

Dokumentation 4Bayzeit

Projektgruppe: Selbsthilfegruppe

14. Dezember 2017

Matthias Theimer, Marco Vogel, Verena Schreyer

VINF2015/2018 - Praktikum Programmieren

Inhaltsverzeichnis

[1. Einleitung 2](#_Toc501020216)

[1.1. Motivation 2](#_Toc501020217)

[1.2. Allgemeine Hinweise 2](#_Toc501020218)

[2. Anforderungen 2](#_Toc501020219)

[3. Komponenten 6](#_Toc501020220)

[3.1. Datenbank 7](#_Toc501020221)

[3.1.1. Techniken 7](#_Toc501020222)

[3.1.2. Aufbau 7](#_Toc501020223)

[3.1.3. Beschreibung 8](#_Toc501020224)

[3.1.4. Sprachen 8](#_Toc501020225)

[3.1.5. Bibliotheken 8](#_Toc501020226)

[3.1.6. Probleme 8](#_Toc501020227)

[3.2. Business Logic 8](#_Toc501020228)

[3.2.1. Techniken 8](#_Toc501020229)

[3.2.2. Aufbau 8](#_Toc501020230)

[3.2.3. Beschreibung 10](#_Toc501020231)

[3.2.4. Sprachen 10](#_Toc501020232)

[3.2.5. Bibliotheken 10](#_Toc501020233)

[3.2.6. Probleme 10](#_Toc501020234)

[3.3. WebService 10](#_Toc501020235)

[3.3.1. Techniken 10](#_Toc501020236)

[3.3.2. Aufbau 10](#_Toc501020237)

[3.3.3. Beschreibung 10](#_Toc501020238)

[3.3.4. Sprachen 10](#_Toc501020239)

[3.3.5. Bibliotheken 11](#_Toc501020240)

[3.3.6. Probleme 11](#_Toc501020241)

[3.4. Web-Applikation 11](#_Toc501020242)

[3.4.1. Techniken 11](#_Toc501020243)

[3.4.2. Aufbau 11](#_Toc501020244)

[3.4.3. Beschreibung 13](#_Toc501020245)

[3.4.4. Sprachen 13](#_Toc501020246)

[3.4.5. Bibliotheken 13](#_Toc501020247)

[3.4.6. Probleme 13](#_Toc501020248)

[4. Umsetzung 13](#_Toc501020249)

1. Einleitung
   1. Motivation
   2. Allgemeine Hinweise

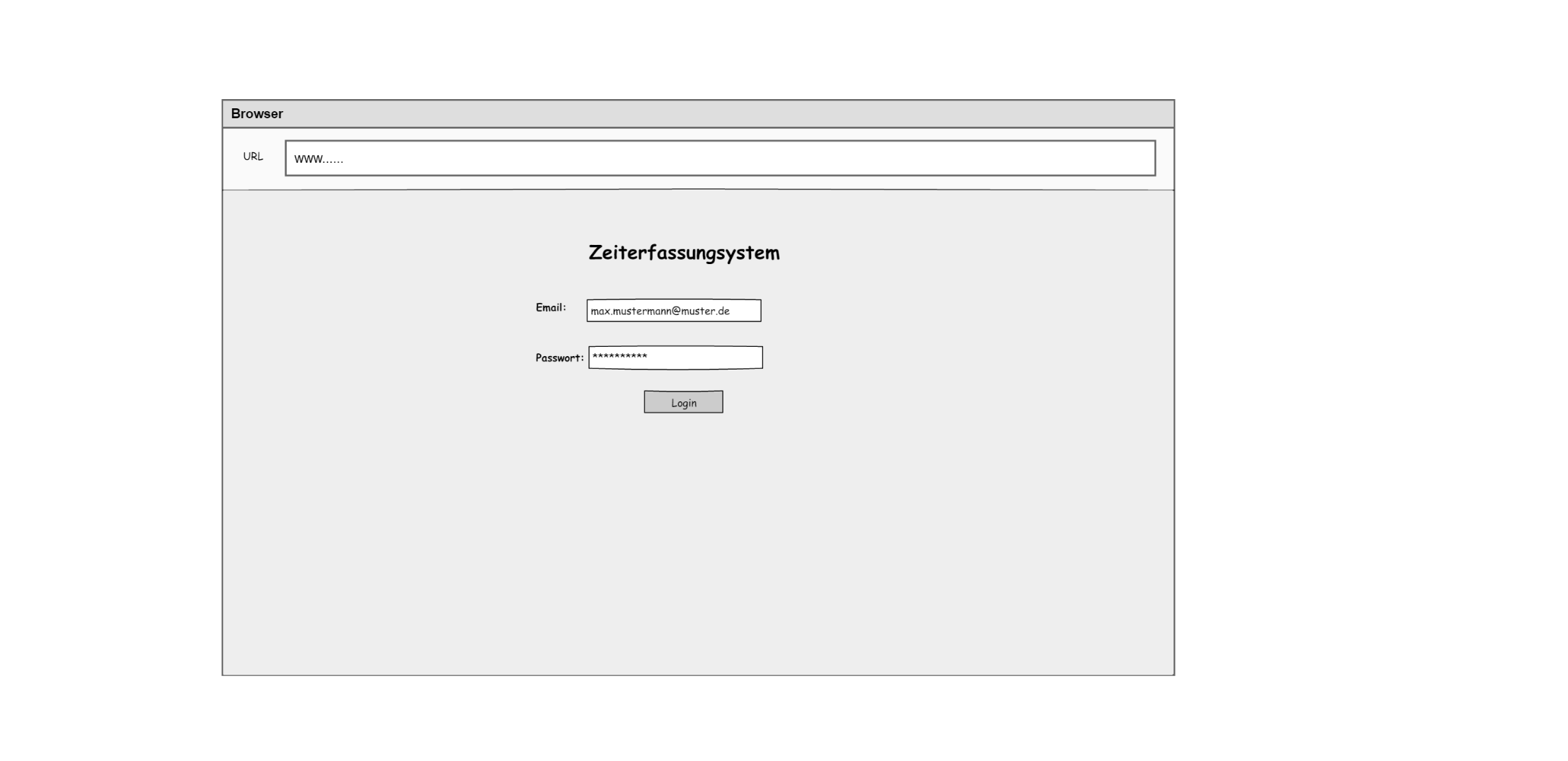
Erreichbar unter: http://52.166.116.86:8080/ZESWebApp/

