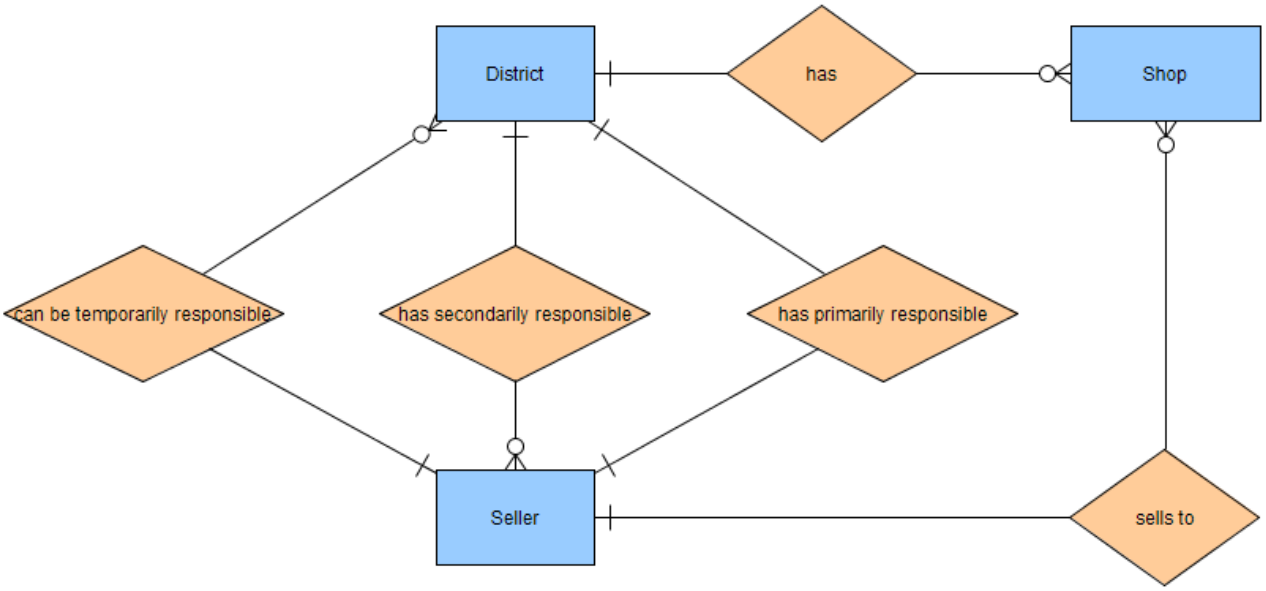
Documentation

Business Rules

Entity	Action	Relation	Entity
Seller	sells to	none to many	Shop
District	has	none to many	Shop
District	always has primarily responsible	one	Seller
District	has secondarily responsible	none to many	Seller
Seller	temporarily responsible for	none to many	District

Entity Relationship Diagram



Enforcing Business Rules in UI

Every district always has one primary seller – the user cannot remove the primary seller before choosing another one first.

A district does not need to have secondary sellers – the user can remove all secondary sellers.

A seller can be temporarily responsible for more than one shop – The user can choose a seller that has already been assigned as the primary to another district as well.

Database Design

I decided to create a table for each entity, and another table to display the relation between the sellers and the districts (Seller2District). This table indicates if a seller is primarily responsible for a district. All tables include an IsDeleted column to ensure that history of previous user choices is kept, instead of deleting rows from the database.

Components

Components of the project shown if folder structure.

Assignment

Controllers

The MVC controllers.

DAL

The AssignmentContext class inherited from the Entity Framework DbContext class provides the connection to the database. The folder also includes the initializer class to drop and recreate database if the model changes.

DBScript

Includes the SQL script that can be used to insert test data.

Models

The models represent the data for District, Seller, Shop, and Seller2District. They are independent from View and Controller.

Repos

The repositories have general methods (such as Get and Insert) that provide access to the database which can be reused in the services. Every repository has an interface which makes it possible to use any repository that fulfills the interface.

Services

The services contain the business logic. The business logic is separated from the controllers to make modifications and testing simpler. Every service has an interface so that the controllers can use any service that fulfills the interface.

ViewModels

The ViewModels are used to transport data between views and controllers.

Views

The project contains five views which display the data and send user actions to the controllers.

Assignment.Tests

Mocks

This folder includes the mock repositories for Shop, Seller, and Seller2District used in the unit tests.

Services

Includes the unit tests for testing the services. The repository interfaces make it possible to use the mock repositories in the test classes.