Crew Link Transformation

Low Level Document

August 22, 2019  
Version 0.1

**Document Control**

|  |  |  |  |
| --- | --- | --- | --- |
| **Prepared by** | Dharmendra Kumar Singh | **Designation** | Associate Architect |
| **Date** | Aug 22, 2019 | **Signature** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Approved by** |  | **Designation** |  |
| **Date** |  | **Signature** |  |

**Document Change History**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Date** | **Version** | **Section** | **Description** | **Author** | **Reviewer** | **Approver** |
| Aug 22, 2019 | 0.1 | NA | Microservices Design consideration | Dharmendra Kumar Singh |  |  |

**Table Of Content**

1 Introduction 4

2 Definitions, Acronyms and Abbreviations 5

2.1 Definitions 5

2.2 Acronyms and Abbreviations 5

3 Design Architecture Consideration 6

4 Development Tool 9

4.1 Middleware Development 9

4.2 Backend Development 9

4.3 Source Control 9

4.4 Development Approach 9

5 API flow Diagram 11

6 Microservice Technical Architecture 12

7 Non Functional Requirement 14

8 GDPR and PbD 17

8.1 Privacy and Security 17

8.2 Threat protection 17

9 Database Diagram 19

9.1 Master Tables 19

9.2 General Set up Tables 19

9.3 Office Set up Tables 19

9.4 Supplier Set up Tables 19

9.5 Booking Management Tables 19

9.6 Customer Set up Tables 19

# Introduction

ATPI CrewLink™ is a travel workflow management tool, specifically designed for the international Shipping and Energy industries. ATPI CrewLink™ will integrate with internal or third-party crew management tools and enables you to manage the entire travel life cycle from scheduling, via travel management to reporting. All in one global platform.

Using ATPI CrewLink™ leads to better control, greater process compliance and optimised traveler safety. With incremental cost savings as a result.

* Operational efficiencies around labor intensive, manual processes and double work
* Cost efficiencies by eliminating disparity of systems and enhanced quality control
* Optimized travel policy compliance
* Visibility of crew or traveler whereabouts and risk control
* Transparency of total cost of travel and opportunity for savings
* Improved forecasting capabilities
* Streamlined communications
* Consolidation of multiple (3rd party) systems in one single global platform
* Automated sourcing of the lowest fare available in the market in accordance with travel policy
* Automated search for the highest possible airline seat availability

# Definitions, Acronyms and Abbreviations

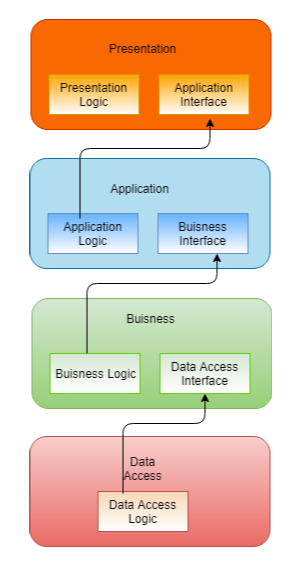
## Definitions

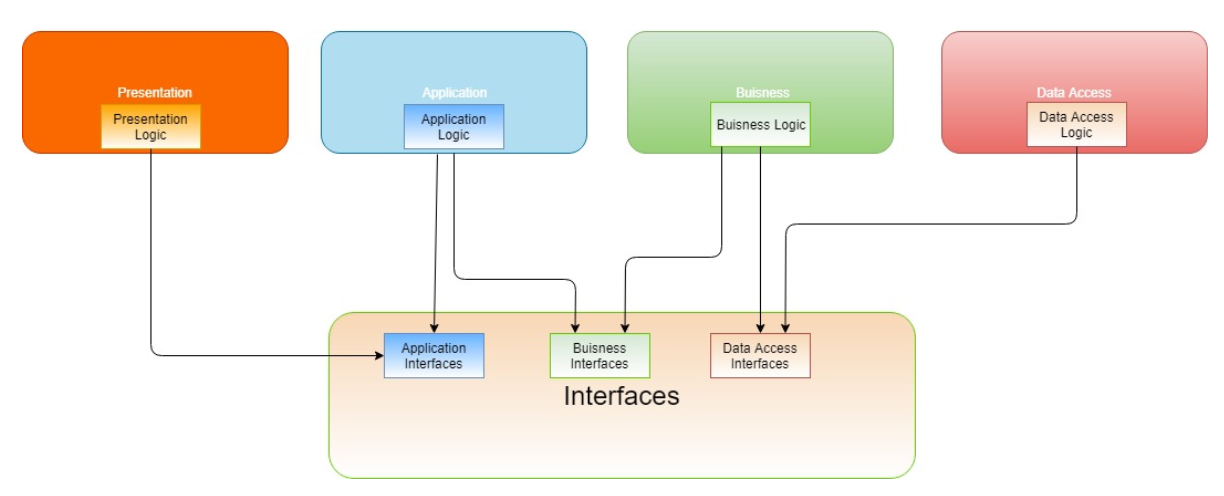
|  |  |
| --- | --- |
| **Terms** | **Definition** |
|  |  |
|  |  |

