VOLLEY-RESERVIERUNGSTOOL

IT-Projekt: System Spezifikationen

2. APRIL 2021

DOZENTEN: LUKAS FREY, PROF. DR. BRADLEY RICHARDS
Fachhochschule Nordwestschweiz, BSc Wirtschaftsinformatik

Autoren	
Robin Roth	+41 77 427 05 68 robin.roth@students.fhnw.ch
Luca Schädler	+41 79 501 51 95 luca.schaedler@students.fhnw.ch

Betreuende Dozenten	
Lukas Frey	+41 62 957 22 65 lukas.frey@fhnw.ch
Prof. Dr. Bradley Richards	+41 62 957 23 87 bradley.richards@fhnw.ch

Inhalt

Architekturübersicht 4
Schichtenarchitektur 4
Server (H2 oder MySQL) 4
Systemkomponenten 5
Client 5
Browser 5
Website 5
Server 5
Service Layer (REST-API) 5
Business Layer 5
Persistence Layer 5
Database 5
Entity Relationship Model 7
Beschreibung 8
Services 9
Erweiterte Services 11
Messages 12
MessageLogin 12
MessageNewAccountRequest 12
MessageNewReservation 12
MessageModifyReservation 12
MessageModifyUser 13
Klassendiagramm 14
Beschreibung 15
Loggerklasse 15
Organisatorisches 16

Architekturübersicht

Wir werden im Rahmen des IT-Projekts eine Web-Anwendung basierend auf Java + MySQL (serverseitig) und HTML + JavaScript + CSS (clientseitig) entwickeln. Speziell werden wir serverseitig das Framework «Springtool» verwenden. Auf der Clientseite wird der Kalender mit den Reservationen mittels JQuery-Bibliotheken (Datatables, Datepicker) dargestellt.

Die Webanwendung soll in einem «modernen» Stil aufgebaut sein. Das heisst, dass nicht bei jedem Klick im Browser die komplette Seite vom Server neu geladen wird, sondern clientseitig Logik für die Benutzerinteraktion in JavaScript umgesetzt wird und Zugriffe auf nötige Geschäftslogik und Daten über AJAX-Aufrufe im Hintergrund durchgeführt werden. Die AJAX-Aufrufe werden serverseitig durch verschiedene Services ermöglicht, die mittels AJAX vom Client aufgerufen werden.

Schichtenarchitektur

Wir haben uns für eine Schichtenarchitektur entschieden. Wir versuchen innerhalb der Schichten die Zuständigkeiten klar voneinander abzugrenzen, um die Anwendung verständlicher zu gestalten und somit eine mögliche Weiterentwicklung oder Wartung zu vereinfachen.

Die Einteilung der Schichten sieht wie folgt aus:

- **Präsentationsschicht:** Darstellung der Daten für den Anwender, Logik welche für die Darstellung und die Benutzerinteraktionen nötig ist.
- **Geschäftslogik:** Die eigentliche fachliche Funktionalität.
- **Persistenzschicht:** Zugriff auf die Datenbank und Bereitstellung der Resultate für die höheren Schichten. Zudem Funktionen zum Ändern der Daten in der Datenbank.

Wir wollen jedoch unsere Services als REST-Services anbieten, daher macht es Sinn eine weitere Ebene zwischen Präsentationsschicht und Geschäftslogik zu definieren (Serviceschicht). Die Serviceschicht stellt der Präsentationschicht gezielt die Inhalte über eine definierte API zur Verfügung. In unserem Projekt werden es REST-Services sein, welche HTTP + JSON als Protokolle gebrauchen.

Server (H2 oder MySQL)

Um unsere Applikation zu testen verwenden wir den von Spring bereitgestellten H2 Server oder den MySQL-Server. Beide Server werden Lokal auf dem Gerät ausgeführt. Der Unterschied besteht darin, dass wird den H2 Server über eine Konsole im Browser aufrufen können und den MySQL Server über die MySQL-Workbench. Wir werden in der nächsten Phase des Projekts die Entscheidung treffen. In unserem Fall sollte es jedoch keine grosse Rolle spielen, für welchen Server wir uns dann entscheiden.

Systemkomponenten

Client

Browser

In einem lokalen Netzwerk wird über den Browser eine Kommunikation mit dem Server gestartet. Die Kommunikation erfolgt durch HTTP-Aufrufe (GET/POST/PUT/DELETE).