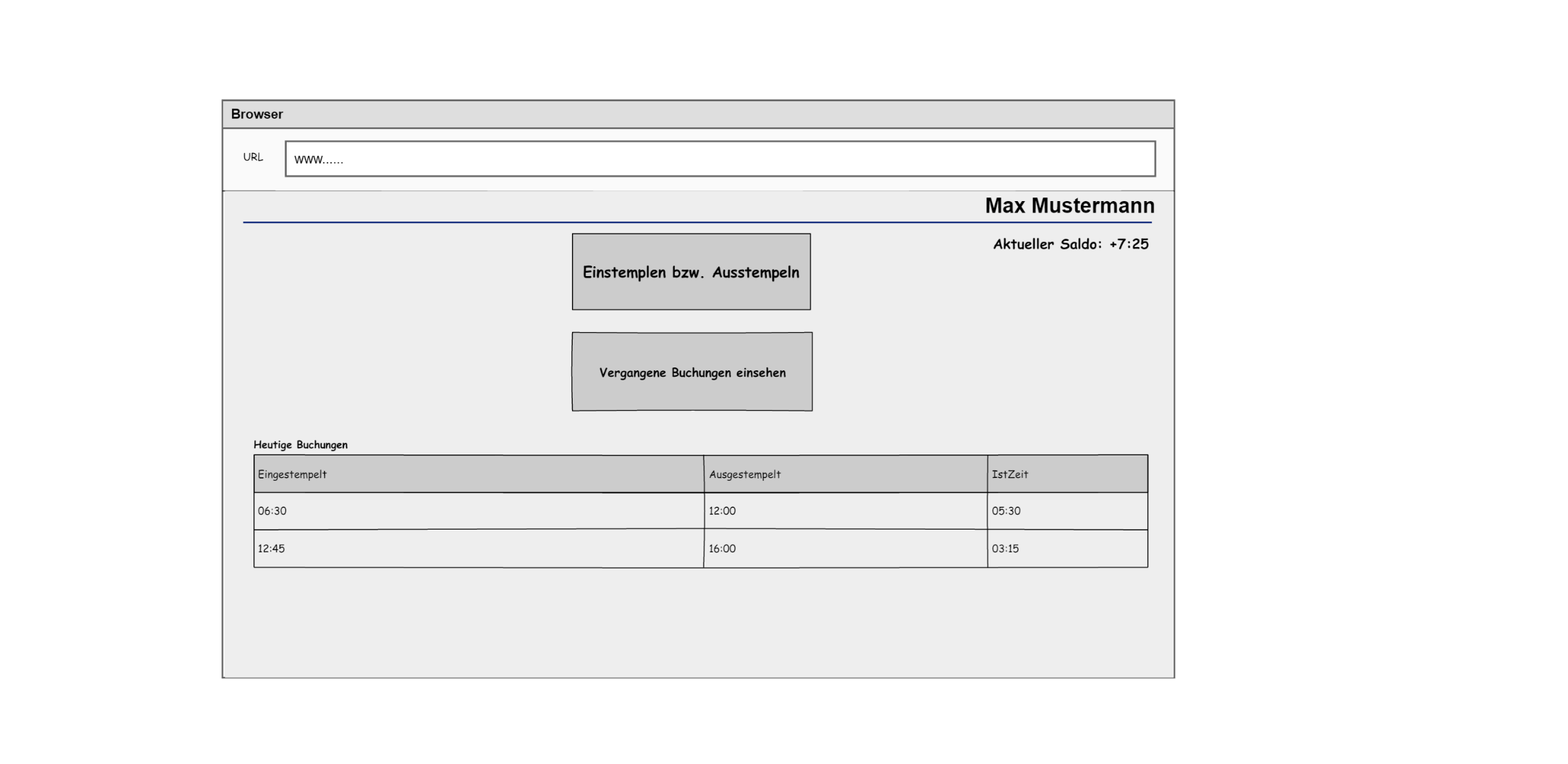
Oder unter: selbsthilfegruppe.westeurope.cloudapp.azure.com/ZESWebApp/