## Acronyms and Abbreviations

|  |  |
| --- | --- |
| **Acronym** | **Meaning** |
| AD | Active Directory |
| API | Application Programming Interface |
| CDN | Content Delivery Network |
| UX | User Experience |
| UI | User Interface |
| GDS | Global Distribution System |
| CL | Crew Link |
| GDPR | General Data Protection Regulation |
| PbD | Privacy by Design |
| SFC | Service Fabric Cluster |
| RG | Resource Group |
| CDS | Common Data Services |
| HBE | Homeport Booking Engine |
| CSR | Client side Rendering |
| SSR | Server side rendering |
| BDD | Behavior driven development |
| DDD | Domain Driven design |
| TDD | Test driven design |

# Design Architecture Consideration







# Development Tool

## Middleware Development

* Visual Studio 2017 Professional
* Azure Subscription
* SSMS

## Backend Development

* Sql Server 2014 and above

## Source Control

* Git HUb

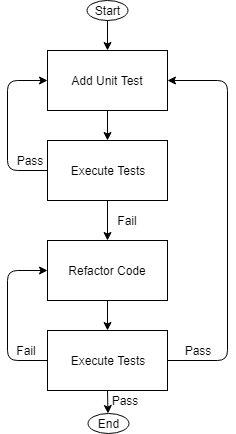
## Development Approach

* Domain Driven Development

Domain Driven Design advocates modeling based on the reality of business as relevant to our use cases. As it is now getting older and hype level decreasing, many of us forget that the DDD approach really helps in understanding the problem at hand and design software towards the common understanding of the solution. When building applications, DDD talks about problems as domains and subdomains. It describes independent steps/areas of problems as bounded contexts, emphasizes a common language to talk about these problems, and adds many technical concepts, like entities, value objects and aggregate root rules to support the implementation.

* Test Driven Development

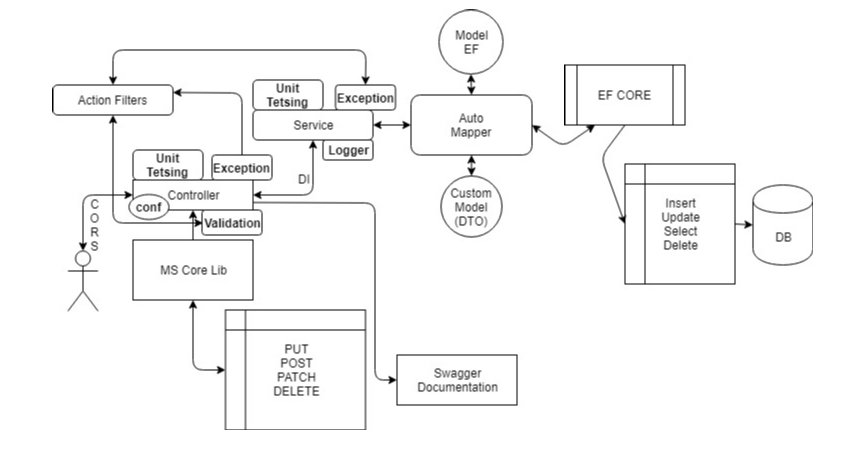
Test-driven development starts with developing test for each one of the features. The test might fail as the tests are developed even before the development.



