“Bird Season” Research Project

Team Members

Antoinne Stewart

Ricardo Andrews

Gabrielle Watson

Table of Contents

Introduction…………………………………………………………………..3

Project Requirements……………………………………………………..…..4

User Interface Requirements………………………………………………….8

Programming Languages Used……………………………………………..…9

Frameworks Used……………………………………………………………..10

Tools Used……………………………………………………………………..11

Incorporated Research Concepts…………………………………...…………..12

References………………………………………………………………..……..25

**Introduction**

**Project Background**

The National Environmental and Planning Agency (NEPA) is an agency of the Ministry of Health and Environment established in 2001 to manage and protect the natural resources and assets of Jamaica. Every year, subsequent to an in-depth feasibility study, the agency facilitates Bird Hunting Season of select game birds in select quantities. The process requires licensed hunters to provide submissions at the end of the season by registered mail or in person. NEPA has considered the proposal from their technical team to automate the data capture, reporting and regulation process. One component of this process is the Bird Individual Reporting Database (B.I.R.D) system which our team has been asked to develop. B.I.R.D will consist of one (1) or two (2) client applications that will communicate with a Main Server application. The Main Server application will facilitate automatic game bird identification and reporting through feedback from a sensory module comprising of a Sensory Module Base Station.

# **Project Requirements**

· Priority Definitions

The following definitions are intended as a guideline to prioritize requirements.

· Priority 1 – The requirement is a “must have” as outlined by policy/law

· Priority 2 – The requirement is needed for improved processing, and the fulfillment of the requirement will create immediate benefits

· Priority 3 – The requirement is a “nice to have” which may include new functionality

Functional Requirements

|  |  |
| --- | --- |
| **Requirement** | **Priority** |
| As a Bird Hunter, I should be able to register for the system by specifying my first name,  last name, bird hunting license no, email, contact number, profile photo and password | **1** |
| As a Bird Hunter, I should be able to log in using my email and password. | **1** |
| As a Bird Hunter, I should be able to view my profile details. My profile details include my first name, last name, bird hunting license no, email, profile photo. | **1** |
| As a Bird Hunter, I should be able to view my hunting statistics | **1** |
| As an Administrator , I should be able to record a bird hunting session for a hunter by  specifying the game reserve (see pages i and ii for list of Game reserves), session(i.e.  morning/afternoon), date, the amount of each game bird (see Table 1 above) shot, the  amount of any other type of bird shot (see policy document for examples of protected  birds) | **1** |
| As an Administrator, I should be able to identify an existing hunting zone as available for hunting this season. This will allow bird hunters and administrators to choose the game reserve when recording hunting sessions | **1** |
| As an administrator, I should be able to view the total amount of birds hunted in a particular year as a chart. | **1** |
| As an administrator, I should be able to view a list of all bag limit violations. Each violation has a system generated Id, violation type, bird hunter name and bird hunting session. | **1** |
| As a Bird Hunter, I should be able to initiate an automatic bird hunting recording session. An automatic recording session should allow the Server to request from the Sensory Module Base Station Server a randomly generated a list (3-25) of hunted birds (game/protected). At the end of the session (i.e. after the Sensory Module Base Station Server has sent the list to the server), the server should return the generated list to the client to be confirmed. | **1** |
| As a bird hunter, I should be able to confirm the list of birds returned by the automatic  bird hunting record session before it is saved to the database | **1** |
| As a bird hunter, I should be able to cancel my automatic bird hunting recording session. | **1** |

## 

## **User Interface Requirements**

## **Usability**

Learnability

Features are easy to use and easy to learn and remember how to use.

Performance

The performance of users in completing a task. For example, 99.5% of users will be able to view videos.

Accessibility

Requirements that things be useful to as broad a group of people as possible..

User Friendliness

Requirements that things be delightful to use.

Error Tolerance

Requirements that things be difficult to get wrong.

Information Scent

Requirements that information and tools be easy to find.

Flow

Requirements that users be able to flow through task without being interrupted.

**Programming Languages used**

Typescript

An open-source programming language developed and maintained by Microsoft. It is a strict syntactical superset of JavaScript, and adds optional static typing to the language. TypeScript is designed for development of large applications and transcompiles to JavaScript. Typescript was used solely in client side development.

Javascript

A high-level, interpreted scripting language that conforms to the ECMAScript specification. JavaScript has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions. Javascript was used in parts of client side and server side development.

C#

A general-purpose, multi-paradigm programming language encompassing strong typing, lexically scoped, imperative, declarative, functional, generic, object-oriented, and component-oriented programming disciplines. C# was used solely in server side development.

SQL

A domain-specific language used in programming and designed for managing data held in a relational database management system, or for stream processing in a relational data stream management system. SQL was used solely in database development.

There may have been other programming languages used that were not specified, as some code was automatically generated and these files and programming languages may have been overlooked.

**Frameworks Used**

Angular 7

Angular 7 is an open source JavaScript framework for building web applications and apps in JavaScript, html, and Typescript. Angular provides built-in features for animation, http service, and materials which in turn have features such as auto-complete, navigation, toolbar, menus, etc. The code is written in Typescript, which compiles to JavaScript and displays the same in the browser. (Tutorialspoint)

Asp.net WebAPI

ASP.NET is an open-source server-side web application framework designed for web development to produce dynamic web pages developed by Microsoft to allow programmers to build dynamic web sites, applications and service. (Microsoft)

The ASP.NET Web API is an extensible framework for building HTTP based services that can be accessed in different applications on different platforms such as web, windows, mobile etc. It works more or less the same way as ASP.NET MVC web application except that it sends data as a response instead of html view. (Microsoft) It is like a web service or WCF service but the exception is that it only supports HTTP protocol.

