Sprint 8 – Webhooks, Integration Tests

1. Implementieren Sie **Webhooks („rest webservice events“)** entsprechend dem Swagger Vetrag Version 1.20.2 auf Moodle.

   Die Webhook Subscriptions werden in der SQL Datenbank gespeichert. Dafür ist ein neues Repository nötig.
   Die WebhookManager kümmert sich um Versenden, Subscriben, ... von WebHooks und wird ähnlich dem SERVICE AGENT abgebildet (also mit Interface, ..). Da es sich um keine echte Business Logic handelt, sondern um Infrastruktur ist er Teil des DAL. Aber wieder in 3+ eigenen C# Projekten.

   Wann immer sich der Status eines Paketes in der Business Logic verändert, wird der WebhookManager aufgerufen und alle Subscriber benachrichtigt.

   Mit der **Final Delivery werden alle Webhook Subscriptions für das Paket** automatisch gelöscht.

   Folgen Sie den bereits bekannten Mustern: Eigene Projekte, Interfaces, Repository, ...

2. Schreiben Sie in einem eigenen Projekt INTEGRATION TESTS, die mittels HttpClient alle Usecases / REST Calls testen (beachten Sie den Unterschied zu UNIT TESTS!)
   Die Integration Tests zählen natürlich nicht zur Code Coverage!

   Die Integration Tests sollen beispielhaft eine Reise eines Paketes testen, also die Folge von Aufrufen von Userstories, z.B. Submit New Parcel, Track Parcel, ReportHopArrival, wieder Track Parcel, wieder ReportHopArrival, ReportFinalDelivery, wieder Track Parcel.

3. In Azure DevOps – RELEASE Pipelines:
   Erstellen Sie zusätzlich ein neues Environment mit Namen „**Testing**“ im Release Prozess mit eigenem 2. Webserver, eigener 2. Datenbank.
   (das existierende Environment soll „**Production**“ heißen).

   Zuerst wird auf „**Testing**“ Environment released, danach wird das RELEASE durch Ausführung der **Integration Tests** gegen den Testing-Webserver überprüft.

   Erst nach erfolgreicher Integration Tests wird mit dem „Production“ Environment fortgefahren.
   Die ConnectionStrings sollen je nach Environment unterschiedlich sein, für Testing bzw. Production Database.

4. Stellen Sie nochmals sicher, dass sie Exception Handling, Logging, Dependency Injection, Unit Testing über alle Bereiche hinweg konsistent umgesetzt haben!

5. Stellen Sie nochmals sicher, dass die Code Coverage im BUILD Ergebnis aufscheint!
   Bei der **Code Coverage (>= 70%)** können Sie folgende Typen ignorieren
   - Alle Entities / Models / DTOs / ...
   - Validators
   - Startup.cs, Program.cs
   - Automapper Profiles
   - Custom Attributes, Custom Filters
6. Abgabe:
   Der MAIN Branch ihres AZURE DEVOPS Git Repositories sollte immer das letzte „Release“, also den Abgabestand enthalten. Arbeiten Sie auf einem anderen Branch und mergen Sie auf MAIN, sobald Sie abgeben / releasen.
   Beide GruppenteilnehmerInnen müssen die Abgabe (=Angabe der Daten) auf Moodle machen!