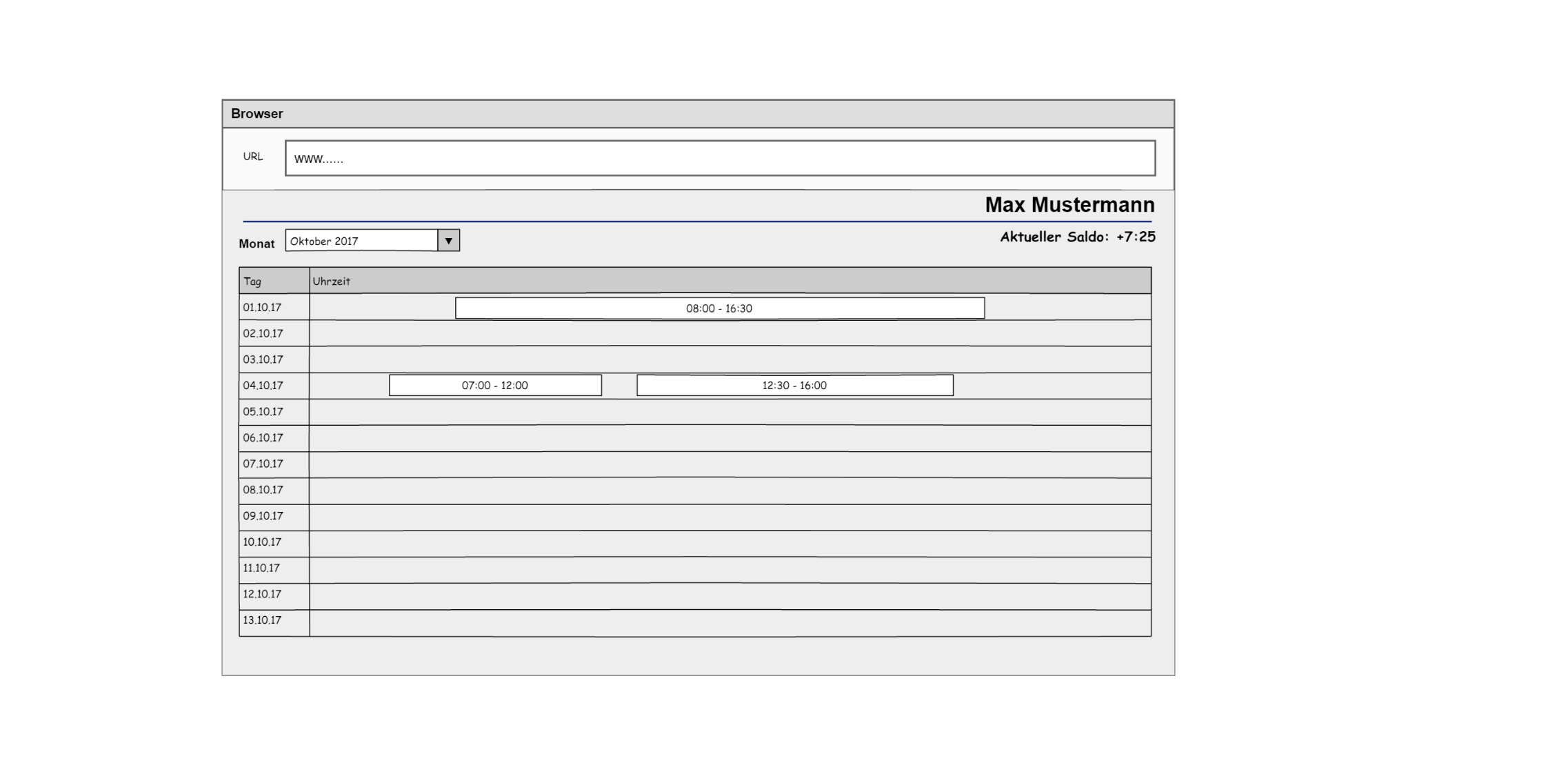
1. Anforderungen

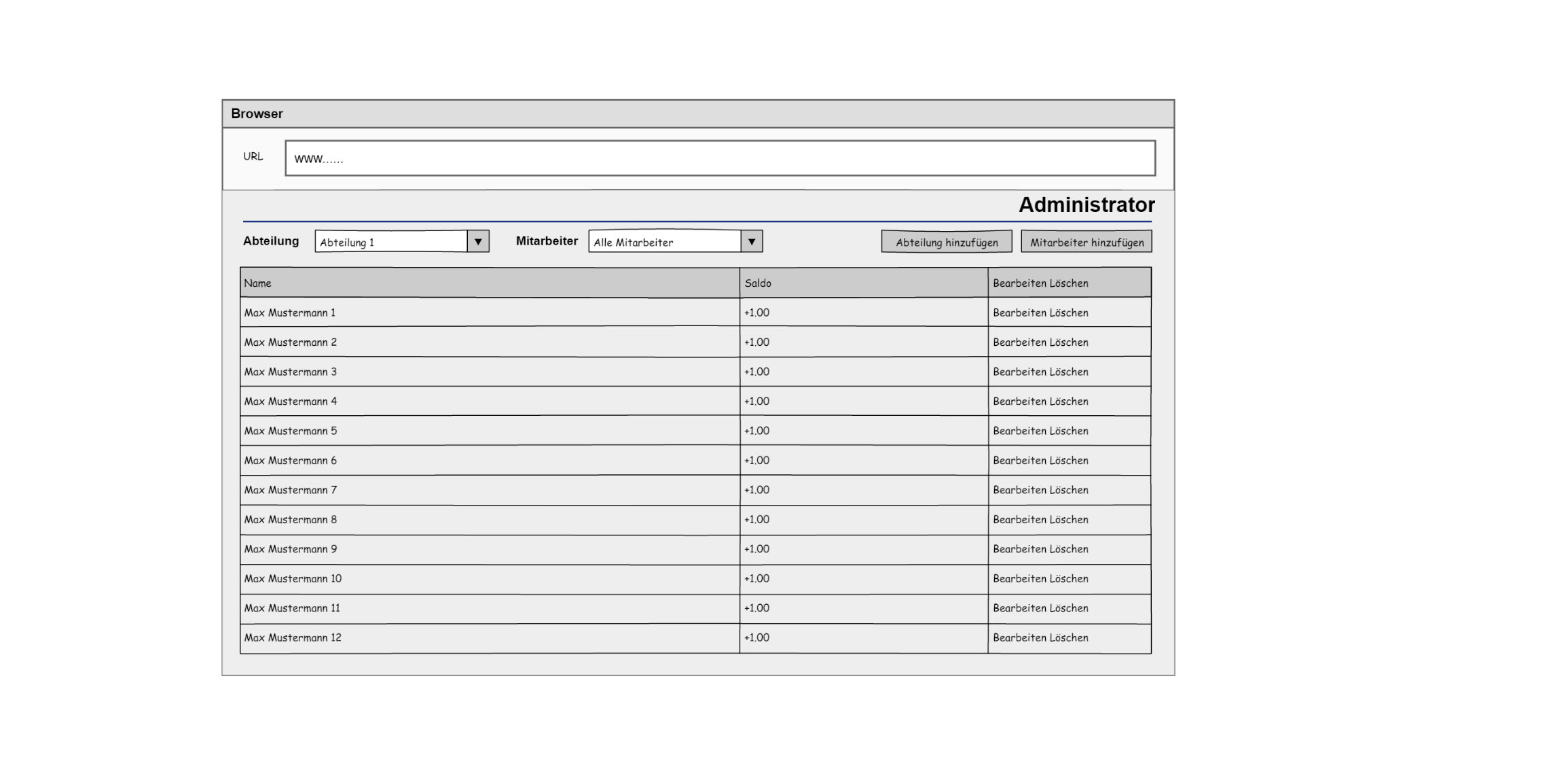
* Buchung:
  + Jede Buchung besteht aus Beginn und Ende
  + mehrere pro Tag möglich (nicht über Mitternacht)
  + Abwechselnd im Rhythmus Kommen/Gehen
* Login
  + Authentifizierung per Email und Passwort
  + 2 Rollen (Admin, Mitarbeiter)
* Admin-Rolle
  + Mitarbeiter hinzufügen, bearbeiten und löschen
  + Abteilungen hinzufügen und löschen
  + Buchungen hinzufügen, bearbeiten und löschen für einzelne Mitarbeiter
  + Jahresabschluss durchführen (Zusammenfassung der Buchungen)
* Mitarbeiter
* Ein-/Ausstempeln
* Vergangene Buchungen einsehen
* Saldo einsehen