Tools Used

Microsoft Visual Studio Community 2017 (Version 15.9.17)

Microsoft Visual Studio is an integrated development environment from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps.

Microsoft Visual Studio Code (Version 1.40.1)

Visual Studio Code is a source-code editor developed by Microsoft for Windows, Linux and macOS. It includes support for debugging, embedded Git control and GitHub, syntax highlighting, intelligent code completion, snippets, and code refactoring.

Microsoft SQL Server Management Studio 17 (Version 14.0.17199.0)

SQL Server Management Studio (SSMS) is an integrated environment for managing any SQL infrastructure, from SQL Server to Azure SQL Database. SSMS provides tools to configure, monitor, and administer instances of SQL Server and databases. Use SSMS to deploy, monitor, and upgrade the data-tier components used by your applications, and build queries and scripts.

Github Desktop (Version 2.2.1)

GitHub provides hosting for software development version control using Git. (Github.com)

Selenium

**Incorporated Research Concepts**

S.O.L.I.D Principles

These design principles encourage us to create more maintainable, understandable, and flexible software. Consequently, as our applications grow in size, we can reduce their complexity and save ourselves a lot of headaches further down the road. (Baeldung.com)

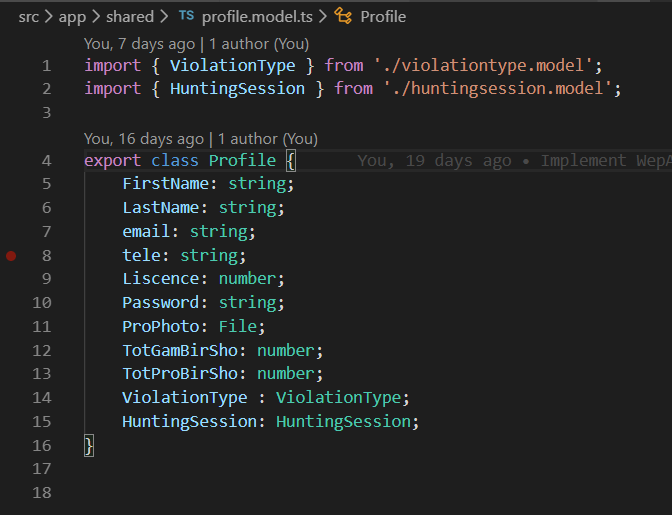
1. Single Responsibility

This principle states that a class should only have one responsibility. Furthermore, it should only have one reason to change.

How does this principle help us to build better software? Let's see a few of its benefits:

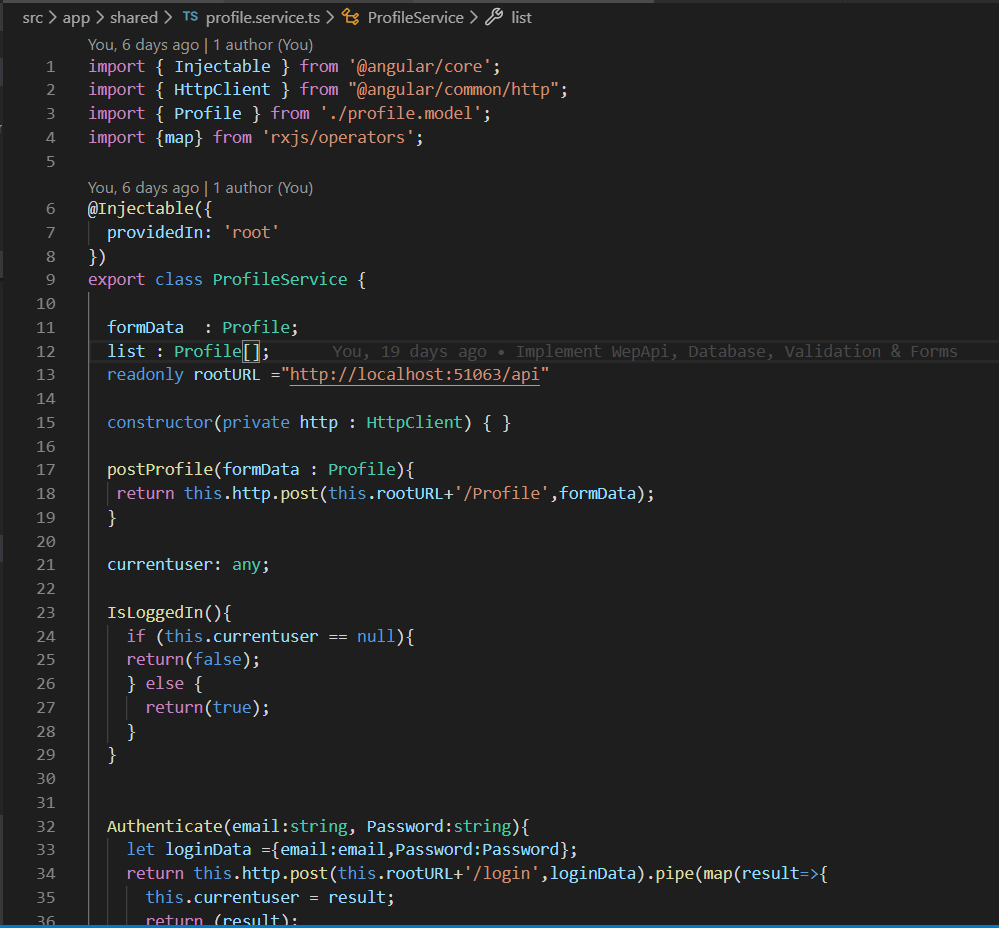
* Testing – A class with one responsibility will have far fewer test cases
* Lower coupling – Less functionality in a single class will have fewer dependencies
* Organization – Smaller, well-organized classes are easier to search than monolithic ones

