## Hotelbuchungssystem

Lenny Langelotz (1468379)

Aleksander Olewinski (1441066)

Andi Latifi (1456310)

Moin UL Haq (1393703)

Felix Chodykin (1444458)

## Inhaltsverzeichnis

- Was kann unsere Webanwendung?
- Projektdurchführung
- Wie ist unsere Anwendung strukturiert?
- Klassendiagramm
- Gateway Funktionalitäten/Verknüpfung
- Demonstration
- Probleme/Verbesserungsvorschläge

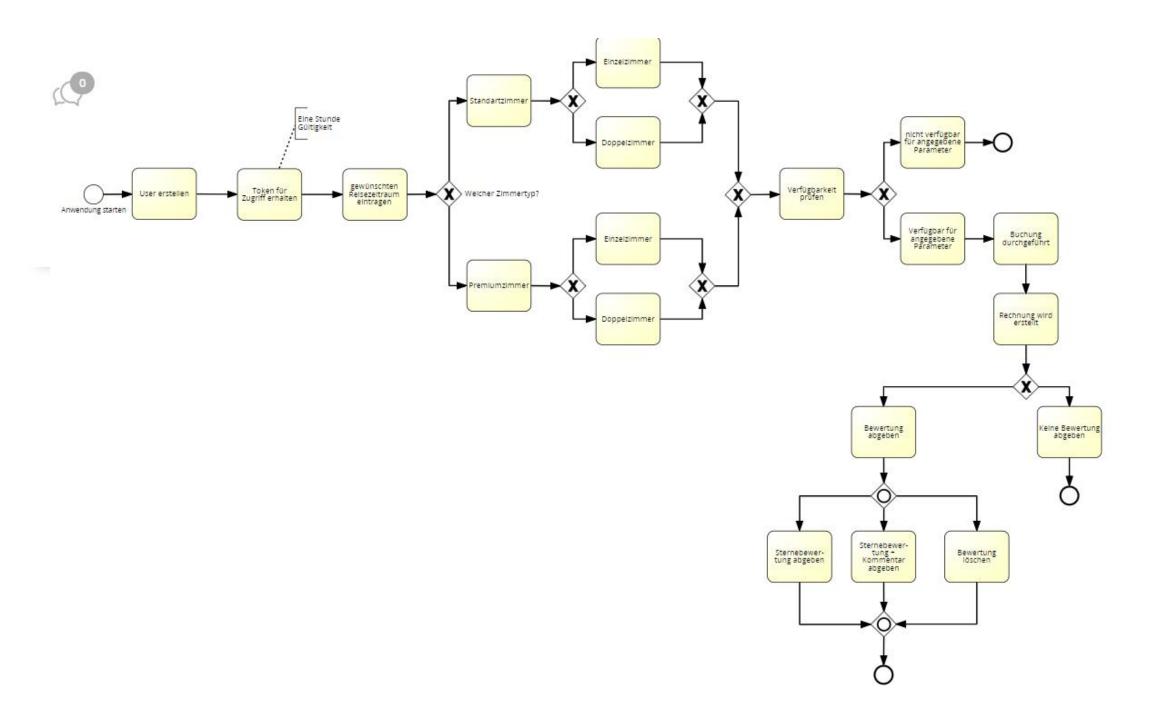
## Was kann unsere Webanwendung?

