Sprint 7 – Business Logic: Use Case Implementierung

1. Implementieren Sie die folgenden Use Cases als Business Components im Business Layer end-2-end:
    a. **Submit a new parcel to the logistics service.**
        i. Validation of parcel data
        ii. Create new unique Tracking ID
        iii. Get GPS coordinates for package sender/recipient (using Geo Encoding Agent)
        iv. Predict **future hops** (=route between sender → recipient)
        v. Set parcel state to "Pickup"
        vi. Write data to database
        vii. Return Tracking ID

    b. **Transfer an existing parcel from the service of a logistics partner.**
        i. Validation of data
        ii. Similar to Submit new parcel (1a) (except reusing existing tracking ID)

    c. **Import a hierarchy of Hops (Warehouse, Truck, TransferWarehouse) objects.**
    (Import can be called MULTIPLE times – in which case DB is completely RESET)
        i. Validation of data
        ii. Clear the existing DB (the entire one)
        iii. Write hop hierarchy to DB

    d. **Export a hierarchy of Hops (Warehouse, Truck, TransferWarehouse) objects.**
        i. Get entire hierarchy from DB and return.

    e. **Get a Hop (Warehouse, Truck, TransferWarehouse) by code**
        i. Get single hop from DB and return it.

    f. **Report Parcel arrival at hop**
        i. Validation of data
        ii. Remove the corresponding hop from future hops of the parcel
        iii. Add the hop to visited hops of parcel
        iv. Update state if parcel
        If hop is Warehouse:
        Change state to "InTransport".

        Only if hop is a Truck:
        Change state of parcel to "InTruckDelivery".

        Only if hop is a TransferWarehouse:
        Call logistics partner API - TRANSFER – which has same contract as yours.
        URL is part of TransferWarehouse data.
        (POST ***https://<partnerUrl>/parcel/<trackingId>***)
        Then change state of parcel to "Transferred" in the DB.
        v. Update the parcel back to the database

g. Report Parcel delivery at final address.
    i. Validation of data
    ii. Change state of parcel to “Delivered” in the DB.

h. Track a parcel
    i. Validation of Tracking Code
    ii. Fetch **visited hops** for parcel from DB
    iii. Predict or fetch **future hops** to final destination

2. Stellen Sie sicher, dass die Use Cases von (1) end-2-end funktionieren.
   Also: Service Layer leitet Aufruf an Business-Layer weiter und returned Ergebnis.
   Und auch der Data-Access Layer soll alle nötigen Methoden enthalten, die der Business Layer benötigt.
   Eventuell müssen fehlende Teile in allen Layers ergänzt oder angepasst werden.
   Exception Handling, Logging und andere Practices der Vor-Übungen sollen auch in den neuen Code Teilen umgesetzt werden.

3. Schreiben Sie neue Unit-Tests oder aktualisieren Sie die existierenden Tests, um bei einer Code Coverage von >=70% zu bleiben.

   Reminder: Stellen Sie nochmals sicher, dass die Code Coverage im BUILD Ergebnis aufscheint! Sie können dabei folgende Typen ignorieren
   - Alle Entities / Models / DTOs / ...
   - Validators
   - Startup.cs, Program.cs
   - Automapper Profiles
   - Custom Attributes, Custom Filters

4. Nutzen Sie die Zeit nochmals über das Projekt zu schauen, und Übung 1-6 nachzuziehen!
5. Refactoren Sie wo nötig existierenden Code, um zum Gesamtstand des Projektes zu passen.
   Z.B. sind die Interfaces, Constructor Injection, Logging, Unit Tests, Exception Handling, .. durchgängig und auch in neuen Code-Teilen umgesetzt?

6. Abgabe:
   Der MAIN Branch ihres Azure DevOps Git Repositories sollte immer den Abgabestand des Source Codes enthalten. Arbeiten Sie auf einem anderen Branch und mergen Sie auf MAIN, sobald Sie abgeben / releasen.
   Die Abgabe auf Moodle einmal pro Gruppe nicht vergessen! (Gruppen müssen auch auf Moodle eingetragen sein!)