Take, for example, a class to represent a user’s profile information:



In this code we store the First name, Last Name, email, telephone number etc. with an instance of Profile.

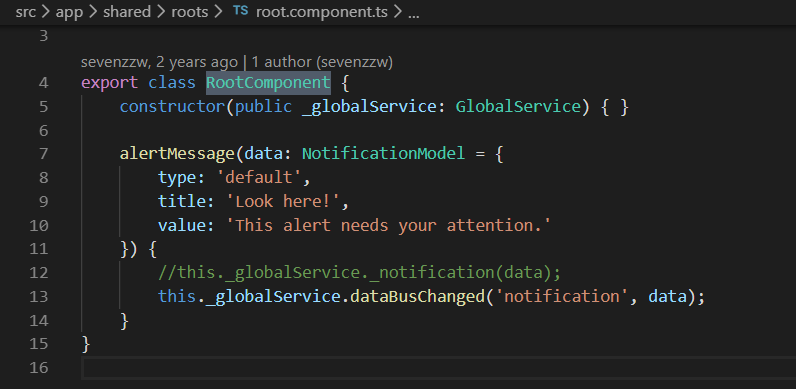
Now the project requirements specify that we need to be able to update and display a user’s profile information from a database and while that may be possible to do all in one class that would break the rule of Singleton pattern so here we have a separate class for such tasks.



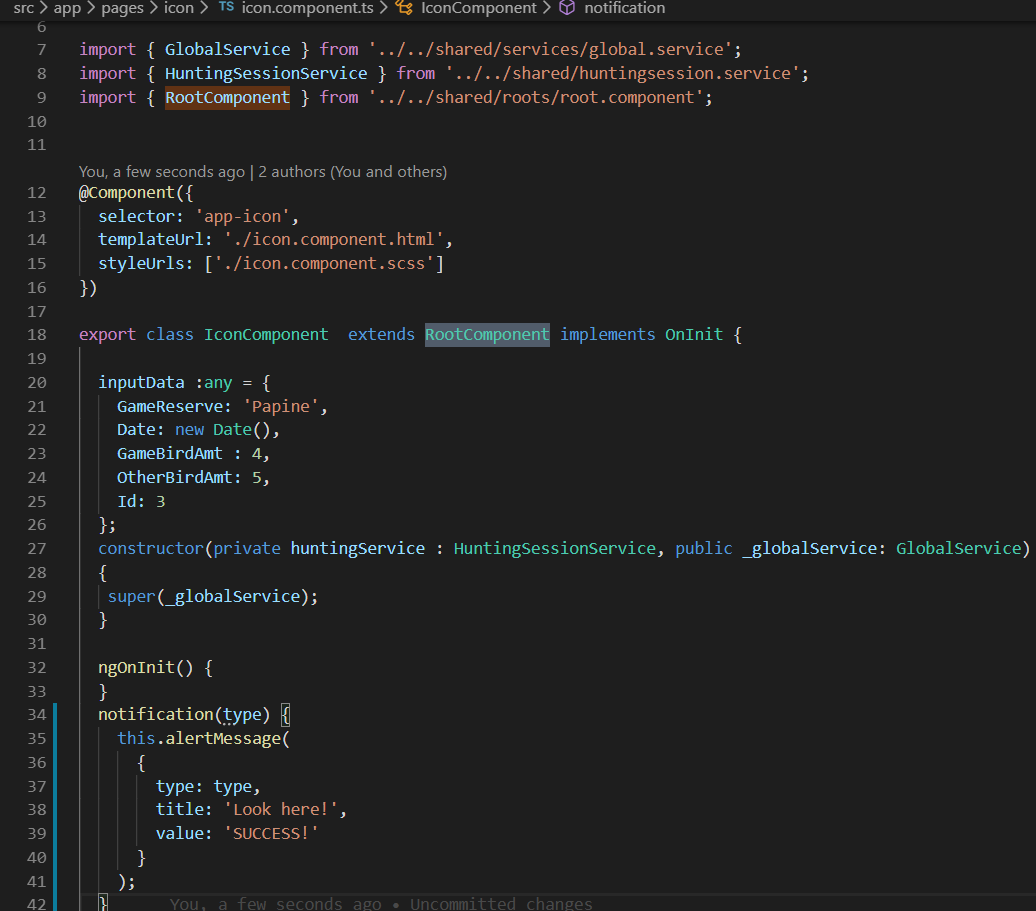
2. Open-closed Principle

Simply put, classes should be open for extension, but closed for modification. In doing so, we stop ourselves from modifying existing code and causing potential new bugs in an otherwise happy application. Of course, the one exception to the rule is when fixing bugs in existing code.

