Praxisprojekt F73

Projektbeteiligte:   
Felix Pape, Marlon Sido

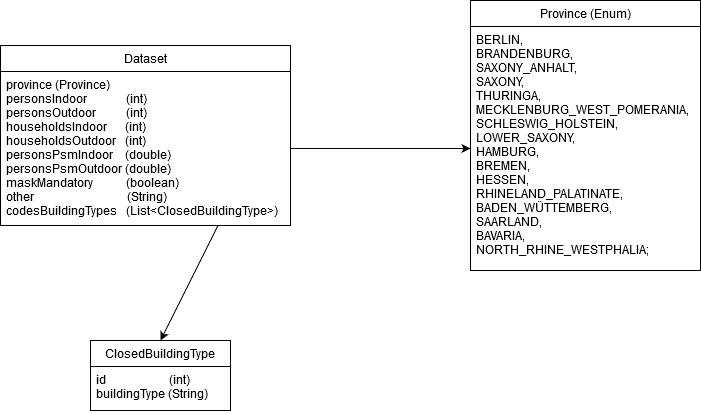
Projektbeschreibung:   
Im Rahmen des Projekts soll eine webgestützte Microservice-Anwendung realisiert werden. Diese Anwendung soll es möglich machen, in einer Weboberfläche Coronarelevante Verordnungen auszulesen, sie in eine Datenbank zu speichern und in einer weiteren Oberfläche anzuzeigen. Außerdem soll möglich sein ein geplantes Vorhaben anhand von Nutzereingaben gegen die Regularien eines ausgewählten Bundeslandes gegenzuprüfen und optisches Feedback zu erhalten.

Projektplanung:  
Den Autoren standen zur Umsetzung des Projekts 4 Tage zur Verfügung (Beginn: Montag 9:00, Abgabe: Donnerstag 17:00)  
Daraus ergab sich folgende Zeiteinteilung:  
Montagvormittag: Entwurf  
Montagvormittag-Mittwochmittag: Implementierung der einzelnen Services  
Mittwochmittag-Donnerstagmittag: Dockerisierung der Services und Fehlerbehebung  
Donnerstagmittag-Donnerstagnachmittag: Weitere Fehlerbehebung und Dokumentation

Entwurf:   
Der an die erstgenannte Oberfläche angebundene Service, soll lediglich schreibende Zugriffe auf die Datenbank erhalten. Der zweite Service, der an die für die Öffentlichkeit zugängliche Oberfläche angebunden ist, soll lediglich aus der Datenbank lesen können.   
Bei der abzuspeichernden Datenstruktur wurde sich nach Absprache mit dem Trainer auf folgendes geeinigt: Es gibt 3 Tabellen.

TLS steht für Transport Layer Security und wird von Browsern für sichere HTTPS-Verbindungen genutzt. Da für die Authentifizierung jedoch Zertifikate benötigt werden, die Zeit der Projektumsetzung begrenzt ist und unser Projekt lediglich lokal und nicht produktiv laufen wird, wurde auf TLS verzichtet.

Die Haupttabelle beinhaltet folgende Spalten: (BuildingType) des Typs String zugeordnet wird.



Diese Festlegungen sind sehr wichtig um anschließend den Datenaustausch der einzelnen Mikroservices über HTTP-requests in Form von JSON-objekten so einfach wie möglich zu gestalten.

Implementierung:  
Die beiden Weboberflächen werden nach Absprache mit dem Trainer entgegen der Vorgabe statt mit Spring Boot und Thymeleaf stattdessen in Form eines Angular-Frontends umgesetzt, da Marlon bereits Erfahrungen damit sammeln konnte, außerdem ist die Technik moderner und austauschbarer.

Implementierung: Es wurde damit begonnen, ein Git-Repository mit passender Ordnerstruktur für das Frontend und Backend sowie die Dokumentation erstellt.  
Anschließend hat Felix angefangen, den Exportservice mithilfe von Spring Boot, Spring Web-, DataJPA und der Lombok dependency umzusetzen.

Im Zuge dessen wurde eine Datenbankverbindung zum testen der Grundfunktionen eine H2-Datenbank konfiguriert.   
Im weiteren Projektverlauf wurde diese dann gegen eine MySQL-Datenbank ausgetauscht, mit welcher anschließend sowohl Import- als auch Exportservice kommunizieren sollen.

Marlon hat zunächst die beiden Projekte für das Import- und Exportfrontend eingerichtet, Angular (material) und Bootstrap installiert.  
Im Anschluss wurde die Loginseite mit dem User "admin" und dem passwort "admin" umgesetzt.

Das auf den Login folgende Dashboard bekam zunächst eine Dropdownliste von verschiedenen Bundesländern und einen Button um einen Http-Request zu senden, mit dem das ausgewählte Bundesland geupdated werden kann, sowie einen Button um alle auf einmal zu updaten.

Darauf folgte die Umsetzung des Hauptfensters des Exportfrontends mit einer Dropdownliste der Bundesländer, eines Buttons um die Daten des jeweiligen Bundeslandes zu Laden und alle Daten der dort gültigen regularien anzuzeigen.

Um die Anforderungen an das Exportfrontend vollständig abzudecken wurde außerdem die Eingabe implementiert, die ein geplantes Event prüft und optisches Feedback gibt.

Zum Abschluss des Frontends blieben nun lediglich Stylinganpassungen und Überarbeitungen der HTTP-Requests.

Beim anschließenden Zusammenführen von Front- und Backend mithilfe von Dockerfiles sowie docker-compose zeigten sich diverse Fehler, die unter anderem auf Objektungleichheiten im Front- und Backend zurückzuführen waren.

Lessons learned:  
-Von Anfang an Tests schreiben. Das kostet zwar Zeit, diese Zeit kann aber unter Umständen sehr schnell wieder durch Zeitersparnis bei der Fehlerbehebung gutgemacht werden

-Noch bessere Absprache der Schnittstellen und Modelle. Verhindert Probleme durch Abweichungen in der Implementation

-Viele Augen-Prinzip: Manche Fehler an denen viel Zeit verloren geht, lassen sich schneller durch jemanden lösen, der diese mit einem neuen Blickwinkel betrachtet

Ausblick:

Anfänglich vergessen wurde die Festlegung des Zeitpunkts der letzten Aktualisierung der Verordnungen eines Bundeslandes. Als dies bemerkt wurde, wurde die Anforderung nach hinten gestellt und konnte aufgrund mangelnder Zeit nichtmehr implementiert werden.

Bewusst nach hinten gestellt wurde die gzip-codierung der Antwort des Exportbackends an die Öffentliche Website.