Website

Dem User wird eine ansprechende Website angezeigt, über diese er interagieren kann.

Server

Service Layer (REST-API)

Die eingehenden HTTP-Aufrufe werden vom Server verarbeitet, d.h. die dazugehörenden Java-Methoden werden aufgerufen.

Business Layer

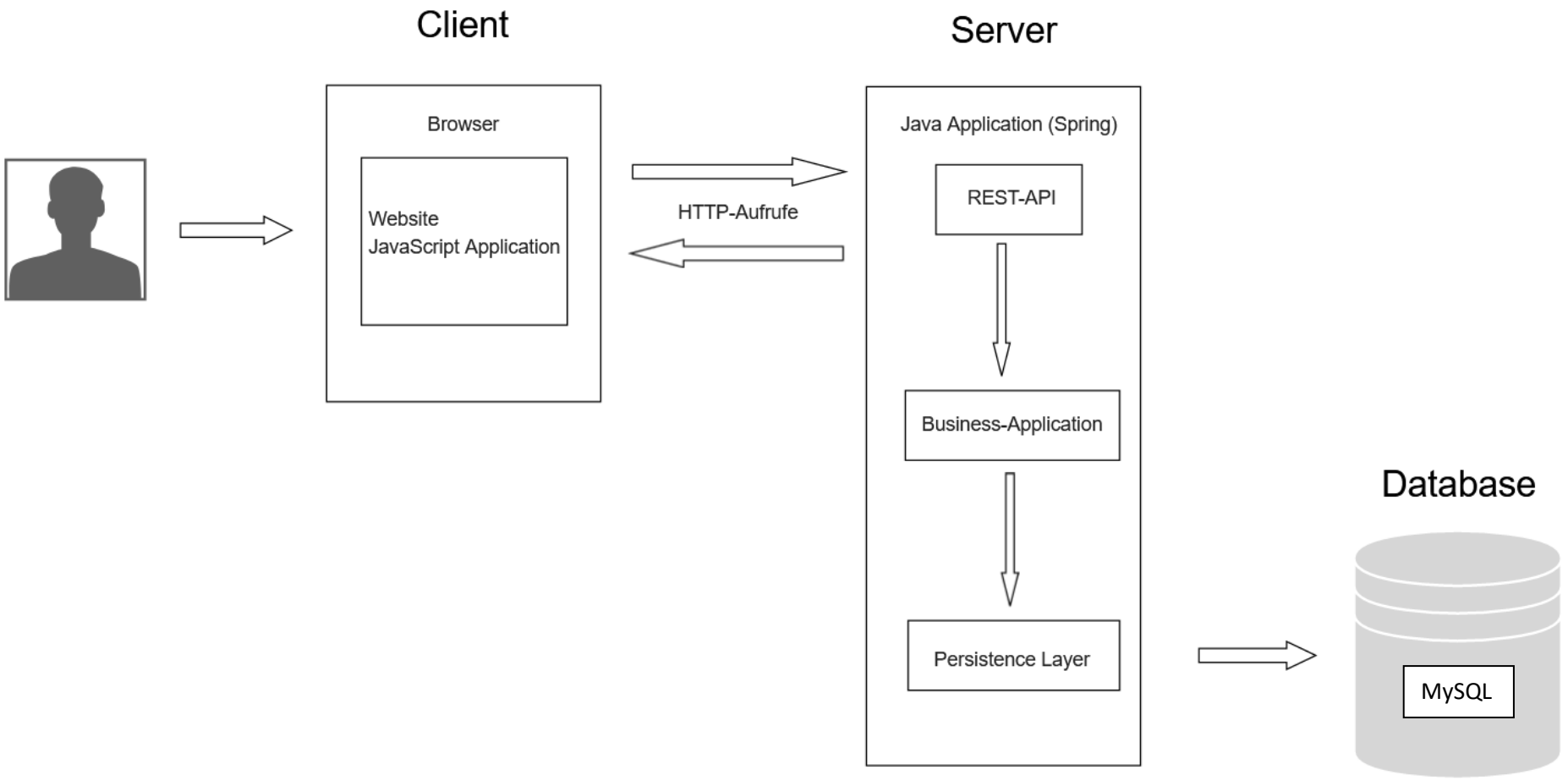
Durch den Methodenaufruf wird die eigentliche Geschäftslogik ausgeführt (wie z.B. Überprüfen ob der Benutzer für diese Transaktion berechtigt ist)