Here in the code we have implemented a Root class with a function “alertMessage” to display notifications when needed throughout the interface.



Now lets say we launch the application, and everyone loves it. However, after a few months, we decide the alertmessage needs to be changed. At this point, it might be tempting to just open up the root class and change the aforementioned function but who knows what errors that might throw up in our application. Instead, let's stick to the open-closed principle and simply extend our Root class:



3. Liskov Substitution

Simply put, if class A is a subtype of class B, then we should be able to replace B with A without disrupting the behaviour of our program. This concept however did not have to be implemented due to the nature of the application. There were no subtypes of any classes.

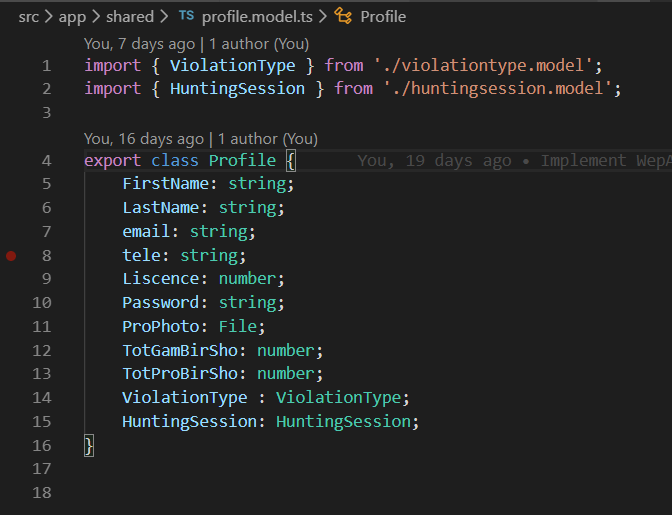
4. Interface Segregation

This simply means that larger interfaces should be split into smaller ones. By doing so, we can ensure that implementing classes only need to be concerned about the methods that are of interest to them. This too did not have to be incorporated as there were no large interfaces.

5. Dependency Inversion

The principle of Dependency Inversion refers to the decoupling of software modules. This way, instead of high-level modules depending on low-level modules, both will depend on abstraction.

Here, we're using the dependency injection pattern here to facilitate adding the ViolationType and HuntingSession dependency into the Profile class.

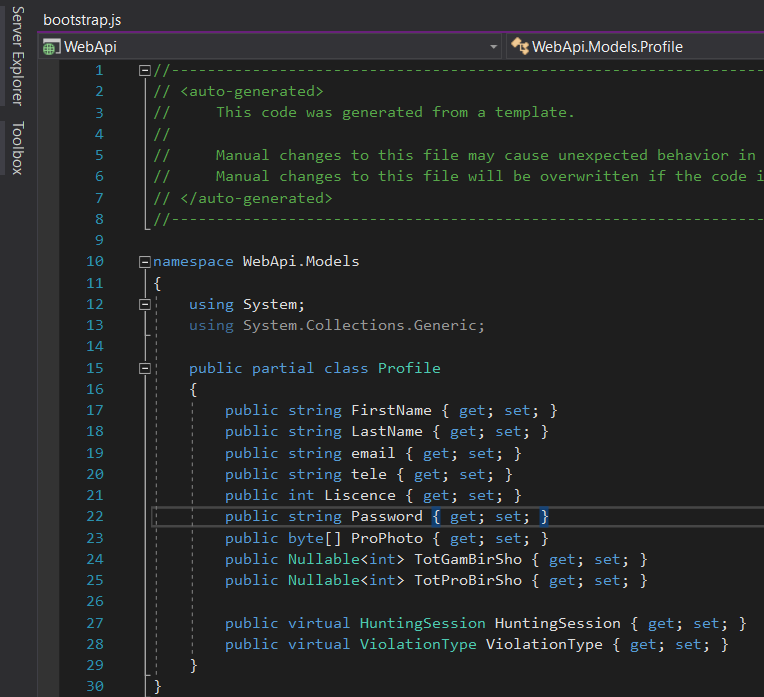


Repository Pattern

A Repository mediates between the domain and data mapping layers (like Entity Framework). It allows you to pull a record or number of records out of datasets, and then have those records to work on acting like an in-memory domain object collection, and you can also update or delete records within those data set, and the mapping code encapsulated by the Repository will carry out the appropriate operations behind the scenes. (kudchikarsk.com)

Entity Framework (EF) itself implements Unit of work pattern and somewhat loosely implements Repository pattern. With EF you can retrieve a set of records from the database in POCO models. Also, EF keeps track of changes for you within these models and save these changes on single SaveChanges method call.

