TAU INF202 Software Engineering
Individuelles Projekt

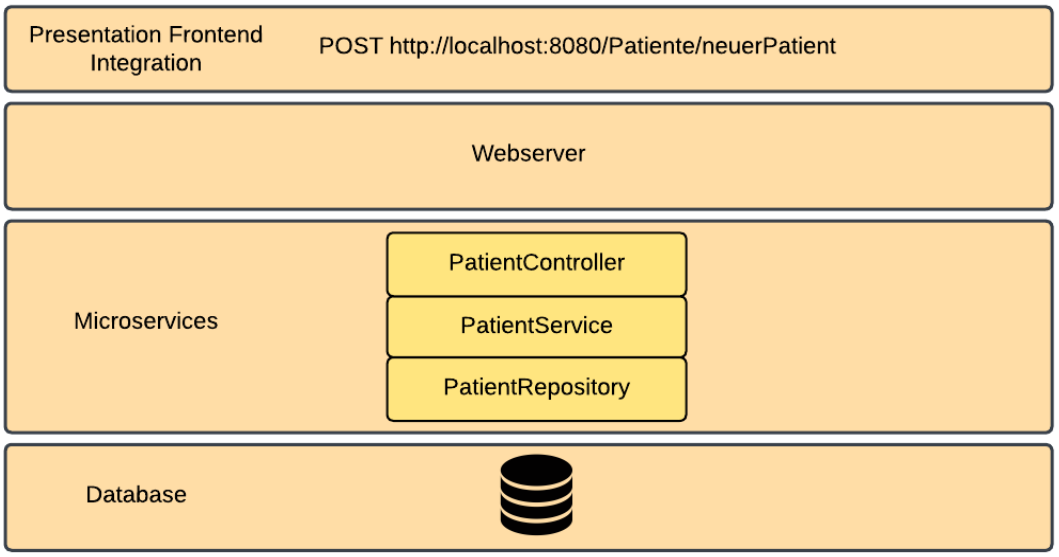
Klinikverwaltungssystem
Architekturspezifikation

Verantwortliche/r:
İpek Yılmaz, e200503052@stud.tau.edu.tr
Haluk Harun Gündoğan, e200503031@stud.tau.edu.tr

