**Documentazione dello Sviluppo della Web API HotelsRegistry**

**Struttura del Progetto**

Ho iniziato la realizzazione del progetto creando una **soluzione vuota**, per poi aggiungere i diversi **layer** rispettando il pattern **Onion/Clean Architecture**. La soluzione è stata suddivisa nei seguenti progetti:

* **HotelsRegistry.Domain** (Class Library - .NET 8): Contiene tutte le entità e le logiche di dominio.
* **HotelsRegistry.Infrastructure** (Class Library - .NET 8): Contiene il DbContext, le configurazioni del database e la cartella delle migrazioni.
* **HotelsRegistry.Application** (Class Library - .NET 8): Destinato alla gestione dei servizi e della business logic.
* **HotelsRegistry.API** (Minimal API - .NET 8): Punto di ingresso della Web API, gestisce la comunicazione con il client tramite API REST.
* **HotelsRegistry.Common** (Class Library - .NET 8): Contiene utility comuni a tutti i layer per evitare dipendenze ridondanti.

Dopo aver definito questa struttura, ho **pubblicato il progetto su una repository privata** nel mio account GitHub utilizzando gli strumenti Git di Visual Studio. Per il monitoraggio dei branch e del **GitFlow**, utilizzo sia Visual Studio che **Sourcetree**.

**Sviluppo del Layer Domain**

Nel **Domain Layer**, ho definito tutte le **entità** utili al progetto.

* Ho utilizzato le **Data Annotations** per aggiungere vincoli e controlli direttamente sulle proprietà dei modelli. Questo non solo garantisce una validazione dei dati lato applicazione, ma fornisce anche un riferimento utile durante la generazione del database con Entity Framework Core.
* Ho implementato un **utility personalizzato** per semplificare la serializzazione degli oggetti, utilizzando **Newtonsoft.Json**.
* Ho introdotto un'**interfaccia** per l'accesso sicuro agli ID delle entità, che sono di tipo **GUID**. Questo garantisce un accesso strutturato e controllato ai dati senza esporre direttamente i modelli.

**Sviluppo del Layer Infrastructure**

Nel **Infrastructure Layer**, ho implementato:

* **Il DataContext** (HotelsRegistryDbContext), che gestisce l’accesso al database.
* **La configurazione della stringa di connessione** in un file statico (InfrastructureConfig.cs). Questo permette di centralizzare la gestione della connessione e di richiamarla facilmente nel Program.cs del progetto API.
* **La cartella Migrations**, dove vengono salvate le migrazioni del database generate con **Entity Framework Core**.

Un problema che ho affrontato riguarda la presenza di **due chiavi esterne (FK) che puntano alla stessa tabella** nella tabella RoomHierarchy, chiamate RoomTypeBase e RoomTypeRelated. Per evitare conflitti durante le migrazioni, ho **esplicitamente configurato le relazioni** nel DbContext con il metodo OnModelCreating.

Dopo Questi passaggi ho eseguito i seguent comandi bel promt dei pacchetti nuget:

1. **Add-Migration InitialCreate -Project HotelsRegistry.Infrastructure -StartupProject HotelsRegistry.API -OutputDir Migrations** per creare la prima migrazione
2. **Update-Database -Project HotelsRegistry.Infrastructure -StartupProject HotelsRegistry.API** per creare il database sfruttando la migrazione e la stringa di connessione.