Here we have a Profile entity in our application and this is how our profile repository interface looks:



And the implementation of the above interface with EF looks like this:

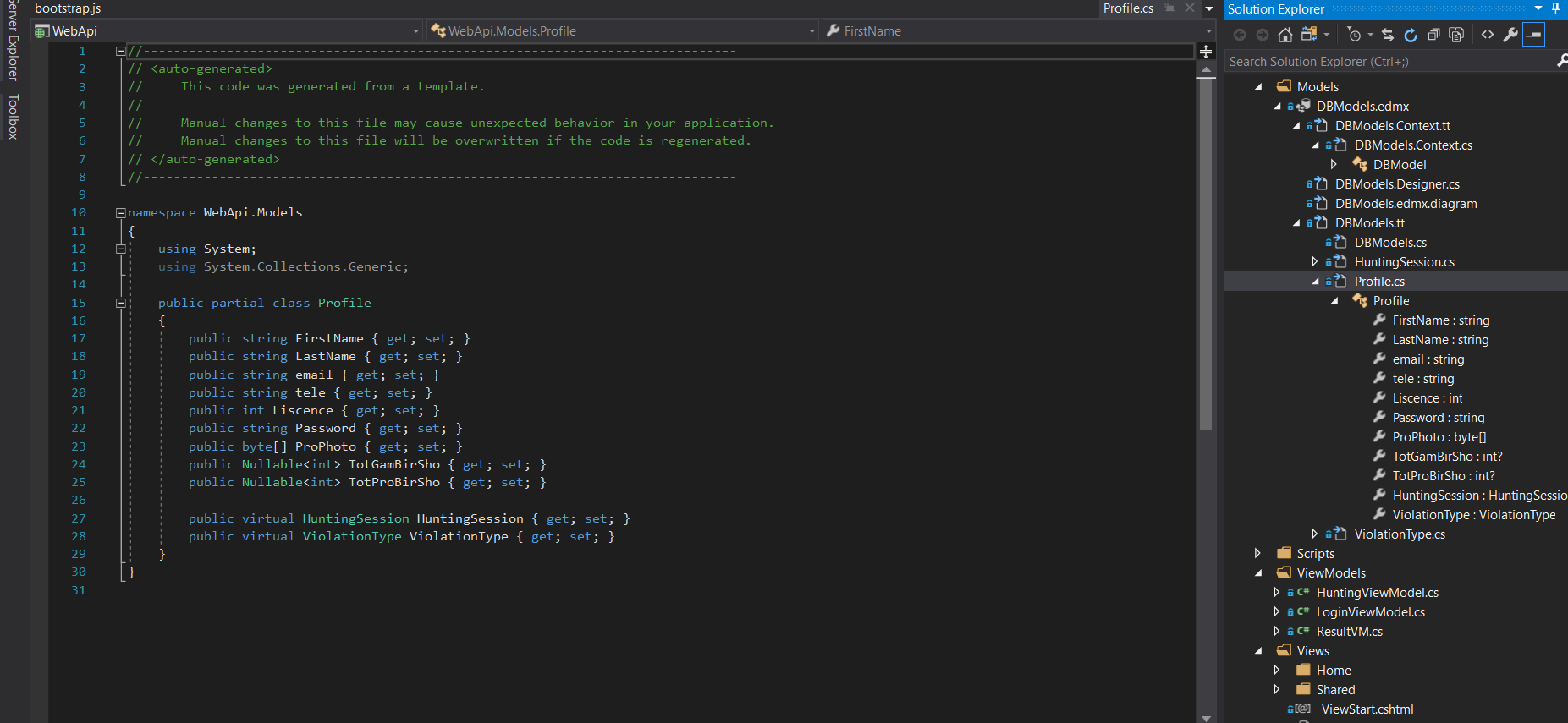


Model View Controller Pattern

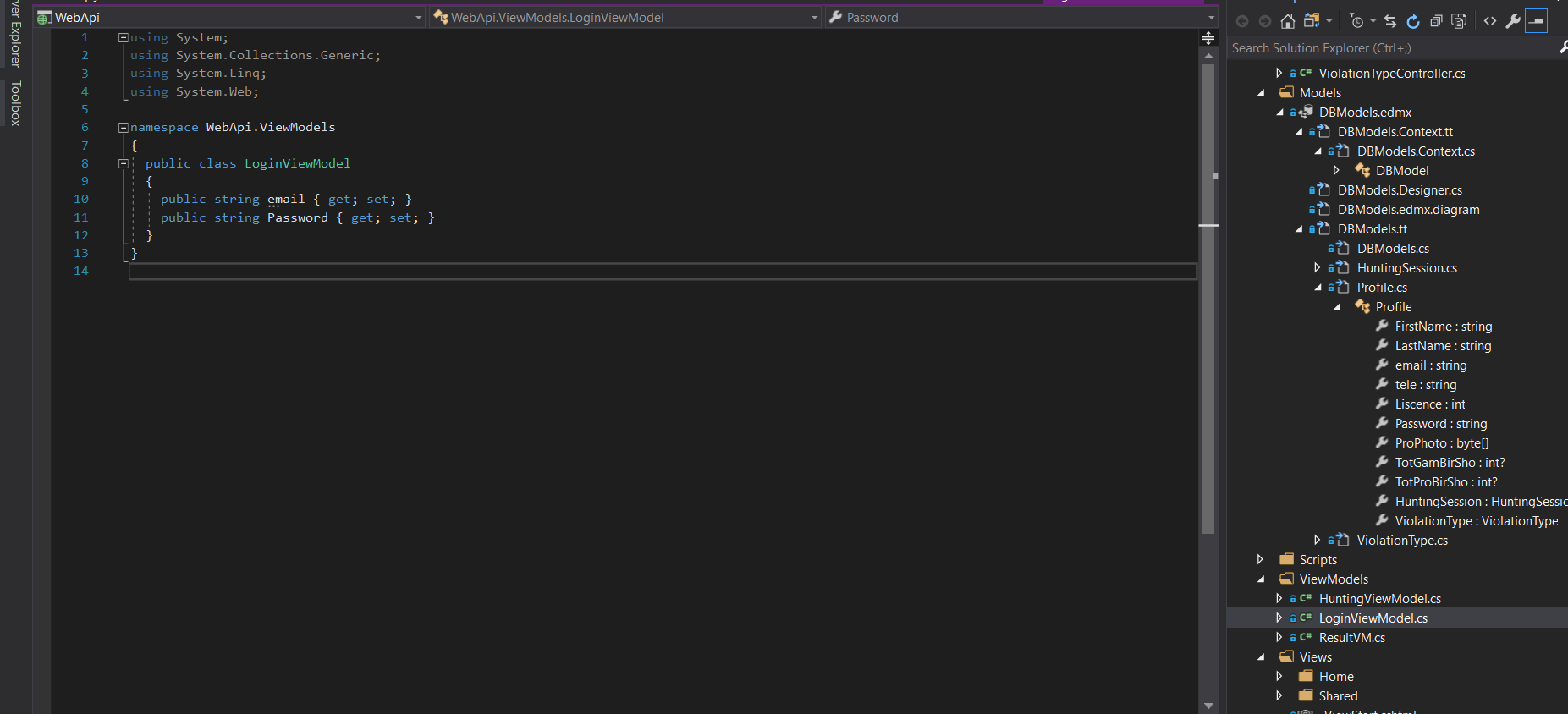
The Model View Controller (MVC) design pattern specifies that an application consist of a data model, presentation information, and control information. The pattern requires that each of these be separated into different objects. MVC is more of an architectural pattern, but