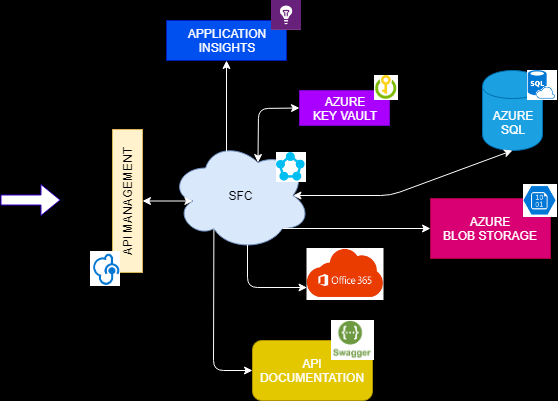
**Microservice** : , xUnit.NET - xUnit.NET has taken a pretty unique, modern, and flexible approach to unit testing. It changes terminology, so you no longer define TestFixtures and Tests...you specify Facts and Theories about your code

You are free to extend the framework to support BDD-style Concern/Context/Observation specifications

# API flow Diagram



# Microservice Technical Architecture

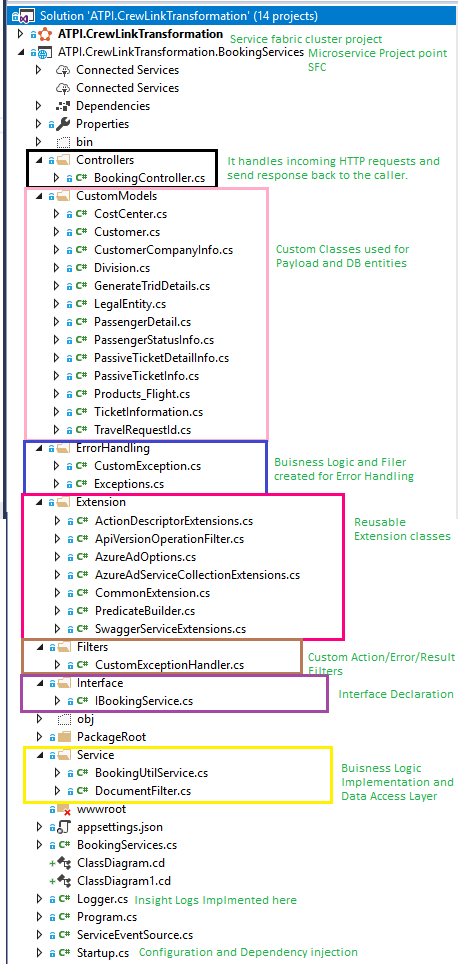


The overall source project structures for micro services and angular based front are depicted below. The non-functional requirements like logging, exceptional handling will be implemented during the development phase. The exiting ng npm modules will be employed to achieve the same in the Angular. For Microservices nLog and application insight will be employed along with generic custom exception handling for error handling.

Data security will be achieved using the Azure services for data at rest and transit. Data at rest will be employed using Azure’s Service encryption using service managed keys. Data in transit will be implemented using the best practices suggested by Azure.

Azure services will be utilized to achieve the data storage, cloud SQL and caching services. The other factors considered are below:

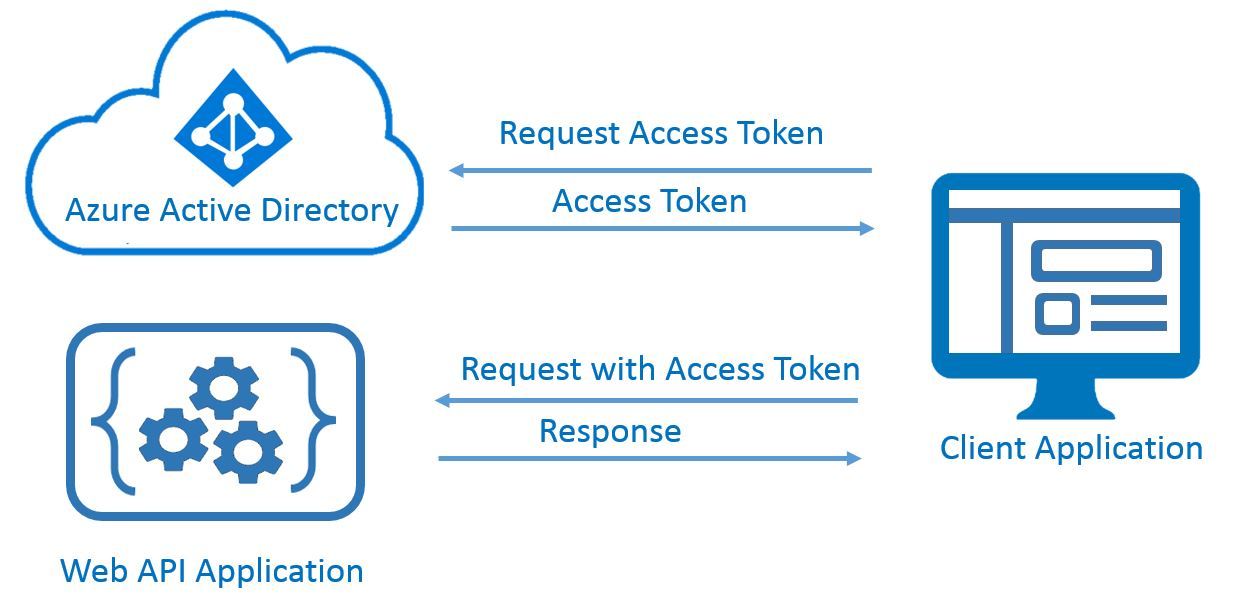
* Configuration
* Performance
* Flexibility
* Disaster recovery
* Accessibility