Stakeholder: DI. Ömer Karacan, omer.karacan@tau.edu.tr

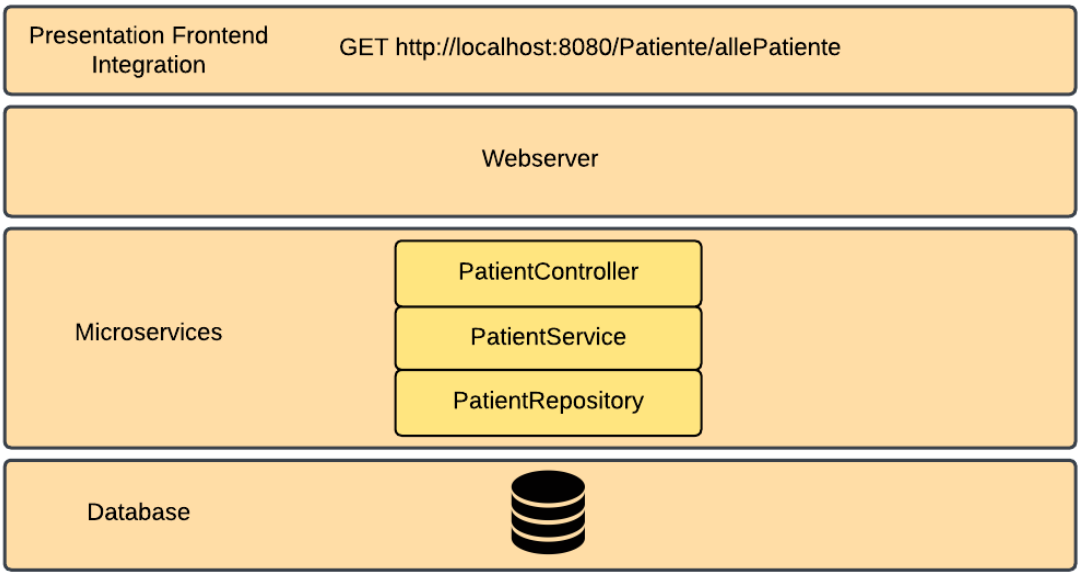
1. Architekturüberblick

1.1. Use Case ,,Patiente einfügen”



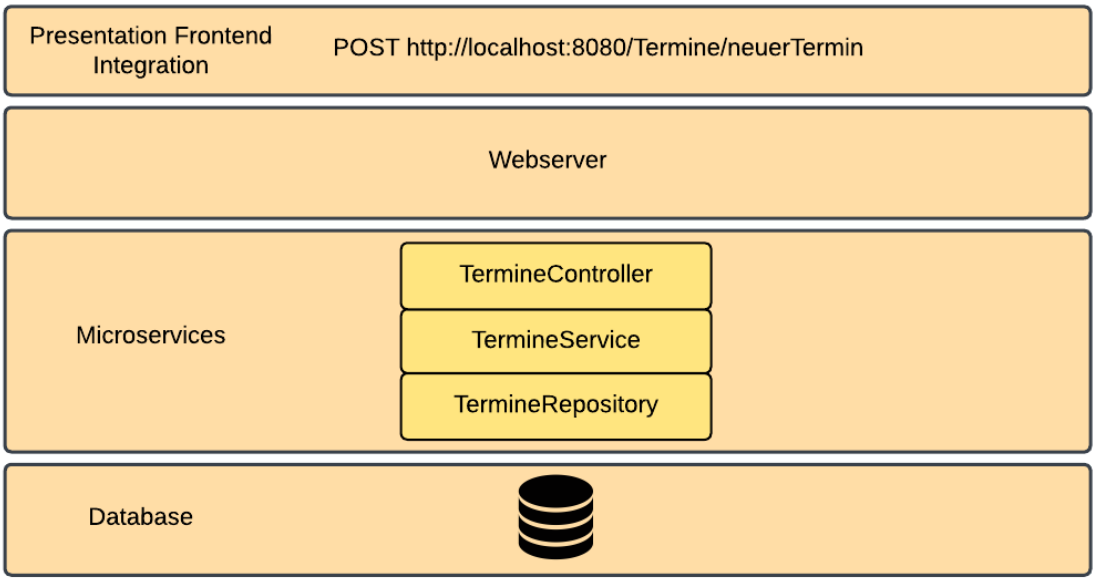
Über die PatientController Klasse bekommt die PatientService Klasse, wo die Geschäftslogik implementiert wird, die gesendeten Informationen. In dieser Klasse fügt die Methode ,,addNewPatient( )” Patienten ein und kontrolliert, ob die gegebenen Informationen im Rahmen der Geschäftshaftlogik gültig sind. Wenn die gegebenen Informationen gültig sind, werden die Patienteninformationen über das PatientRepository in der Datenbank gespeichert.

1.2. Use Case ,,Patiente auflisten”



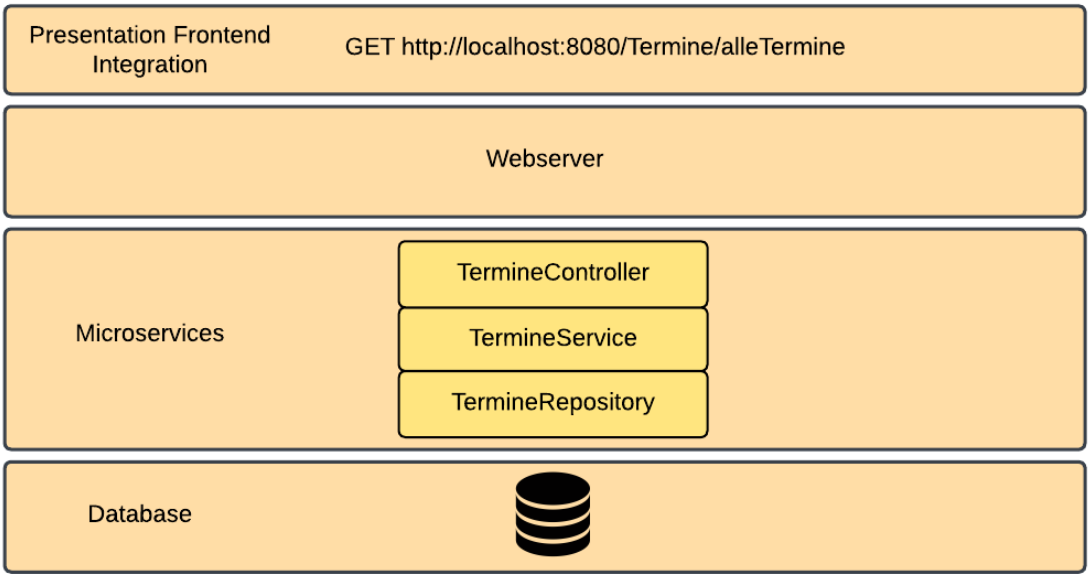
Über die PatientController Klasse bekommt die PatientService Klasse die gesendeten Informationen. In dieser Klasse listet die Methode ,,getPatients( )” Patienten auf. Diese Methode benutzt die PatientRepository, um die Patienteninformationen aus der Datenbank zu bekommen.

