Notes

* The Application is deployed on Azure and it can be accessed here: <http://trainticketmachine.azurewebsites.net/>
* To run the application locally you need to open the solution As Administrator because the project is configured to create VirtualDirectories in IIS for the Web App and for the WebApi.
* I have created 2 sets of UI:
  + One using JQuery
  + Another one using AngularJS
* Used .NET Framework 4.5 to be able to use async programming (async await) for improving the overall throughput of a multi-core system.
* Created a prefix tree data structure to optimize the search based on prefix.
* Created different layers (API, BLL, Repository) and added dependency injection using Unity.
* Created Unit tests for all classes in those layers.
* A future improvement would be to implement the repository with an ORM framework like Entity Framework or NHibernate.
* Packages used:

<packages>

<package id="angularjs" version="1.4.3" targetFramework="net45" />

<package id="Antlr" version="3.5.0.2" targetFramework="net45" />

<package id="bootstrap" version="3.3.5" targetFramework="net45" />

<package id="jQuery" version="2.1.4" targetFramework="net45" />

<package id="jQuery.Validation" version="1.14.0" targetFramework="net45" />

<package id="Microsoft.AspNet.Mvc" version="5.2.3" targetFramework="net45" />

<package id="Microsoft.AspNet.Razor" version="3.2.3" targetFramework="net45" />

<package id="Microsoft.AspNet.Web.Optimization" version="1.1.3" targetFramework="net45" />

<package id="Microsoft.AspNet.WebApi" version="5.2.3" targetFramework="net45" />

<package id="Microsoft.AspNet.WebApi.Client" version="5.2.3" targetFramework="net45" />

<package id="Microsoft.AspNet.WebApi.Core" version="5.2.3" targetFramework="net45" />

<package id="Microsoft.AspNet.WebApi.HelpPage" version="5.2.3" targetFramework="net45" />

<package id="Microsoft.AspNet.WebApi.WebHost" version="5.2.3" targetFramework="net45" />

<package id="Microsoft.AspNet.WebPages" version="3.2.3" targetFramework="net45" />

<package id="Microsoft.jQuery.Unobtrusive.Validation" version="3.2.3" targetFramework="net45" />

<package id="Microsoft.Web.Infrastructure" version="1.0.0.0" targetFramework="net45" />

<package id="Modernizr" version="2.8.3" targetFramework="net45" />

<package id="Newtonsoft.Json" version="7.0.1" targetFramework="net45" />

<package id="Respond" version="1.4.2" targetFramework="net45" />

<package id="Unity" version="3.5.1404.0" targetFramework="net45" />

<package id="Unity.WebApi.5.1" version="5.2" targetFramework="net45" />

<package id="WebGrease" version="1.6.0" targetFramework="net45" />

</packages>

**Solid Principles**

|  |  |  |
| --- | --- | --- |
| Initial | Short Description | Application Usage |
| S | **Single responsibility principle**: a class should have only a single responsibility (i.e. only one potential change in the software’s specification should be able to affect the specification of the class) | Each class in the application has only one responsibility. |
| O | **Opened/Closed principle**: software entities should be opened for extension and closed for modifications | Each class in the application is closed for modifications but it is open for extension as it depends on the interfaces which can have different implementations. |
| L | **Liskov Substitution Principle**: objects in a program should be replaceable with instances of their subtypes without altering the correctness of the program. | All interface implementations could be changed without altering the correctness of the application. |
| I | **Interface Segregation Principle**: many client specific interfaces are better than one general purpose interface | All interface in the App are simple and very specific for client use. There is no interface for general purpose to amalgamate many functions. |
| D | **Dependency Inversion Principle**: should depend upon abstractions, do not depend upon concretions. | Every Concrete class has their dependencies injected by Unity. |

# Future improvements

* Add integration tests
* Create JS unit tests
* Exception logging
* DevOps, build configurations
* Implement Repository using Entity Framework or NHibernate.
* Authentication and Authorization