EndoTrack Backend Documentation

Luigi Taglialatela

September 2020

1 Introduction

The application's backend is a very simple RESTful web API, which retrieves data from a relational database hosted on Microsoft Azure. The app is written in .NET Core 3.1, and it employes Entity Framework Core as an ORM. It's fully unit-tested: I've employed XUnit as a test suite.

2 DB Schema

The database schema is pretty straightforward. There are only three tables: Customers, Machines and Processes. There's a one-to-many relationship between customers and machines, and therefore the Machines table contains a foreign key pointing to the Customers table's primary key.
There's a one-to-many relationship between machines and processes, and therefore the Processes table contains a foreign key pointing to the Machines table's primary key.

3 Architecture

The application is made up of an executable, "Presentation", which contains the api controllers, and two shared libraries: "Entities", which contains the domain entities, and "Persistence", which is responsible for retrieving data from the DB through the repository pattern, and lies at the core of the application's logic.
The "Persistence" shared library contains three main interfaces: "ICustomersRepository", which is responsible for interacting with the Customers table; "IMachinesRepository", which is responsible for interacting with the Machines table; "IProcessesRepository" which is responsible for interacting with the Processes table.
These interfaces are bound to their implementations through .NET Core's native IoC container in the Startup class of "Presentation"
It's very important to note that nowhere in the program are domain entites passed over the wire: they get mapped to and from deticated "ORM Entities" for every DB interaction, and to and from dedicated "View Models" for every client interaction.