WebApp

Create a single page web application! The application must use SPA frameworks on the client-side (ReactJS, etc.), Ninject and Entity Framework Code-First on the server-side.

The application shall have two screens; the user can switch between these screens without reloading the whole page.

On Screen A, the user can see the list of the products (name, price, description), can add or remove products, or modify products. They must be stored in the database. Changing an existing product, creating a new product or deleting a product must not cause the whole page to be reloaded, but the item must be sent to the server and stored in the database. Every operation must be logged. Log entries must contain a timestamp, and a message: create, update or delete product.

On Screen B, the user can see the log entries. They must be ordered by the timestamp (latest first), and they must contain the date (formatted according to the users current locale settings), and the log message.

Switching between these screens must re-populate the values from the server (refresh the lists), but without reloading the whole page.

The logging and the products database schemas must be defined in separate Database Contexts and separate Visual Studio projects.

The tool can have a startup project (Asp.NET MVC5 project), the domain projects (Logging and Products), and an Extensibility project (class library to define the interfaces).

The startup project must not have direct reference to the domain projects! It can only reference the Extensibility project. However, it is allowed to add the domain projects to the build dependency of the startup project. Extensibility project cannot have reference to the startup project, nor the domain projects. It can only reference NuGet packages and built-in namespaces.

Domain projects must use the repository pattern to manipulate data. The repository interface definitions must take place in the Extensibility project, and the implementation shall be in the corresponding domain project. Each domain project should bind these interfaces in their own Ninject module. The startup project can inject these interfaces in the corresponding Controllers. It is allowed to have DTO (Data Transfer Object) classes for products and log entries in the Extensibility project. The repositories must not return entity objects, but DTO objects.
Client-server data manipulation should be implemented in WebApi2 controllers.