Persistence Layer

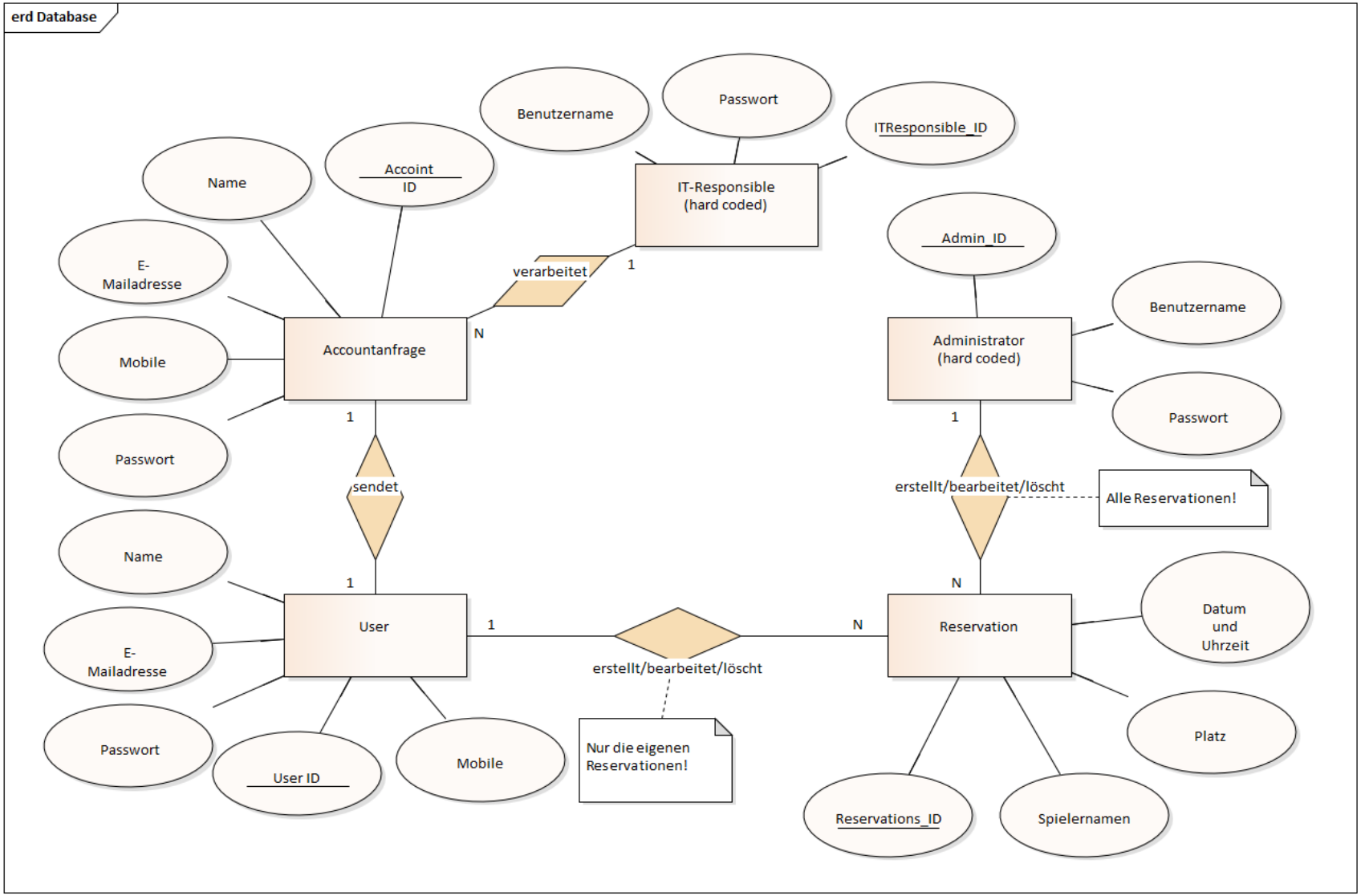
Die Änderungen an den Datenobjekten werden vorgenommen (wie z.B. die Löschung einer Reservationsinstanz)

Database

Die Daten werden in Form von Tabellen gespeichert.


