Dokumentation: Kinoticketreservierungsapp

Gruppe: Adrian, Aleks, Sam, Justin, Marcel E., Dennis, Fabian

Gliederung:

- Motivation und Hintergrund
  - Welche Probleme behebt die App
  - Für wen ist die App (Userstories)
- Architekturrelevante Anforderungen
  - Aufbau der App (UML-Diagramm)
- Lösungsansätze und eingesetzte Tools/Sprachen
  - Dantenbank
  - Frontend
  - Backend
- Testabdeckung
  - Code-Coverage im Überblick
- Ausblick und weitere Informationen
  - Gibt es in der Zukunft neue Erweiterungen

Welche Probleme behebt die App

Personas	Karl Kartenkontrolleur	Karsten Kinobesitzer	Anton Administrator	Volker Verkäufer	Timona Ticket	Gustav Großkunde
						
Goals	Karten beim Einlass kontrollieren, Platz ungefähr zuweisen	Statistiken einsehen können	Kinoräume bearbeiten, Website gestalten etc.	Kunden Auskunft geben können	Ticket einfach kaufen und stornieren	Sehr viele Plätze buchen, evtl. ganzes Kino
Pain Points	Weiß nicht alles über die Säle.	Verwaltung ist kompliziert	Schlechte Performance der Website	Vergleich von Tickets soll möglich sein	Website schlecht gestaltet	Zu wenige Tickets
Skills	Niedrige IT Kenntnisse, Karten abscannen	Einfache IT Kentnisse	Hohe IT-Kentnisse	Mittelmäßige IT-Kentnisse	Niedrige IT-Kentnisse	Mittlere IT-Kenntnisse
Alter	25	70	32	38	62	58

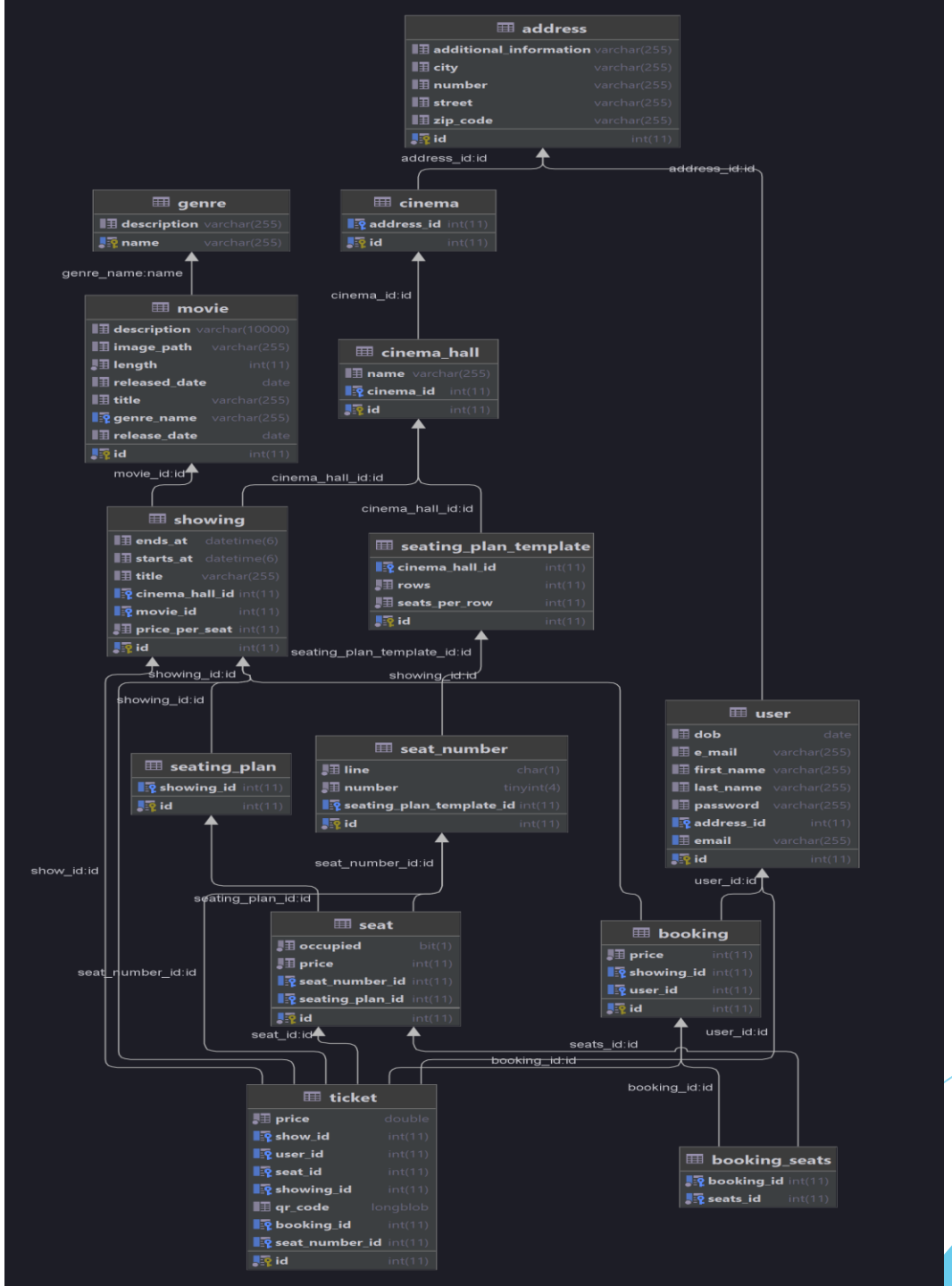
Für wen ist die App (Userstories)