1.3. Use Case ,,Termine Erstellen”



Über die TermineController Klasse bekommt die TermineService Klasse die gesendeten Informationen. In dieser Klasse erstellt die Methode ,,erstellNeuenTermin( )” einen neuen Termin und kontrolliert, ob die gegebenen Informationen im Rahmen der Geschäftshaftlogik gültig sind. Wenn die gegebenen Informationen gültig sind, werden die Termineinformationen über das TerminRepository in der Datenbank gespeichert.

1.4. Use Case ,,Termine Auflisten”



Über die TermineController Klasse bekommt die TermineService Klasse die gesendeten Informationen. In dieser Klasse listet die Methode ,,getTermine( )” Termine auf. Diese Methode benutzt die TermineRepository, um die Termineninformationen aus der Datenbank zu bekommen.

2. Beschreibung der „Controller” Klassen

2.1. PatienteController

```java
@RestController
@RequestMapping(path="/Patiente")

public class PatientController {
   private final PatientService patientService;

   @Autowired
   public PatientController(PatientService patientService) {
       this.patientService = patientService;
   }

   @GetMapping(path="/allePatiente")
   public List<Patient> getPatients() {
       return patientService.getPatients();
   }
   @PostMapping(path="/neuerPatient")
   public void registerNewPatient(@RequestBody Patient patient){
       patientService.addNewPatient(patient);
   }

}
```

In dieser Klasse verwaltet unser System die Web Requests, indem es die Controller-Klasse für alle Komponenten(Doktor, Patient und Termine) benutzt. In unserer Implementation haben wir zwei Web Services für die Patiente Klasse, nämlich Patiente Einfügen und Patiente Auflisten. Für die Einfügeoperation werden die Patienteninformationen im JSON Format über entsprechende API-Endpoint gesendet. Zum Beispiel in unserem Pflichtenheft wurde es spezifiziert, dass für die Einfügeoperation die Ausweisnummer, Name, Nachname und Phonenummer benutzt wird. Die erwähnten Informationen werden wie in dem Beispiel gesendet:

[{"ausweisnummer":39271172908,
"name":"Merve",
"nachname":"Can",
"phone number":"05417897890"}]

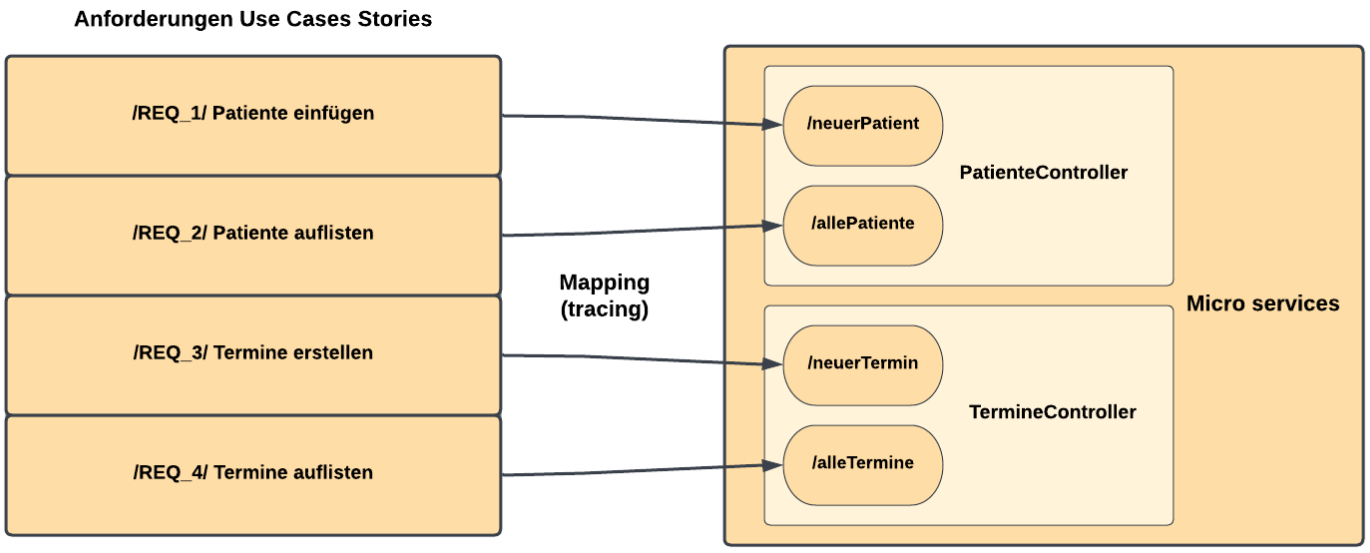
Für das Auflisten der Patienten werden die Patienteninformationen über die entsprechende API-Endpoint bekommen. Dann erhält der Benutzer alle Patienteninformationen.