Client
Server
Browser
Website
JavaScript Application
HTTP-Aufrufe
Java Application (Spring)
REST-API
Business-Application
Persistence Layer
Database
MySQL

Entity Relationship Model



Beschreibung

Im Entity Relationship Model sind alle Entitäten mit ihren dazugehörigen Attributen abgebildet. Unsere Anwendung benötigt diese spezifischen Tabellen auf der Datenbank und müssen daher erstellt werden. Die Erstellung der Tabellen macht Springtool automatisch über Annotationen in den Entitätsklassen. Die Befüllung der Tabellen übernimmt dann die Persistenzschicht.

Services

Das Anzeigen der bereits erstellten Reservationen (Reservationen werden auf den Kalender geladen) erfolgt beim Start der Applikation (in Spring-Start-Methode enthalten). Alle anderen Services werden in den nachfolgenden Tabellen erläutert.

Account anfragen	UC-101
Input	Unter Angabe des Namens, E-Mailadresse und Mobile und Passwort
HTTP-Methode	POST
Path	/api/account_request
Parameter	-
Requestbody	MessageNewAccountRequest - JSON
Resultat	boolean

Login	UC-102
Input	Unter Angabe der E-Mailadresse und Passwort
HTTP-Methode	POST
Path	/api/login
Parameter	-
Requestbody	-MessageLogin - JSON
Resultat	Boolean - JSON

Reservation erstellen	UC 201
Input	Unter Angabe des Datums, Uhrzeit, Platz und Spielername
HTTP-Methode	POST
Path	/api/reservation
Parameter	-
Requestbody	MessageNewReservation - JSON
Resultat	ID der neuen Reservation als JSON

Reservation löschen	UC-202
Input	-
HTTP-Methode	DELETE
Path	/api/reservation/{reservationid}
Parameter	-
Requestbody	-
Resultat	Boolean - JSON

Reservation ändern	UC-203
Input	Unter Angaben der zu ändernden Informationen
HTTP-Methode	PUT
Path	/api/reservation/{reservationid}/modify
Parameter	-
Requestbody	MessageModifyReservation
Resultat	Boolean - JSON

Account löschen	UC-302
Input	-
HTTP-Methode	DELETE
Path	/api/user_delete/{userid}
Parameter	-
Requestbody	-
Resultat	Boolean - JSON

Userliste anzeigen	UC-303
Input	-
HTTP-Methode	GET
Path	/api/users
Parameter	-
Requestbody	-
Resultat	List<User> - JSON

Erweiterte Services

Accountdaten ändern	UC-203
Input	Unter Eingabe der zu ändernden Informationen
HTTP-Methode	PUT
Path	/api/user/{userid}/modify
Parameter	-
Requestbody	MessageModifyUser
Resultat	Boolean - JSON

Reservationsliste anzeigen	UC-203
Input	-
HTTP-Methode	GET
Path	/api/reservations
Parameter	-
Requestbody	-
Resultat	List<Reservation> - JSON

Messages

MessageLogin

{

«email» : «String»,

«password» : «String»

}

MessageNewAccountRequest

{

«name» : «String»,

«email» : «String»,

«mobile» : «String»,

«password» : «String»

}

MessageNewReservation

{

«date» : «date»

«Time» : «time»

«court» : «int»

«playernames» : «String»

}

MessageModifyReservation

{

«Date» : «date»

«Time» : «time»

«court» : «int»