# Non Functional Requirement

* Security

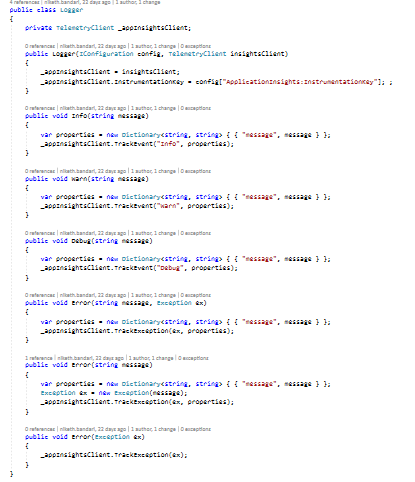
Using Azure AD Authencation and Authorization.





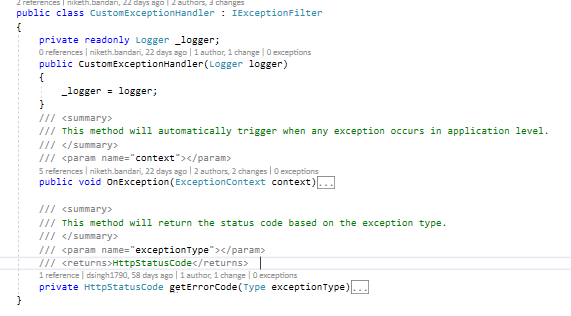
* Logging

Logging using App Insight



* Exception Handling

Using custom exception filter to handle error with proper HttpStatusCode



* Storage

Using Azure Blob storage.

* Configuration

Utilizing StartUp.cs and Appsetting.json

* Performance

Monitoring using Microsoft Diagnostic tool

* Flexibility
* Disaster recovery

Using Azure Services

* Accessibility

Using Azure Services.

# GDPR and PbD

## Privacy and Security

* Personal data identification
* Privacy classification of data for CIA (confidentiality, integrity and availability)
* Purpose of data collection
* Consent of data collection
* Categories of personal data
  + - Personnel
    - Profile data
    - Claim
    - Booking
    - Address
    - Public data
* Analysis of personal data processing
  + Storage and usage
  + Data flows
* Applications and Processes
* Third parties
* Locations
* Purpose
* Risk Assessment
* Control Definition
* Risk Prioritization

## Threat protection

* SQL Injection

Using ORM(Entity framework)

* Weak Account management

Using Azure AD Authentication and Authorization.

* Cross Site Scripting

Using encoder and configuring in ConfigureServices() method in Startup.cs

* Insecure Direct object references

Using Https Redirects in ConfigureServices() method in Startup.cs

* Security Misconfiguration

Directory Listing, Stack trace exposure, AppSetting.json configuration manager implemented.

