Name: Samuel Renan Costa Morais **Email:** samuelrenancm@gmail.com

Project Architecture Report

The purpose of this document is to explain the architecture used to implement the proposed challenge.

Stack Utilized

The client part was developed using the react library as requested

The the front end of this project uses the following libraries:

- AxiosJs – fetch back end information;
- Redux - Facilitate communication between components and project scalability;
- Material UI - Facilitate styling

The the back end of this project uses the following packages:

- AutoMapper - Automate conversion between ModelView and Model;
- Dapper – Data persistence framework;
- SwashBuckle - generate Swagger and facilitate manual testing.

Solution Architecture

The client side is divided into three main modules: Actions, Reducers and Components. Actions are the functions responsible for communicating with server side. Reducers are responsible for storing the state of the variables used in several components simultaneously. Components are all components used directly to generate visual components to use the solution.

The components dispatch the actions that update or search for information on the server side. With that, the actions receive this data and update the data through the Reducers. When reducers are updated and a new updated state is generated, components use that new state to update themselves.

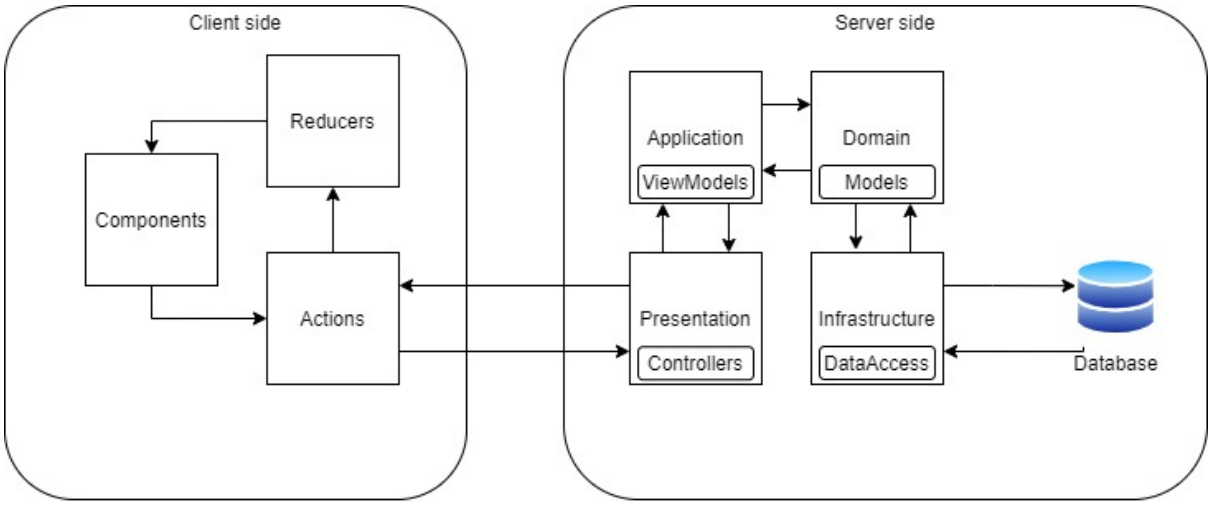

Figure 1: Solution Architecture diagram

On the server side, the solution is divided into four layers: Presentation, Application, Domain, Infrasctructure. The Presentation layer is responsible for communicating with the client side returning data and error messages. The Application layer is responsible for converting the received data into the format used in the Domain, another function of this layer is to check if the received data is valid. The Domain layer concentrates the business rules of the solution and the Infrasctructure layer is responsible for communicating with the Database.

Each layer communicates with its back and front layers. A CrossCutting layer will be implemented to facilitate error message logging.

The solution presented can be found at the following link: StockManager