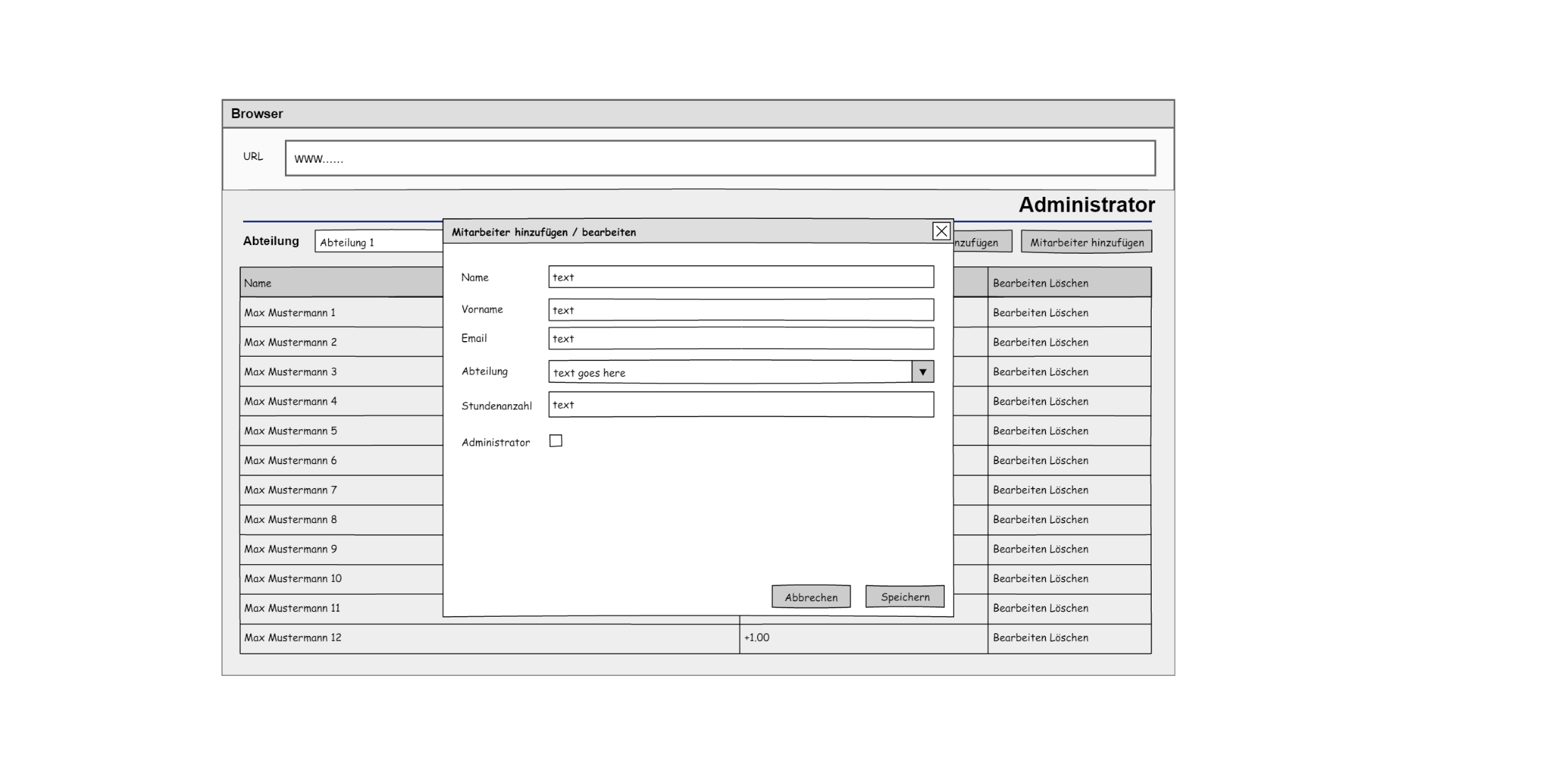
**Nicht geplant:**   
Urlaubsverwaltung, Buchungsmöglichkeit am Samstag/Sonntag/Feiertag, Sicherung der zusammengefassten Buchungen, Barrierefreiheit, HTTPS Unterstützung