2.2. TermineController

```
@RestController
@RequestMapping(path="/Termine")
public class TermineController {
   private final TermineService termineService;

   @Autowired
   public TermineController(TermineService termineService) {
       this.termineService = termineService;
   }

   @GetMapping(path="/alleTermine")
   public List<Termine> getTermine() {
       return termineService.getTermine();
   }

   @PostMapping(path="/neuerTermin")
   public void ertellNeuenTermin(@RequestBody Termine termin) {
       termineService.erstellNeuenTermin(termin);
   }

}
```

Für die Terminerstellung und Auflisten wird die TerminController Klasse benutzt. Diese Klasse bekommt die Web Requests und sendet die Request bzw. Requestköpfer zu dem TerminService Klasse. Für das Auflisten der Termine werden alle aktuellen Termine mit Hilfe der TermineRepository erhalten und aufgelistet. Für die Erstellung wird wieder das JSON Format benutzt, indem die Doktorinformationen, Patienteninformationen, Datum des Termins und Zeit des Termins im JSON Format über die spezifizierte API-Endpoint gesendet werden.

3. Rückverfolgbarkeit der Anforderungen



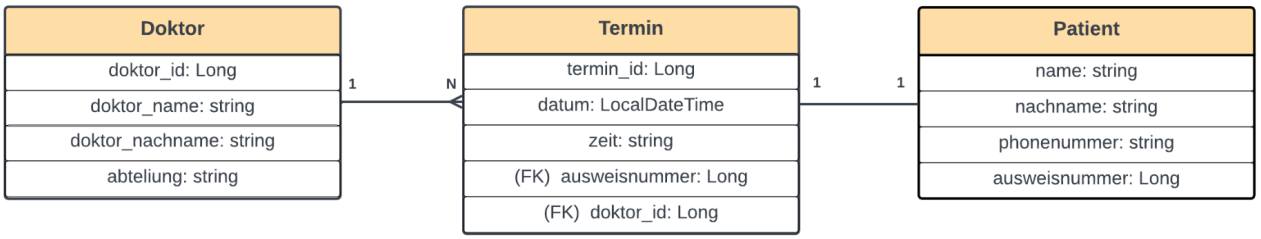
Die zwei Use Cases ,,Patiente einfügen” und ,,Patiente auflisten” werden von PatienteController verwaltet.

- Die Use Case ,,/REQ_1/ Patiente einfügen” wird mit Hilfe der in der Klasse definierten Post Mapping ,,/neuerPatient” geschafft.
- Die Use Case ,,/REQ_2/ Patiente auflisten” wird mit Hilfe der in der Klasse definierten Get Mapping ,,/allePatiente” geschafft.

Die zwei Use Cases ,,Termine erstellen” und ,,Termine auflisten” werden von TermineController verwaltet.

- Die Use Case ,,/REQ_3/ Termine erstellen" wird mit Hilfe der in der Klasse definierten Post Mapping ,,/neuerTermin” geschafft.
- Die Use Case ,,/REQ_4/ Termine auflisten” wird mit Hilfe der in der Klasse definierten Get Mapping ,,/alleTermine” geschafft.

4. Beschreibung der DB-Zugriffsschicht(Daten-Modelle)



In unserer Datenbank werden drei Tabellen namens Doktor, Termin und Patient gespeichert. Wie in dem Diagramm zu sehen ist, speichert die Doktortabelle die Doktorinformationen bzw. Doktor Name, Nachname, Abteilung und ID von Doktor. Obwohl der Name, Nachname und die Abteilung zu der Stringklasse gehört, ist die ID Long und es ist auch zu merken, dass die ID vom Doktor sequenziell bei dem Backend generiert wird. Diese Eigenschaft trägt auch die ID der Termine. Außerdem in der Termintabelle speichert man sowohl das Datum(TT/MM/JJJJ) der Termine und Zeit der Termine als auch die Foreign Keys, die für das Mapping zwischen den Tabellen verantwortlich sind. Die Patienttabelle macht die Patienteninformationen persistent, indem sie die Patientausweisnummer als Primärschlüssel benutzt. Außer der Ausweisnummer, speichert die Datenbank der Patient Name, Nachname und die Telefonnummer.