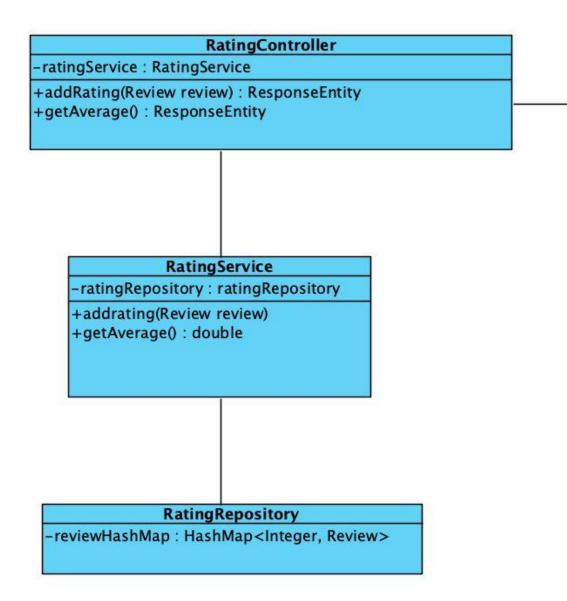


### Projektdurchführung

## Die Projektdurchführung erfolgte mithilfe von folgenden Meilensteinen:

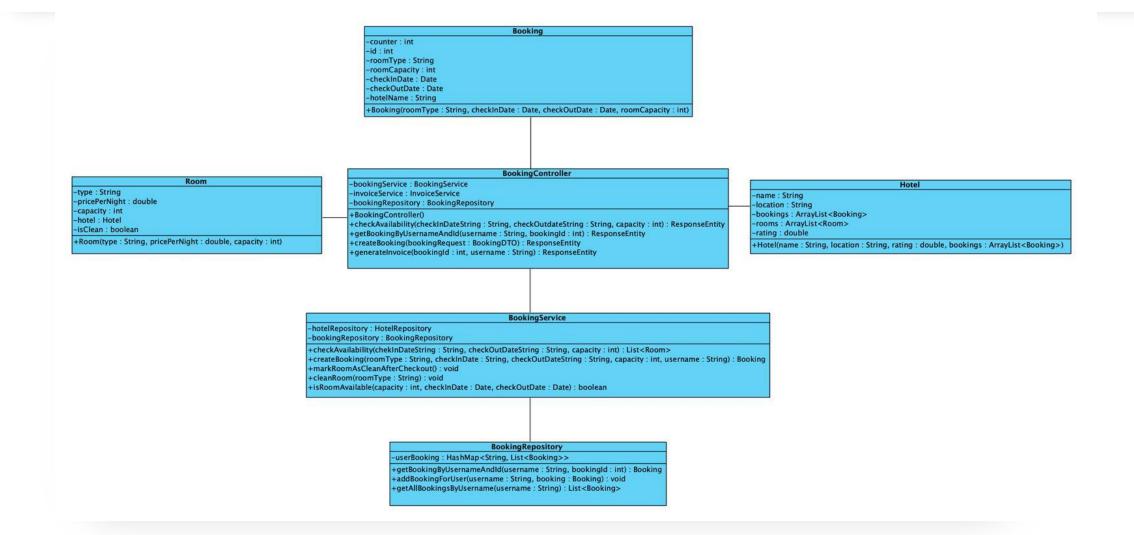
- Projektplanung
- Architekturdesign
- Aufteilung der Aufgaben
- Entwicklung
- Testing

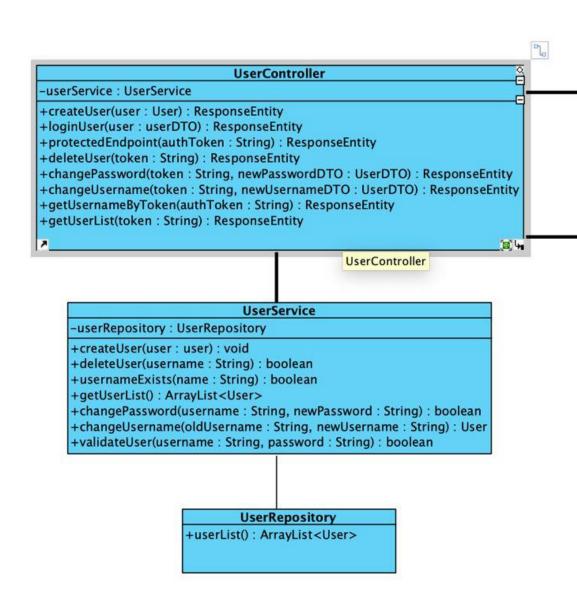




# Review -stars: int -comment: String +Review(stars: int, comment: String) +getComment(): String +setComment(comment: String): void +getStars(): int +setStars(stars: int): void +toString(): String