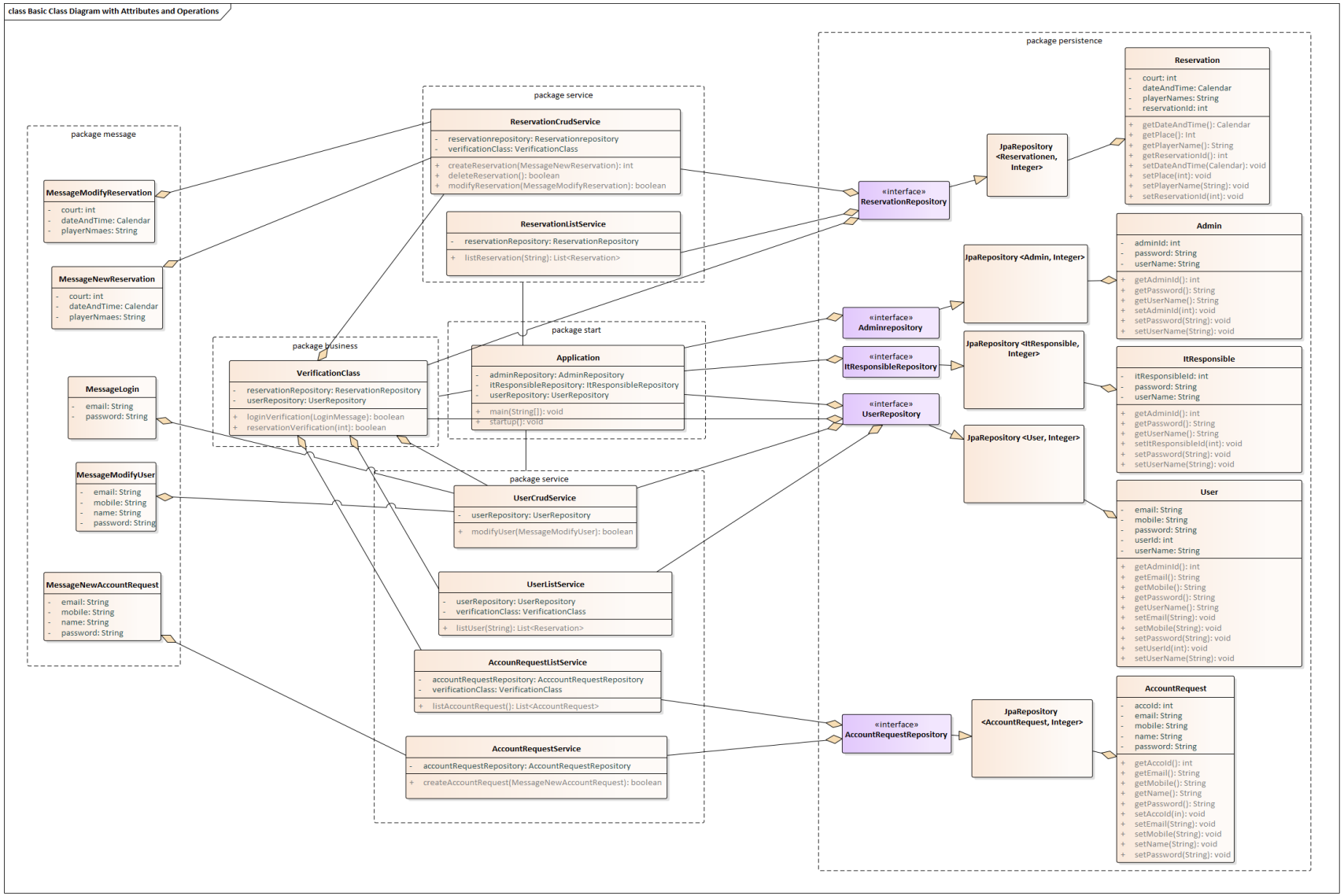
«playernames» : «String»

}

MessageModifyUser

```
{
        «name» : «String»,
        «email» : «String»,
        «mobile» : «String»,
        «password» : «String»
}
```

Klassendiagramm



Beschreibung

Beim Start der Applikation (Startklasse ist die die Klasse «Application») werden die Serviceklassen ebenfalls gestartet, daher die Verbindung des Servicepackages zu der Startklasse. Wie die Verbindung letztlich aussieht, wissen wir zu dem jetzigen Zeitpunkt noch nicht, daher haben wir dies als undefinierte Verbindung modelliert.

Loggerklasse

Die Loggerklasse wird im Klassendiagramm aufgrund der Übersichtlichkeit nicht berücksichtigt. Wir werden eine Klasse «LoggerClass» erstellen, welche von der vordefinierten Javaklasse «Logger» erbt. Die Klassen in den Packages Service, Start und Business werden jeweils eine Instanz der «LoggerClass» aufweisen, um so ein eigenes Logger Objekt zu instanziieren, welches alle wichtigen Meldungen in einem Textfile festhält.

Organisatorisches

Da wir zu zweit sind wird es wohlmöglich viele Überschneidungen geben. Im Quellcode werden wir jeweils angeben wer welche Methode geschrieben hat. In der nachstehenden Tabelle haben wir grob die Aufgabenbereiche aufgeteilt.

Thema	Teammitglied
Backend	v.a. Luca Schädler
Frontend	v.a. Robin Roth
Git / Verwaltung	Robin Roth / Luca Schädler
Testing	Robin Roth / Luca Schädler