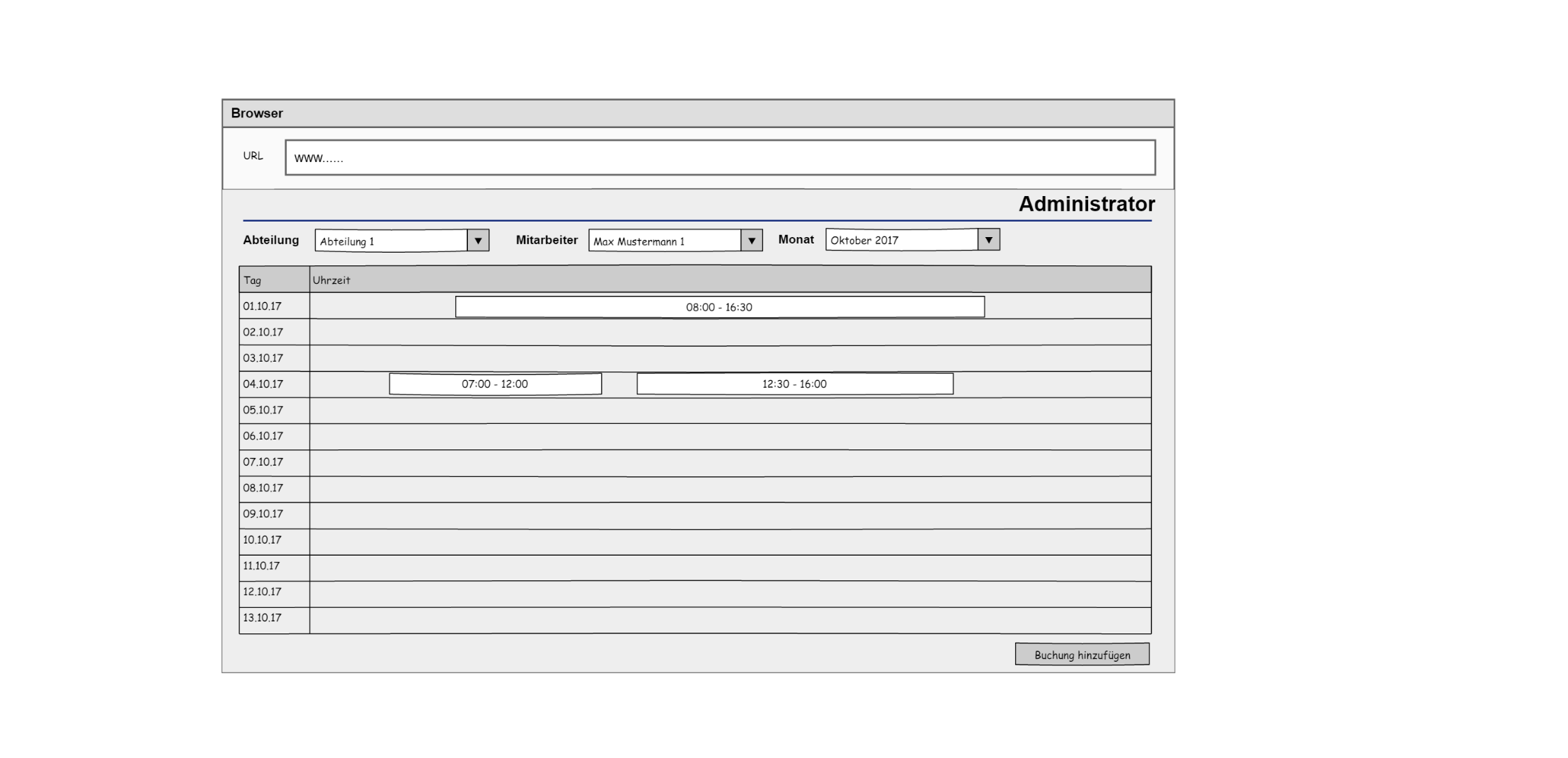


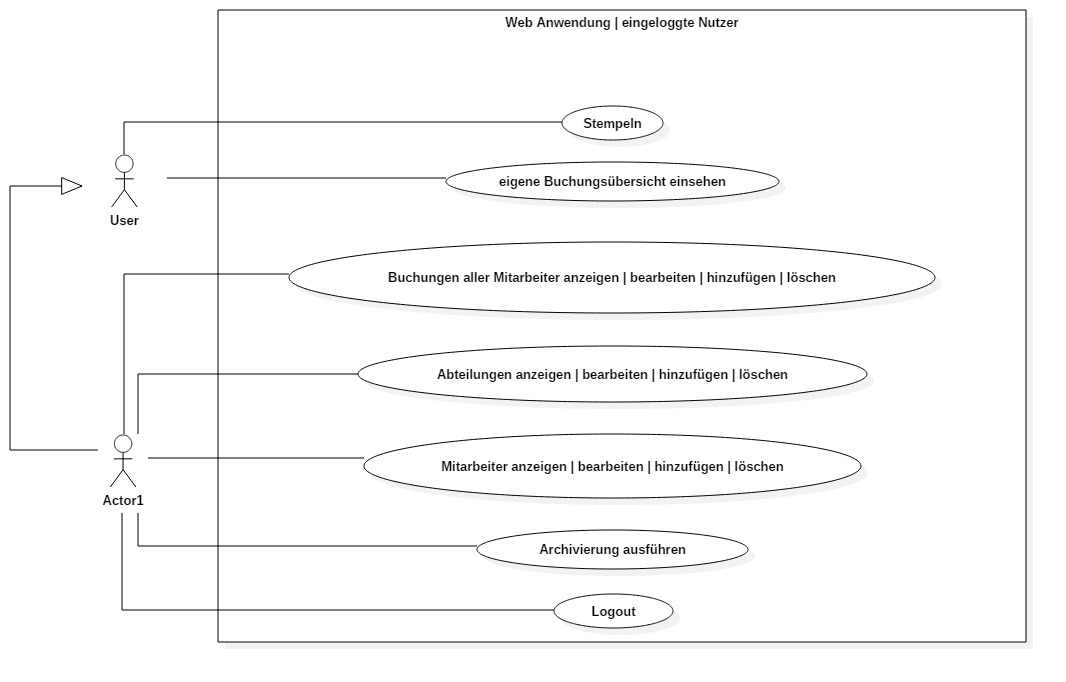




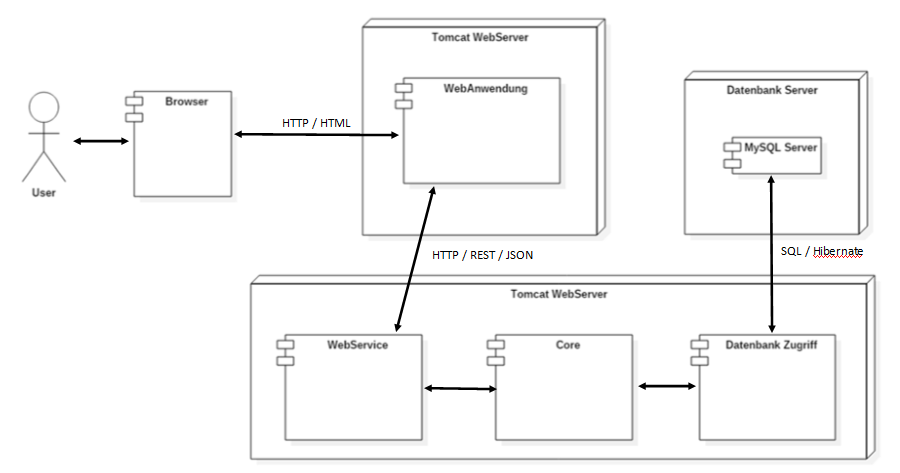




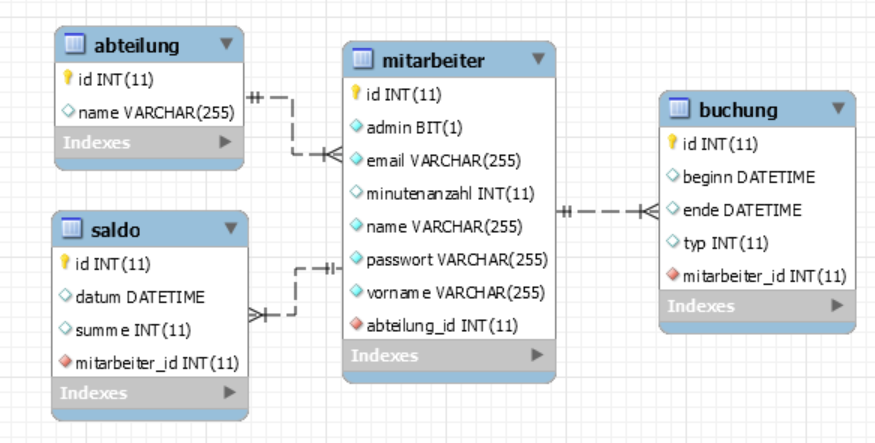




1. Komponenten



* 1. Datenbank



* Relationale Speicherung der Entitäten
* Passwörter gehashed mit SHA-256

IP-Adresse: 52.166.116.86

Port: 3306

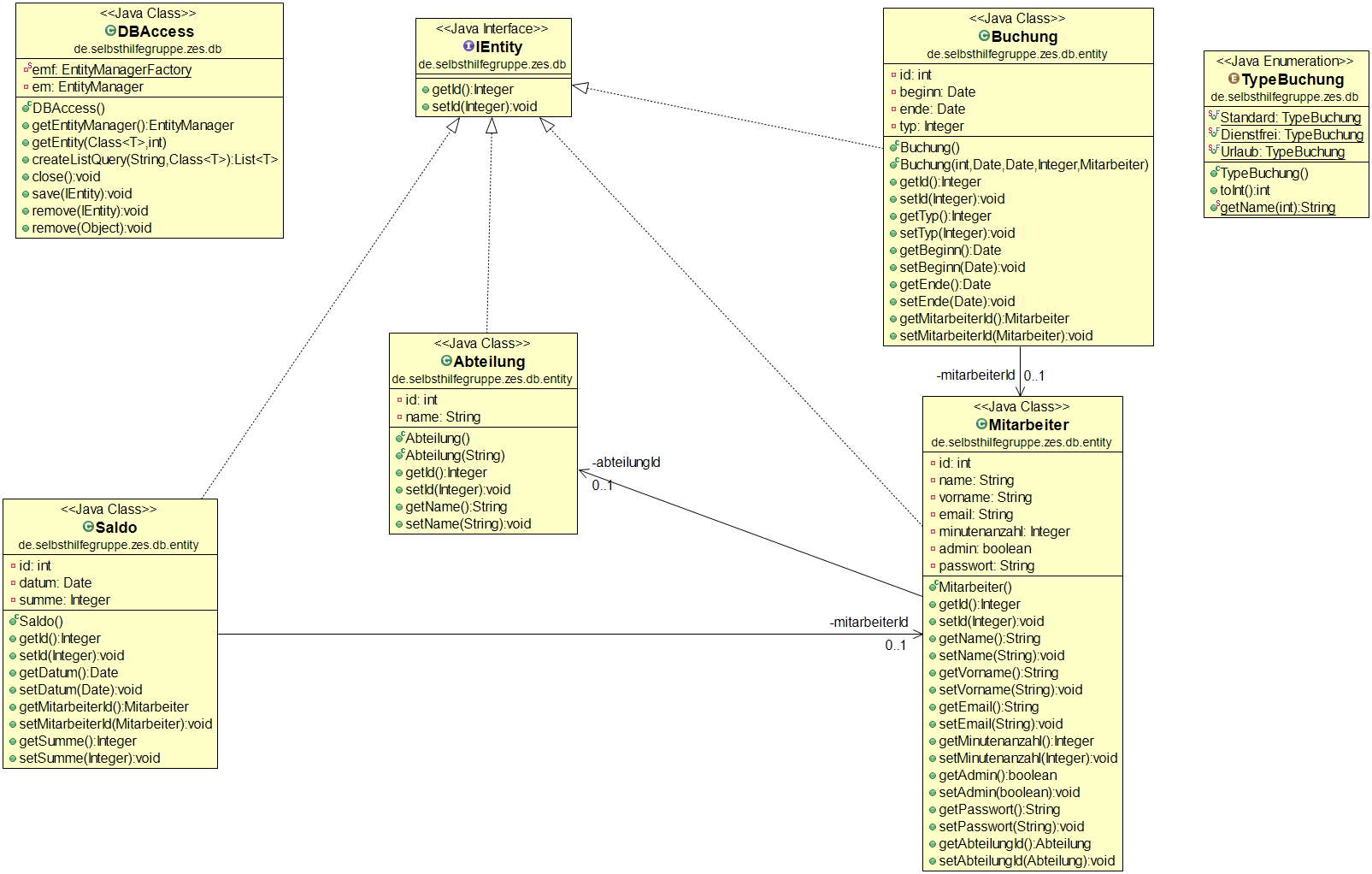
Schema: zes\_dbtheimer

Nutzer: root

Passwort: 4FSD21efcsAS

* + 1. Techniken
    2. Aufbau

Das zes\_db Projekt ist ein Wrapper um die benutzte MySQL Datenbank. Sie benutzt Hibernate als Framework. Die Entity Klassen benutzen