not for complete application. MVC mostly relates to the UI / interaction layer of an application. (geeksforgeeks.com)



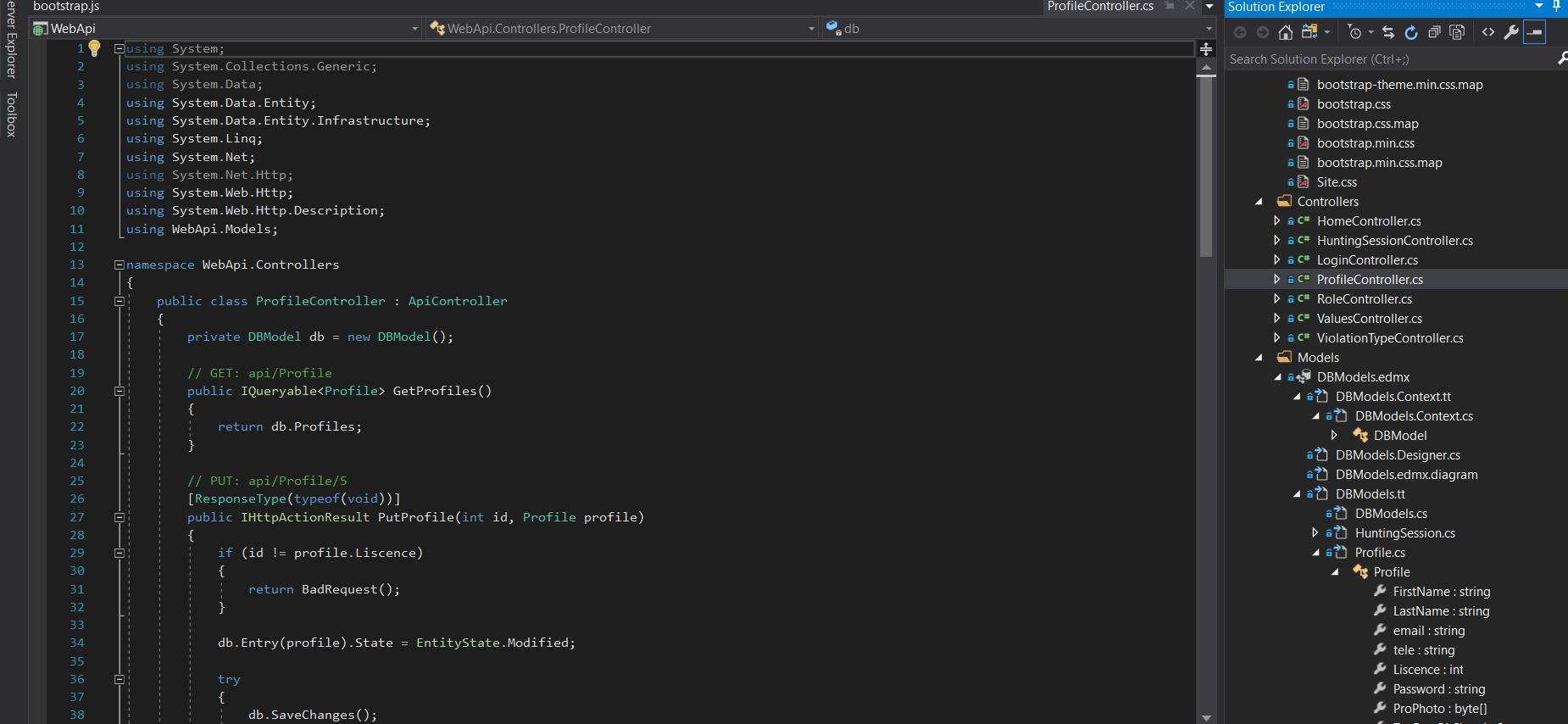
The Model contains only the pure application data, it contains no logic describing how to present the data to a user. Here we see how one of our “models” was implemented



The View presents the model’s data to the user. The view knows how to access the model’s data, but it does not know what this data means or what the user can do to manipulate it. Here we see how one of our “views” were implemented.