Folgende Personas wurden umgesetzt:

- Karl Kartenkontrolleur
- Volker Verkäufer
- Timona Ticket
- Gustav Großkunde

- Die App ist für Kartenkontrolleur
- Webseite ist für Kunden
- Man kann Tickets reservieren/kaufen

Aufbau der App (UML-Diagramm)



Lösungsansätze und eingesetzte Tools/Sprachen

Datenbank

- MySQL
- Open Source DBMS
- Leichte Verknüpfung mit Azure und Spring

Backend

- SpringBoot
- Open Source Framework
- Wird zum Erstellen von Webanwendungen und Mikroservices verwendet

Backend: Projektstruktur am Beispiel der Movie-Entität

- Definition der Entität:

```java
@Entity
public class Movie {
    2 usages
    @Id
    @GeneratedValue(strategy
            = GenerationType.AUTO)
    private Integer id;
    6 usages
    private String title;
    4 usages
    private String imagePath;
    4 usages
    private String description;
    4 usages
    private int length;
    4 usages
    private LocalDate releaseDate;
    6 usages
    @ManyToOne
    @JoinColumn(name = "genre_name", referencedColumnName = "name"
    private Genre genre;
```

Backend: Projektstruktur am Beispiel der Movie-Entität

Repository für die Entität:

```
public interface MovieRepository extends CrudRepository<Movie, Integer> {
    3 usages  Adrian Hagen
    Iterable<Movie> findByGenreName(String genre);
}
```

Backend: Projektstruktur am Beispiel der Movie-Entität

Controller für die Entität:

```java
@RestController
@RequestMapping(path = "/movie")
public class MovieController {
    5 usages
    @Autowired
    private MovieService movieService;

    2 usages  Adrian Hagen +1
    @PutMapping("/add")
    public ResponseEntity<Object> addMovie(@RequestBody MovieRequest movieRequest){
        try {
            return new ResponseEntity<>(movieService.addMovie(movieRequest), HttpStatus.ACCEPTED);
        }catch (InvalidRequestException invalidRequestException){
            return new ResponseEntity<>(invalidRequestException.toString(), HttpStatus.BAD_REQUEST);
        }
    }
```

Backend: Projektstruktur am Beispiel der Movie-Entität

Request-Klasse für die Entität:

```
public class MovieRequest {
    2 usages
    private final String title;
    2 usages
    private final String imagePath;
    2 usages
    private final String description;
    2 usages
    private final int length;
    2 usages
    private final String releaseDate;
    2 usages
    private final String genreName;
```

Frontend

- Angular
- JavaScript-Web-Framework zur Erstellung von Single-Page-Webanwendungen

Frontend: Projektaufbau am Beispiel der Sitze

Verbindung mit dem Backend

```
readonly URL: string = 'https://cinema-backend-group1.azurewebsites.net/seat/all';

httpOptions = {
  headers: new HttpHeaders({
    'Content-Type': 'application/json',
    'Authorization': 'ApiKey MWYeGwzOEJ2WDF0Qm3ua3hhYWw6Z2piWVFoMUfRT0NxS2k5RlhXdzdPQQ=='
  })
};
```

Frontend: Projektaufbau am Beispiel der Sitze

```
connect(): Observable<any> {
  return this.http.get<any>(this.URL, this.httpOptions);
}


getPost() {
  this.sendGet().subscribe(data => {
    for(let i = 0; i < data.length; i++) {
        this.seatsLine[i] = data[i].seatNumber.line;
      }
      this.seatsLine.sort();
      this.sortAlgString();

    for(let i = 0; i < 4; i ++) {
        this.seatsline[i] = this.seatsL[i];
      }
    for(let i = 0; i < 10; i++)  {
        this.seatsNumber[i] = data[i].seatNumber.number;
      }
      this.sortAlg();
  });
}
```

Frontend: Projektaufbau am Beispiel der Sitze

```
sendPost() {
  let num: any;
  let line: string;
  for(let i = 0; i < this.selected.length; i++) {
   line = this.selected[i].substring(0,1);
   num = this.selected[i].substring(1,2);
   this.body[i] =  {
      "seatNumber": {
        "id": this.selected[i],
        "line": line,
        'number': num
      },
      "price": "",
      "occupied": "true"
    }
  }
  return this.http.post(this.URL, this.body, this.httpOptions);
}
```