## Klassendiagramm





-username : String
-firstname : String
-lastname : String
-email : String
-password : String
-role : Role
+User(username : String, firstname : String, lastname : String, email : String, password : String)

-username : String -tokenValue : String

-expiryTime : LocalDateTime

+Token(username : String, tokenValue : String)

#### InvoiceController

-invoiceService : InvoiceService

+generateInvoice(bookingId: int, username: String, hotelDto: Hotel): ResponseEntity

+calculatePricePerNight(booking: Booking, rooms: List<Room>)

+createInvoiceDTO(booking : booking, totalCost : double) : InvoiceDTO

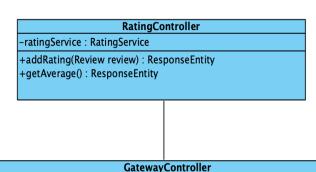
#### **InvoiceService**

-logger : Logger -bookingId : int

-roomType : String

+InvoiceService()

+generateInvoice(bookingId: int, username: String, hoteIDTO: Hotel): InvoiceDTO



#### BookingController

bookingService : BookingServiceinvoiceService : InvoiceService

-bookingRepository : BookingRepository

+BookingController()

+checkAvailability(checkInDateString: String, checkOutdateString: String, capacity: int): ResponseEntity

+getBookingByUsernameAndId(username : String, bookingId : int) : ResponseEntity

+createBooking(bookingRequest : BookingDTO) : ResponseEntity +generateInvoice(bookingId : int, username : String) : ResponseEntity

-restTemplate : RestTemplate

+GatewayController(restTemplate : RestTemplate)

+createUser(user : User) : ResponseEntity +loginUser(user : User) : ResponseEntity +deleteUser(token : String) : ResponseEntity

+changePassword(token : String, newPasswordDTO : UserDTO) : ResponseEntity +changeUsername(token : String, newUsernameDTO : UserDTO) : ResponseEntity

+getUserList(token : String) : ResponseEntity

+createBookingThroughGateway(bookingData: Map<String, Object>, authToken: String): ResponseEntity +getBookingByUsernameAndIdThroughGateway(authToken: String, bookingId: int): ResponseEntity

+getAllBookingsThroughGateway(authToken : String) : ResponseEntity

+generateInvoiceThroughGateway(authToken: Sting, bookingId: int): ResponseEntity

+addRatingThroughGateway(ratingData: Map<String, Object>)

+getAllRatingsThroughGateway(): ResponseEntity +getAverageRatingThroughGateway(): ResponseEntity

#### UserController

-userService : UserService

+createUser(user : User) : ResponseEntity +loginUser(user : userDTO) : ResponseEntity

+protectedEndpoint(authToken : String) : ResponseEntity

+deleteUser(token : String) : ResponseEntity

+changePassword(token: String, newPasswordDTO: UserDTO): ResponseEntity +changeUsername(token: String, newUsernameDTO: UserDTO): ResponseEntity

