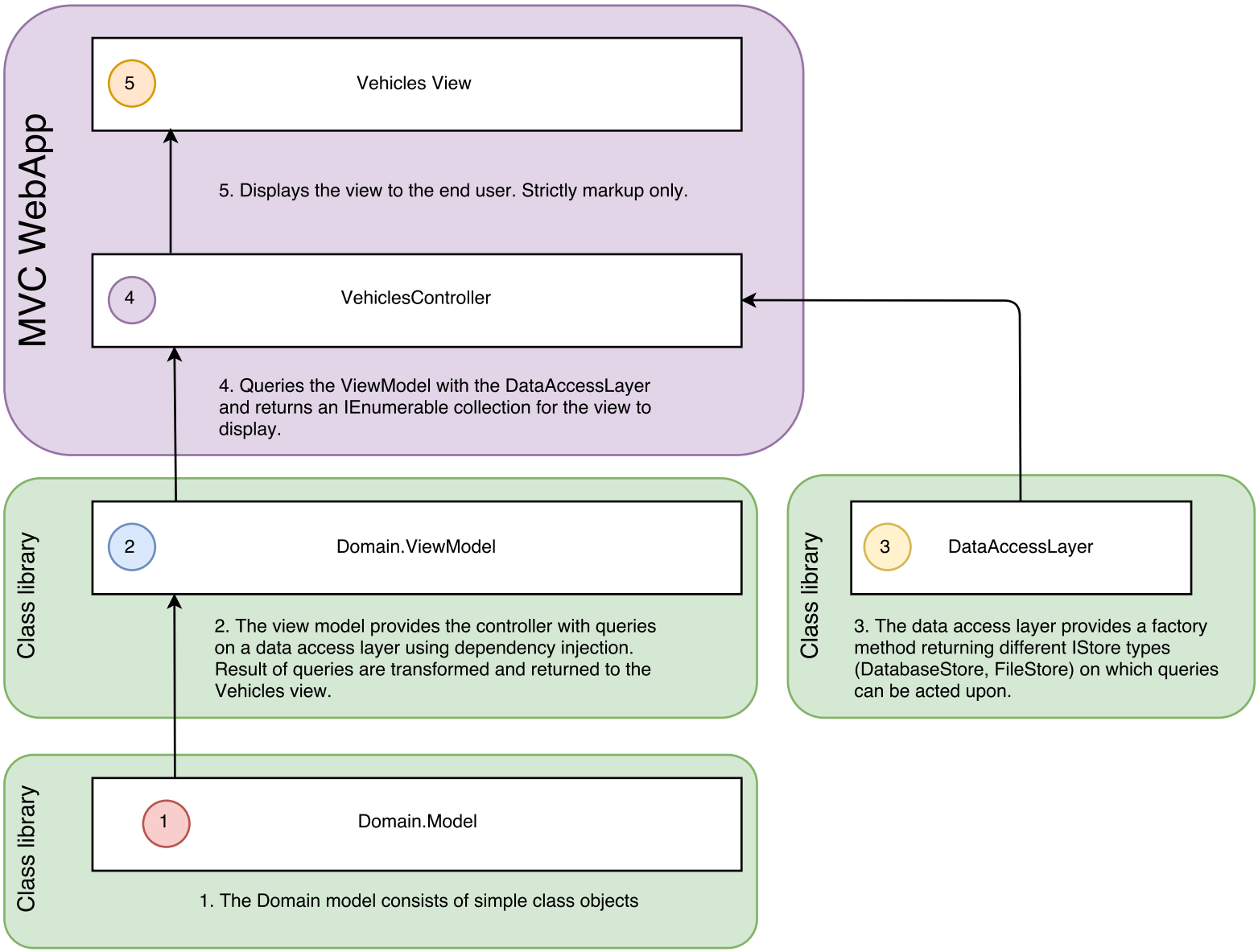

Design notes

* Each class library has its own individual tests written.
* Testing is much easier as there is no tight coupling between the components.
* Able to have real or mock up data shown to the user by using a factory method. New IStores can be returned easily by implementing the IStore interface.
* Ability to test front end and backup components independently.
* Ability to switch between front end frameworks easily if MVC is no longer required for example as Domain.ViewModel, Domain.Model and DataAccessLayer components are not dependent on the MVC framework.
* Consistent behaviour between front end and back end since all queries go through the ViewModel.
* Small files and few dependencies means the solution builds faster.
* Data transfer objects are used between the layers to provide flexibility if the model changes between components.
* Front end can be built as soon as a common interface is defined between the ViewModel and Controller.
* Using the idea of CQS in the ViewModel where results are returned by running queries which are idempotent.
* Domain.Model is set to have internal only access modifier for better encapsulation.

Technology used

* Entity framework (Used by DatabaseStore) in DataAccess class library.
* Microsoft MVC (VehiclesController and View).
* Bootstrap (as part of MVC solution).
* Microsoft Unit Testing Framework.
* SQL Server (Local instance with integrated Windows authentication running required for DatabaseStore).

Patterns

* MVVM variant (The View is implemented using Microsoft MVC rather than building a separate view component)
* Factory (DataBaseLayer component)
* Adapter (Using DTOs to convert between component layers)