Praxisprojekt F73

Marlon Sido, Felix Pape

Corona-Verordnungen

Volkswagen AG | [Firmenadresse]

**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

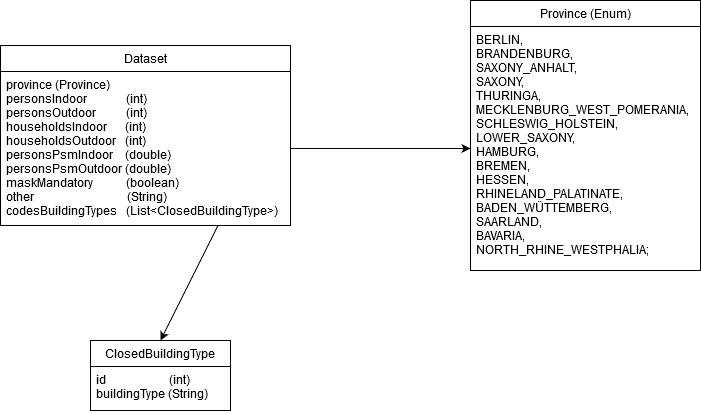
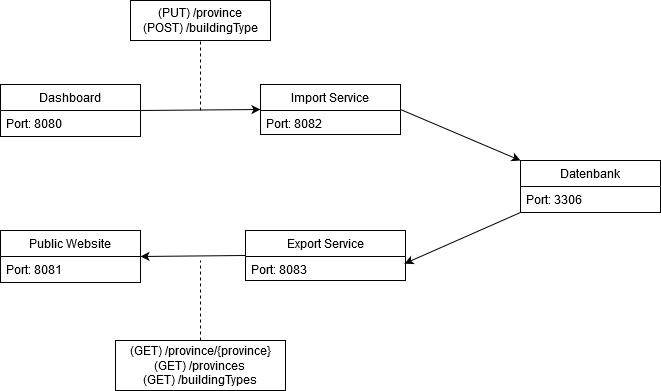
**Entwurf:**   
  
Der an die erstgenannte Oberfläche angebundene Service, soll lediglich schreibende Zugriffe auf die Datenbank erhalten. Der zweite Service, der an die für die Öffentlichkeit zugängliche Oberfläche angebunden ist, soll lediglich aus der Datenbank lesen können.   
Bei der abzuspeichernden 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.

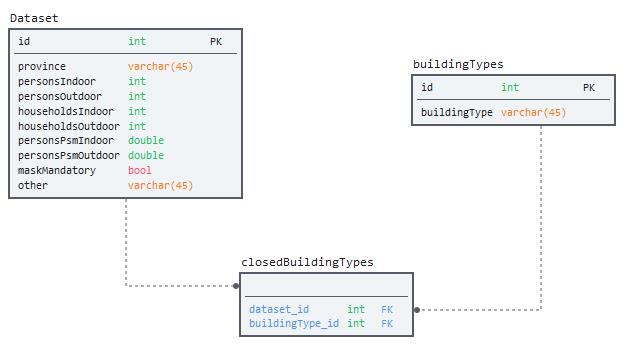
Da in beiden Frontends alle Daten zum Speichern in die Datenbank sowie für die Evalutation von Zusammenkünften gebraucht werden, bewegt sich unsere gesamte Applikation innerhalb eines Bouded Context. Würden die Daten im Dashboard jedoch mehr Informationen enthalten, als unmittelbar für die Anzeige und Evaluation öffentlichen Frontend benötigt, ließe sich dafür ein weiteres Domänenmodell und somit ein separater Bouded Context entwerfen. Dies ist einer der Gründe weshalb es ungünstig wäre, die öffentliche Website direkt auf das Dashboard zugreifen zu lassen. Weil der Nutzer sonst Zugriff auf mehr Daten bekommen würde, als für ihn nötig. Außerdem ist um die persistenz von Daten zu gewährleisten, sowie performante Abfragen zu erhalten, eine Datenbank von großem Vorteil. Des weiteren gewährleisten die weiteren Schichten der Anwendung zusätzliche Sicherheit für die Daten des Gesundheitsamts. Zudem würde der Webservice des Gesundheitsamtes so bedeutend mehr ausgelastet werden und ein Loadbalancing wäre nicht möglich.