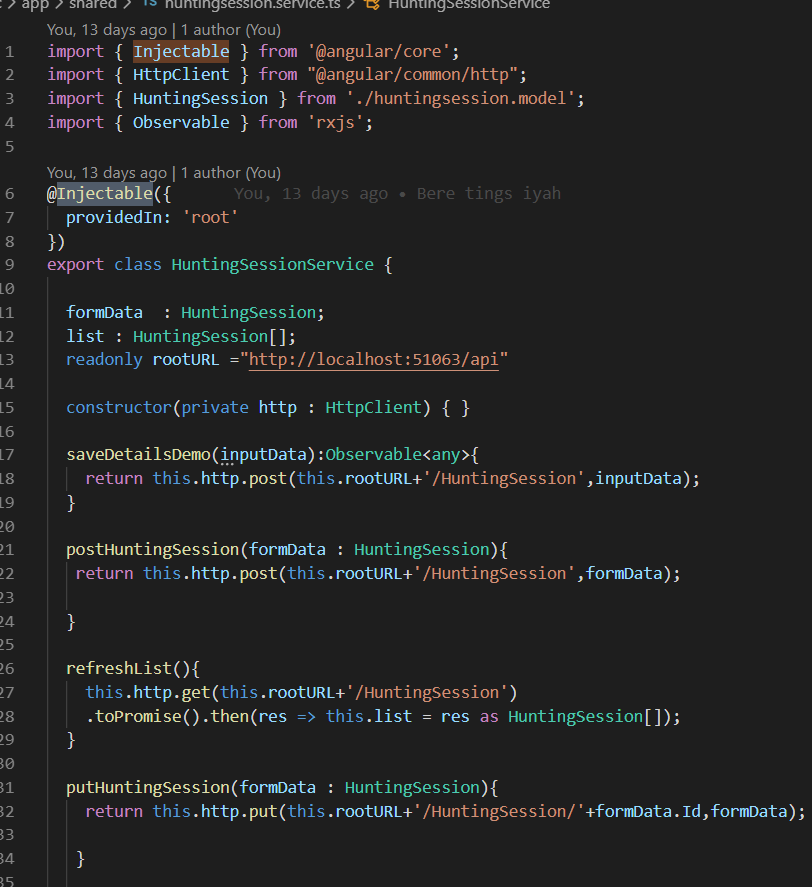
The Controller exists between the view and the model. It listens to events triggered by the view (or another external source) and executes the appropriate reaction to these events. In most cases, the reaction is to call a method on the model. Since the view and the model are connected through a notification mechanism, the result of this action is then automatically reflected in the view. Here we see how one of our “controllers” were implemented



Singleton Pattern

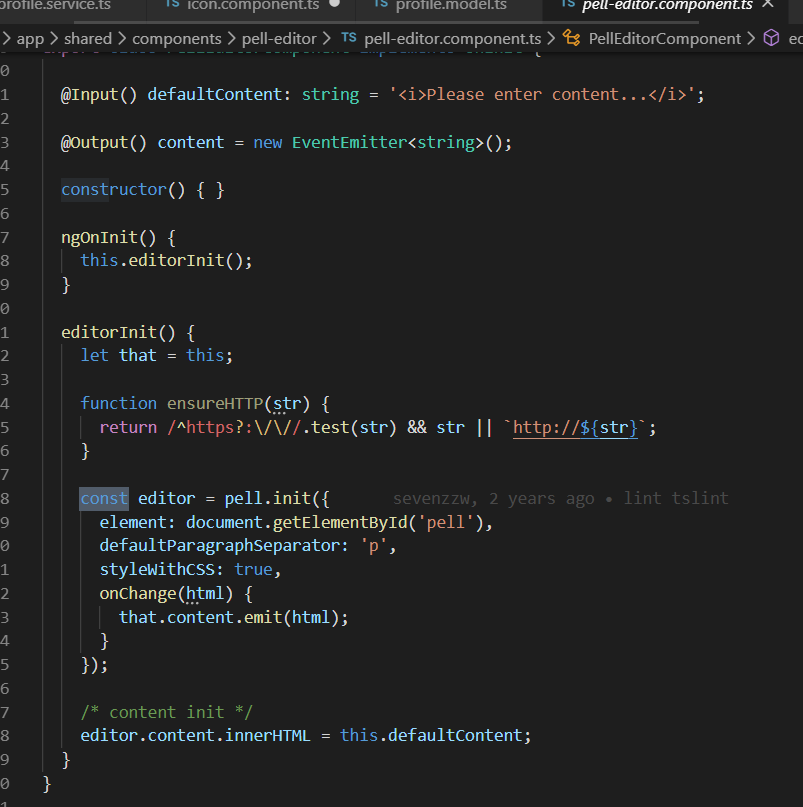
The singleton pattern is a design pattern that restricts the instantiation of a class to one object. There are two ways to make a service a singleton in Angular: Declare root for the value of the @[Injectable](https://angular.io/api/core/Injectable)() [providedIn](https://angular.io/api/core/Injectable#providedIn) property and include the service in the AppModule or in a module that is only imported by the AppModule. (angular.io)