* Sensitive data exposure

Using Azure Key vault to protect and expose the sensitive data

* Missing function-level access control

Implementing the OWASP principles

* Cross-site request forgery

CORS Middleware handles cross-origin requests implemented in StartUp.cs

* Using components with known vulnerabilities

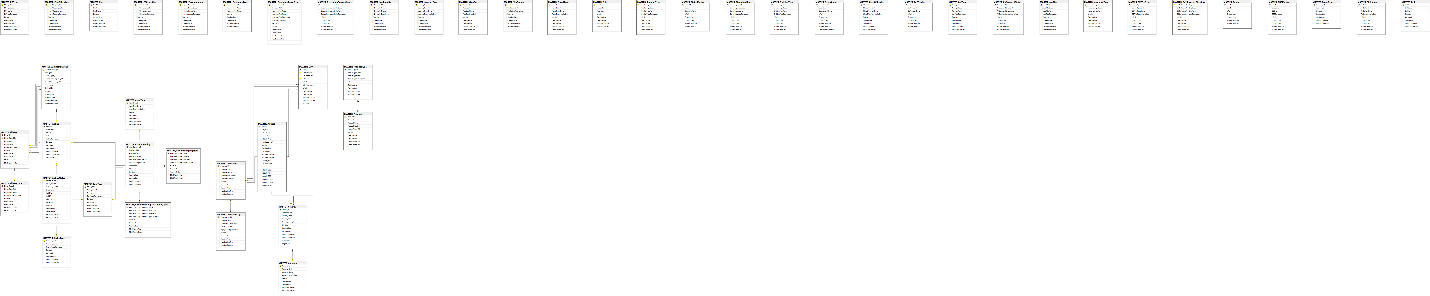
Implementing the OWASP principles

* Un-validated redirects and forwards

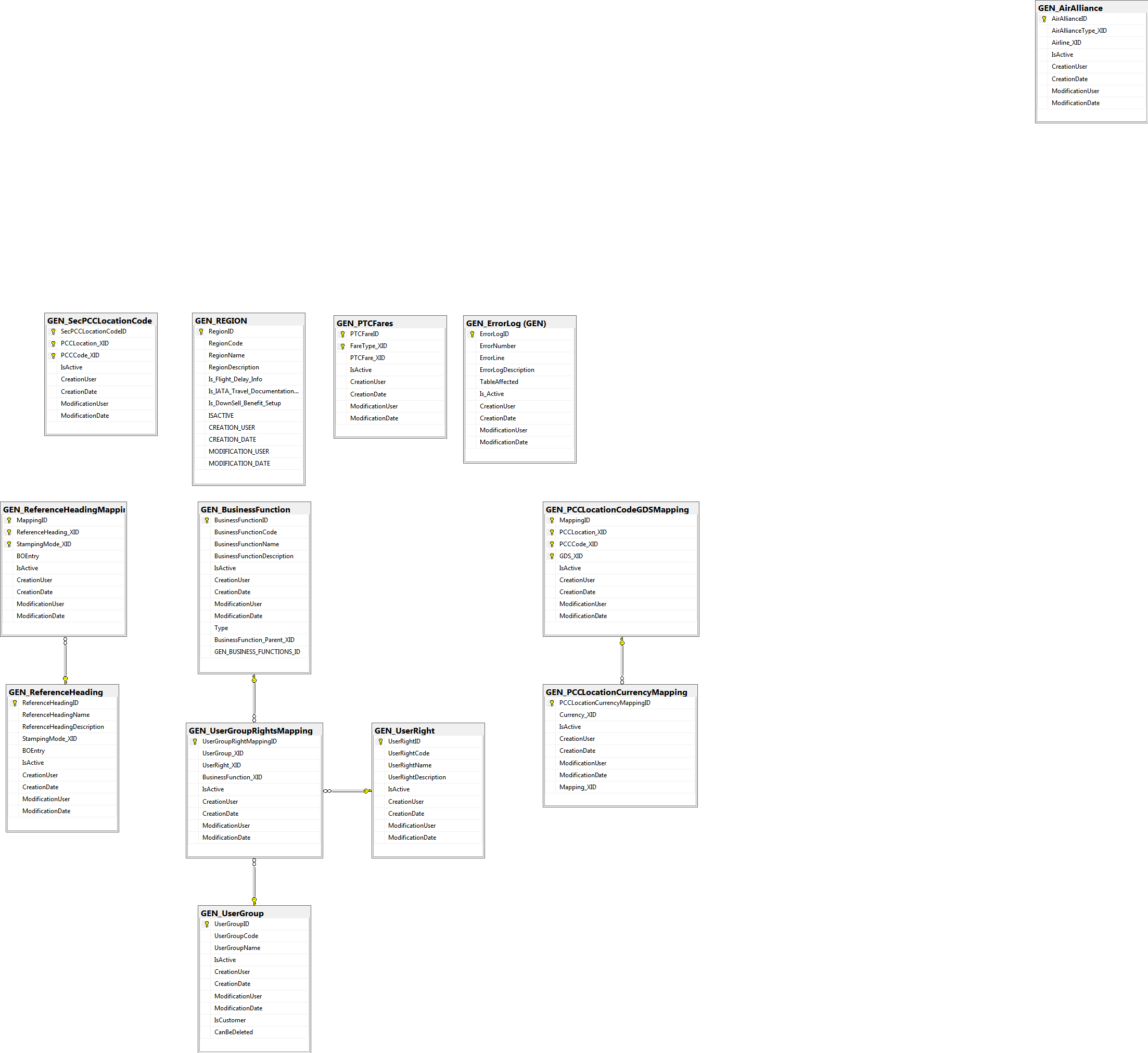
Implemented ASP.NET Core antiforgery configuration in StartUp.cs