Vorteile des direkten Zugriffs auf das Dashboard wären ein geringerer Entwicklungsaufwand der Software, sowie ein geringerer Ressourcenverbrauch, da lediglich ein weiterer Service laufen würde.

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.



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.  
Die Umsetzung der Spring Boot Export Backends hat Felix mit der Erstellung der Web- und Datacontroller begonnen. Darauf folgte die Erstellung der PersistenModels sowie der Repository Klassen. Konkret wurde das Crud-Repository benutzt.   
Zuletzt wurden die Funktionen für get Requests zum Abfragen der Informationen für ein Bundesland implementiert.

Anschließend hat Felix ein Projekt für das Import Backend erstellt und die Models auf dem Export Backend übernommen, die Web- und Datacontroller erstellt sowie die PersistentModels und die Repository Klassen übernommen.  
Nun wurden Put-Requests für die Bundesländer sowie das hinzufügen von Gebäudetypen in Form von Post-Requests implementiert.

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.

Durch initiales Testen zeigte sich eine fehlerhafte Portbelegung, das Importfrontend beispielsweise hat HTTP-Requests an sich selbst, da der Javascriptcode lokal beim Anwender ausgeführt wird und sich somit nichtmehr innerhalb des Dockernetzwerks befindet.  
Dies wurde doch eine Änderung bei den Portfreigaben behoben.

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.

Zunächst wurde versucht die Fehler nach dem Start des docker-compose zu beheben. Da dies den build-Prozess beinhaltet, kostete dies sehr viel Zeit und es wurde begonnen Tests zu implementieren.

**Beschreibung der Services:**Dashboard:  
Das Dashboard ist der Zugriffspunkt zur Verwaltung der Anwendung. Dort können nach erfolgreichem Login Aktualisierungen der Daten vorgenommen werden. Dies erfolgt indem der Anwender die Update-Funktion für das jeweilige Bundesland ausführt. Dadurch werden automatisch die entsprechenden Daten an den Importservice gesendet und dort weiter verarbeitet.

Importservice:  
Der Importservice erhält aktualisierte Daten vom Dashboard. Diese werden anschließend den entsprechenden vorhanden Daten zugeordnet, aktualisiert und in die Datenbank geschrieben.

Exportservice:  
Der Exportservice erhält Anfragen von der Öffentlichen Website. Je nach Anfrage holt dieser die angeforderten Informationen aus der Datenbank und reicht diese zurück an die Website.  
Dies sind entweder eine Liste aller Bundesländer, alle Informationen die es zu einem Bundesland gibt oder eine Liste an möglichen gesperrten Gebäudetypen.

Public Website:  
Auf der Öffentlichen Website kann sich jeder die für ein Bundesland geltenden Verordnungen nachschauen und ob seine geplante Veranstaltung den aktuellen Vorgaben des Bundeslandes entspricht. Dies geschieht indem man zuerst ein Bundesland auswählt und die Daten aktualisiert (holt die Daten vom Exportservice). Anschließend kann man die Eckdaten seiner Veranstaltung angeben um angezeigt zu bekommen, ob diese den Regularien entsprechen.

**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

-Javascript wird interpretiert, nicht compiliert. Dies hat zur Folge, dass lokale Portfreigaben nicht funktionieren, da sich der Ausführende nicht im Netzwerk befindet. Ideallösung -> Reverse-Proxy

**Ausblick:**Zum Zeitpunkt der Abgabe ist nur die Funktionskette vom Dashboard bis zur Datenbank bedingt funktionsfähig. Nach dem ersten Ausführen von docker-compose up gibt es aktuell noch einen Fehler, da die Java-Anwendungen zu schnell hochgefahren sind und die Datenbank nicht finden. Außerdem ist das Updaten der Daten vom Improtfrontend zur Datenbank nur einmalig möglich, danach fliegt ein Internal Server Error. Das Exportieren auf die Öffentliche Website bereitet aktuell noch einige Schwierigkeiten, dies zu beheben wäre unser nächster Schritt.

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, da diese für einen Prototypen von mangelnder Bedeutung ist und zugleich einen erheblichen zeitlichen Rahmen beanspruchen würde.