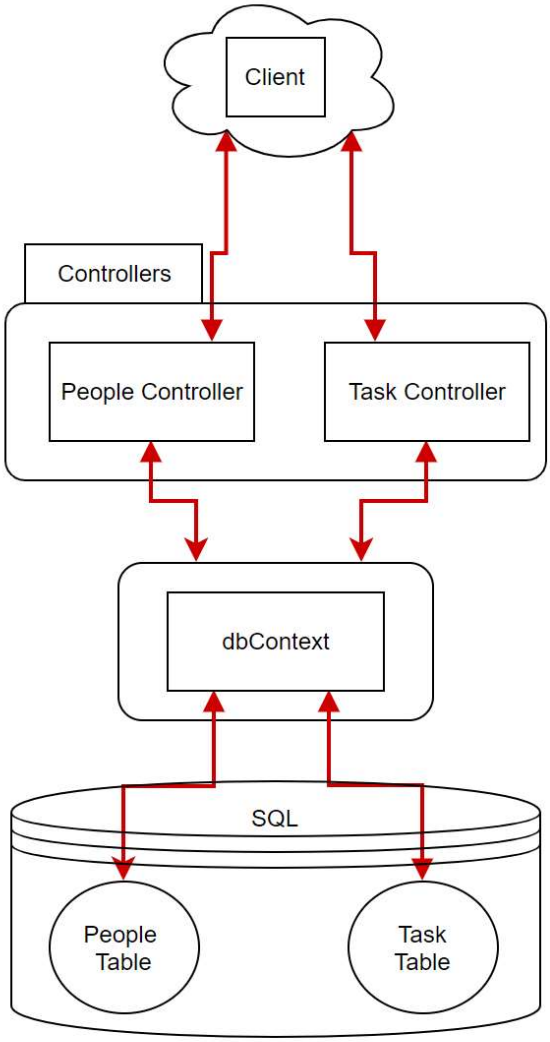
We are using the appropriate DTO's for each object in the system:

-People: When a new person object is created, the system will automatically generate a unique-id for him. When a person is returned, only not null values will be presented.

-Tasks: When a new task object is created, the system will automatically generate a unique-id and make sure that the supplied due-date is converted into Time object.
Moved most of the legal input checks to the DB, as we want to make the code as simple as possible.

System Design and Client Communication:

The system structure and client communication can be described using the following diagram:



Yair Zederman – 208991711
Noga Zax – 316490259

When a client sends a message to the server, it reaches the relevant controller, and redirected according to the request path (i.e post – api/people{id}/task).

The controller, using the database-context, communicates with the database to get or update the existing information as requested, and sends back a response to the client.

Data corruption:

The .NET framework is built in a way that it gives direct access to the database. Using the db-context of the project, which acts as a separation layer, we are keeping the data integrity (throws relevant exceptions for errors like duplicate keys or unauthorized access to non-existing entity).

In addition, we implemented manual checks for any input coming from the user to make sure that the requests will not damage the database.

Errors:

For any request containing illegal values according to the API, we send the appropriate response error.

Concurrency:

The server is expected to handle multiple requests simultaneously. The advantage of the framework we chose (.Net) is that the concurrency is handled by the SQL server, and the parallel connections are being handled by the controllers.