+getUsernameByToken(authToken : String) : ResponseEntity

+getUserList(token : String) : ResponseEntity

#### InvoiceService

logger : LoggerbookingId : introomType : String

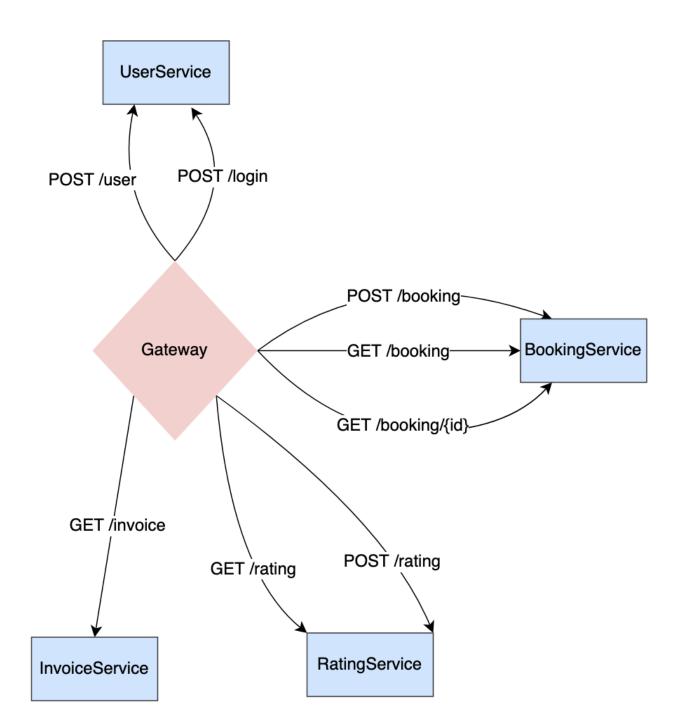
+InvoiceService()

+generateInvoice(bookingId: int, username: String, hoteIDTO: HoteI): InvoiceDTO

## Endpunkte

HTTP-Methode	URI	HTTP Statuscodes
POST	/gateway/user	200 OK, 400 Bad Request
POST	/gateway/login	200 OK, 400 Bad Request, 401 Unauthorized
DELETE	/gateway/user	200 OK, 401 Unauthorized, 404 Not Found
PUT	/gateway/password	200 OK, 400 Bad Request, 401 Unauthorized
PUT	/gateway/username	200 OK, 400 Bad Request, 401 Unauthorized
GET	/gateway/list	200 OK, 401 Unauthorized
POST	/gateway/booking	200 OK, 400 Bad Request, 401 Unauthorized
GET	/gateway/booking/{bookingId}	200 OK, 400 Bad Request, 401 Unauthorized, 404 Not Found
GET	/gateway/booking	200 OK, 401 Unauthorized
GET	/gateway/availability	200 OK, 400 Bad Request
GET	/gateway/booking/invoice/{bookingId}	200 OK, 400 Bad Request, 401 Unauthorized, 404 Not Found
POST	/gateway/rating	200 OK, 400 Bad Request, 401 Unauthorized
GET	/gateway/rating	200 OK, 204 No Content
GET	/gateway/average	200 OK, 204 No Content, 500 Internal Server Error
DELETE	/gateway/ratings/{reviewId}	200 OK, 400 Bad Request, 401 Unauthorized, 404 Not Found

## Gateway Verknüpfungen



## Entscheidende Service Funktionen

UserService: Erstellen eines Users sowie bereitstellen des Tokens zur Authorisierung.

BookingService: Buchungen für Nutzer durchführen und aufrufen. InvoiceService: Erstellen von Rechnungen für User mit Buchungen. RatingService:
Bewerten von
Buchungen durch
User sowie aufrufen
der Bewertungen.

## Probleme bei der Durchführung



Zeitmanagement : Deadlines wurden zu früh gesetzt, schwierig immer einzuhalten



Kommunikationsproblem: Mangelnde Kommunikation während der Entwicklung verursachte Missverständnisse



Fehleinschätzung beim Branches zusammenführen: Unterschätzter Aufwand führte zu Verzögerung



Fehleinschätzung der Komplexität des Gateway und der enthaltenen Verknüpfungen

## Verbesserungsbedarf & Zukunftsperspektive

- HATEOAS Integration f
  ür simplere Navigation & transportieren des Tokens
- Frontend für benutzerfreundliche Anwendung und schnelleres Testen
- Website mit Hypermedia zur Veranschaulichung ergänzen
- Zahlungsservice weiterentwickeln
  - E-Mail Service für Rechnungsversand
  - Verschiedene Zahlungsarten auswählen (API's nutzen)