Here is an example of the first within the application



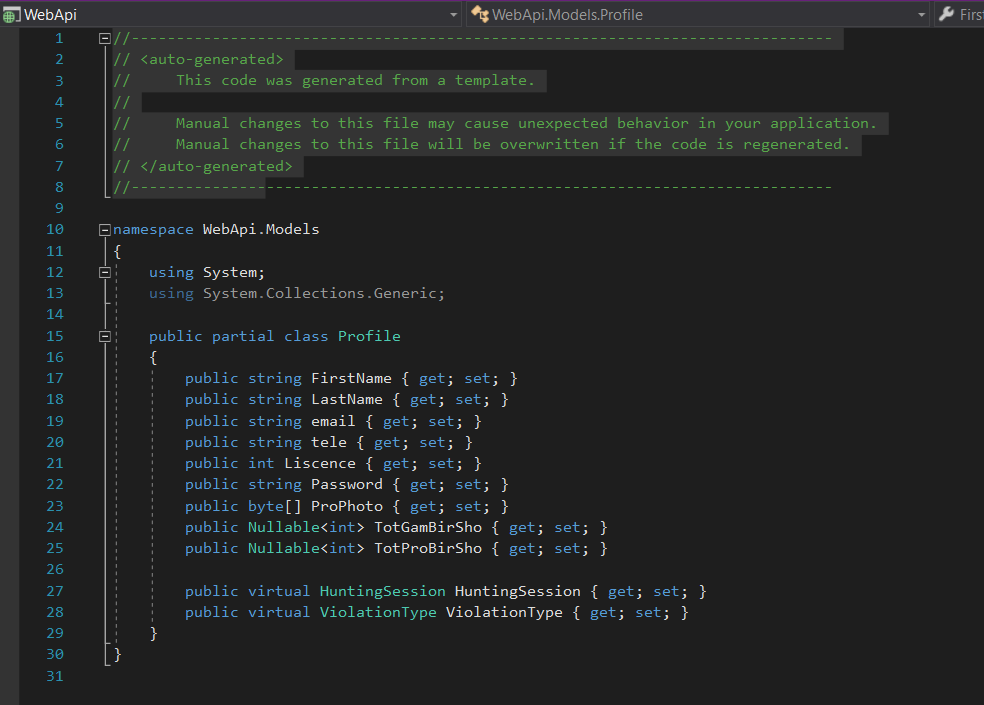
Factory Pattern

The factory pattern is one of the creational software design patterns. This design pattern provides a generic interface to create objects. It works with a creator that produces products. When using the factory pattern, we are not required to use the new keyword because the instantiation of additional classes are done in the factory implementation. The factory pattern can be used in situations where the object creation process is complex or in situations where multiple object are created that share the same properties. (blog.fullstacktraining.com) There implementation is here:



Code generator tool

A code generator is a tool or resource that generates a particular sort of code or computer programming language. Both angular and the WepApi frameworks automatically generated portions of code. (techopedia.com) Example below:



Source Control Management Tool

Source Control Management (SCM) is all about the way software changes are made. It has a number of goals which are fundamentally geared toward ensuring that development teams can deliver higher quality code changes at faster speeds. By improving tracking, visibility, collaboration and control throughout the release lifecycle, SCM tools provide more creativity, freedom and possibilities for developers when undertaking complex and challenging work. Moreover, SCM can protect the original source files from any kind of mishap and enables all team members to look at who has made what changes at what point (assessment-tools.ca.com). Version control was achieved by integrating git with visual studio using the tool Github.

Package Management Tools

A package manager or package-management system is a collection of software tools that automates the process of installing, upgrading, configuring, and removing computer programs for a computer's operating system in a consistent manner. A package manager deals with packages, distributions of software and data in archive files. Packages contain metadata, such as the software's name, description of its purpose, version number, vendor, checksum, and a list of dependencies necessary for the software to run properly. A package manager that was used in development was NuGet package manager.

Unit Testing and Test automation

Unit testing was implemented as the development team test every function/feature that was implemented directly after initial implementation to verify the efficiency of the code. Test automation was done using a tool called Selenium.

Continuous Integration and Continuous Integration Server

Continuous integration was achieved using Github with several “commits” as features and functions were being implemented.

**References**

<https://angular.io/guide/singleton-services>

<https://assessment-tools.ca.com/tools/continuous-delivery-tools/en/source-control-management-tools>

<https://www.baeldung.com/solid-principles>

<https://blog.fullstacktraining.com/factory-pattern-in-typescript/>

<https://dotnet.microsoft.com/apps/aspnet>

<https://github.com/>

https://www.geeksforgeeks.org/mvc-design-pattern/

<https://kudchikarsk.com/repository-pattern-csharp/>

<https://www.techopedia.com/definition/17062/code-generator>

<https://www.tutorialspoint.com/angular7/index.htm>