Frontend: Projektaufbau am Beispiel der Sitze

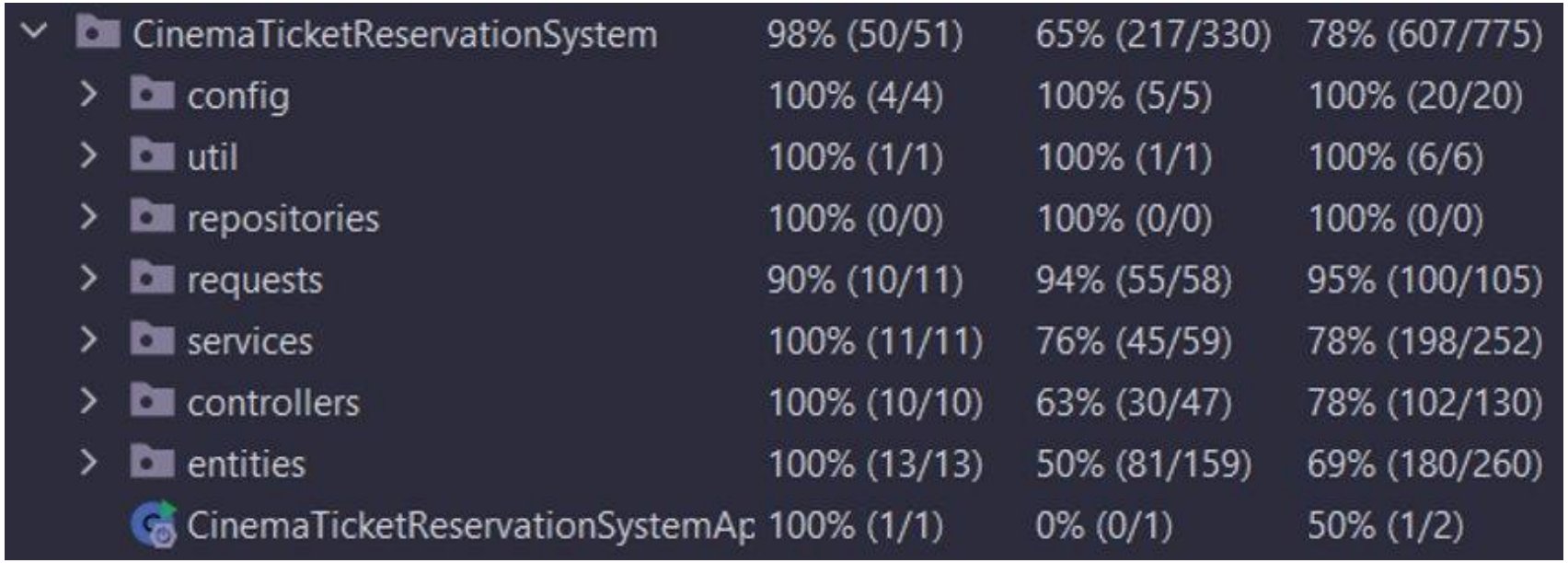
```
seatClicked = (seatPos: string) => {
  var arrIndex = this.selected.indexOf(seatPos);

  if (arrIndex !== -1) {
    this.color = 'white';
    this.selected.splice(arrIndex, 1);
  } else {
    if (arrIndex === -1) {
      this.color = 'black';
      this.selected.push(seatPos);
    }
  }
};
```

Frontend: Projektaufbau am Beispiel der Sitze

```
getSeatState = (seatPos: string) => {
  if (this.reserved.indexOf(seatPos) !== -1) {
    return 'reserved';
  } else if (this.selected.indexOf(seatPos) !== -1) {
    return 'selected';
  }
  return;
};
```

Testabdeckung

Code-Coverage im Überblick



CinemaTicketReservationSystem	98% (50/51)	65% (217/330)	78% (607/775)
config	100% (4/4)	100% (5/5)	100% (20/20)
util	100% (1/1)	100% (1/1)	100% (6/6)
repositories	100% (0/0)	100% (0/0)	100% (0/0)
requests	90% (10/11)	94% (55/58)	95% (100/105)
services	100% (11/11)	76% (45/59)	78% (198/252)
controllers	100% (10/10)	63% (30/47)	78% (102/130)
entities	100% (13/13)	50% (81/159)	69% (180/260)
CinemaTicketReservationSystemAp	100% (1/1)	0% (0/1)	50% (1/2)

Scan App

- Android Studio als DIE
- Google Pixel 4a als Testgerät für Demo...

Ausblick auf weitere Funktionen

- Karsten Kinobesitzer
  - Oberfläche für einen Überblick über das Unternehmen
  - Zahlen/Erfolge usw.

- Anton Admin
  - Filme durch grafische Oberfläche leicht einstellbar und änderbar