# Database Diagram

## Master Tables

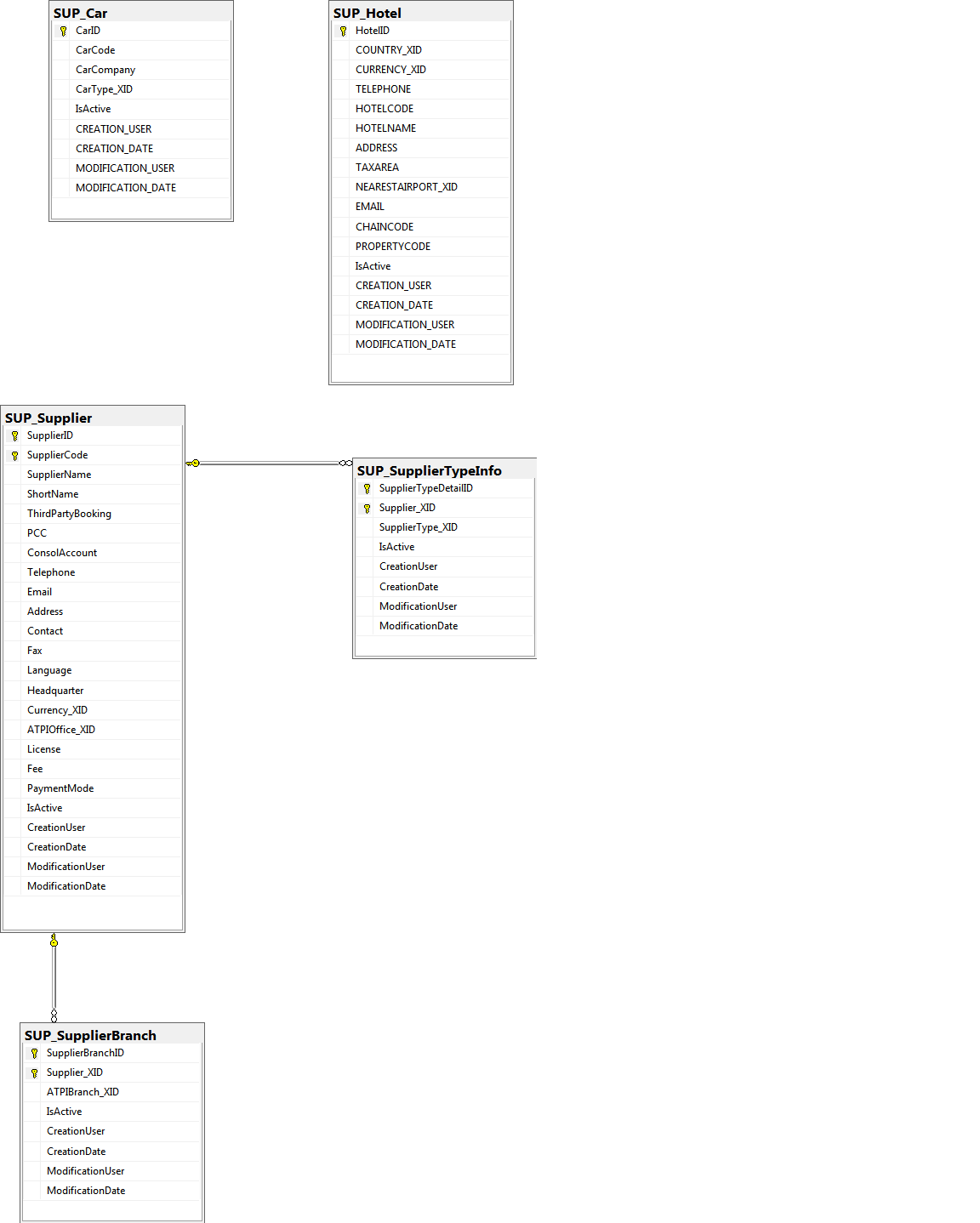


## General Set up Tables

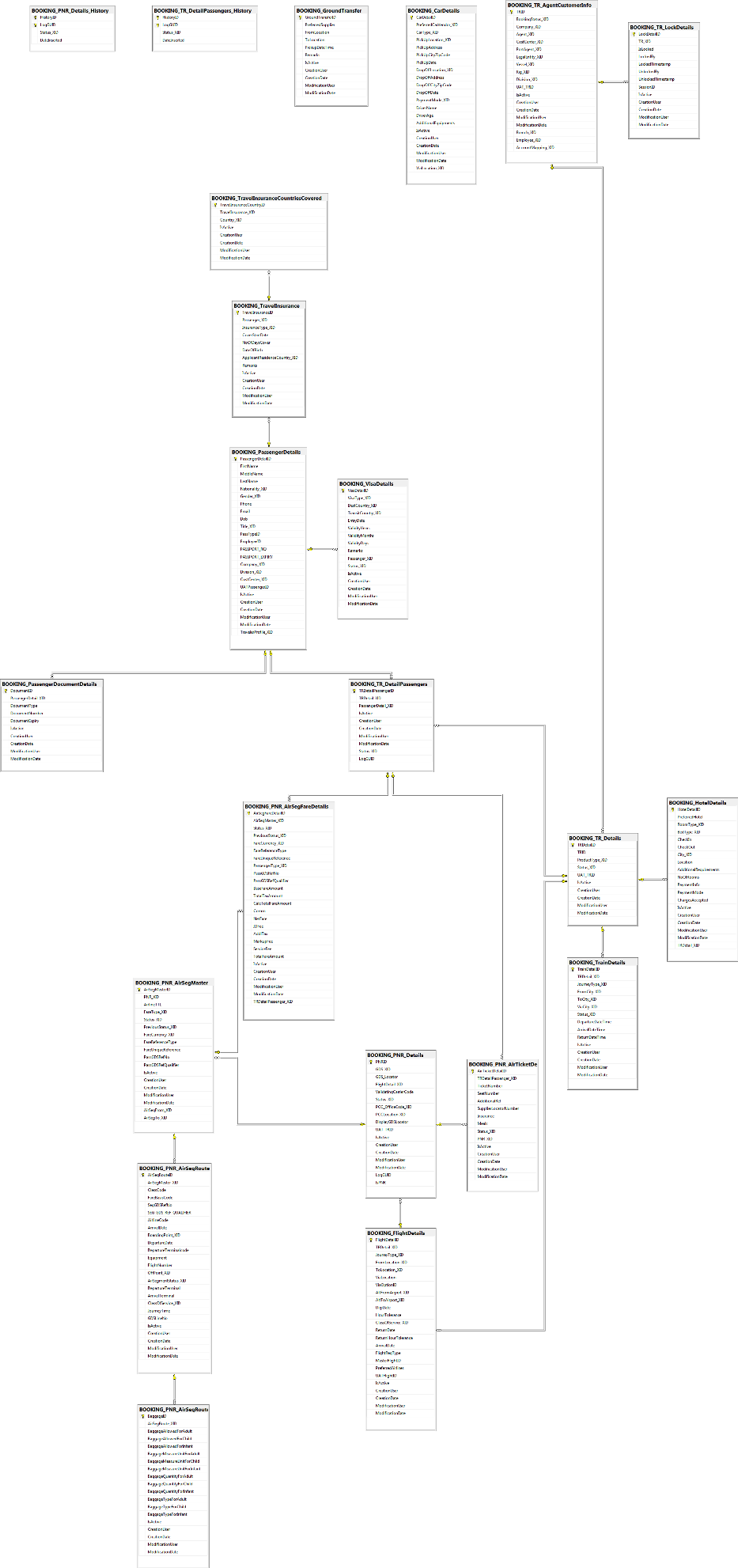


## Office Set up Tables

## Supplier Set up Tables



## Booking Management Tables



## Customer Set up Tables