Hibernate Annotationen und bilden dadurch die Datenbanktabellen ab. In der Datei persistence.xml ist die Konfiguration des Hibernate-Frameworks gespeichert. In ihr sind zum Beispiel die IP-Adresse des SQL-Servers und das Passwort hinterlegt. Der Zugriff auf dieses Projekt, und damit auf die DB, erfolgt durch die Klasse „DBAccess“. Diese bietet nach Erzeugung verschiedene Methoden zur Interaktion mit der Datenbank an. Zum einen besitzt das Objekt die klassischen CRUD (Create, Read, Update, Delete) Operationen, zum anderen kann ein externes Projekt auf die Entitymanagerfactory und den EntityManager über entsprechende Getter und Setter zugreifen.



* + 1. Beschreibung
    2. Sprachen

-Java

* + 1. Bibliotheken

- Hibernate

- Google Gson  
- Log4J

* + 1. Probleme
  1. Business Logic

In der Business Logic (zes\_core) findet sich die Logik hinter der Anwendung. Dazu gehört die Berechnung des Saldos für die jeweiligen Tage, Prüfungen beim Hinzufügen und Bearbeiten von Buchungen, die Authentifizierung und ein Datenbankzugriff, um Änderungen festschreiben zu können. Außerdem werden die Objekte für den Webservice aus der Datenbank gelesen und entsprechend umgewandelt.

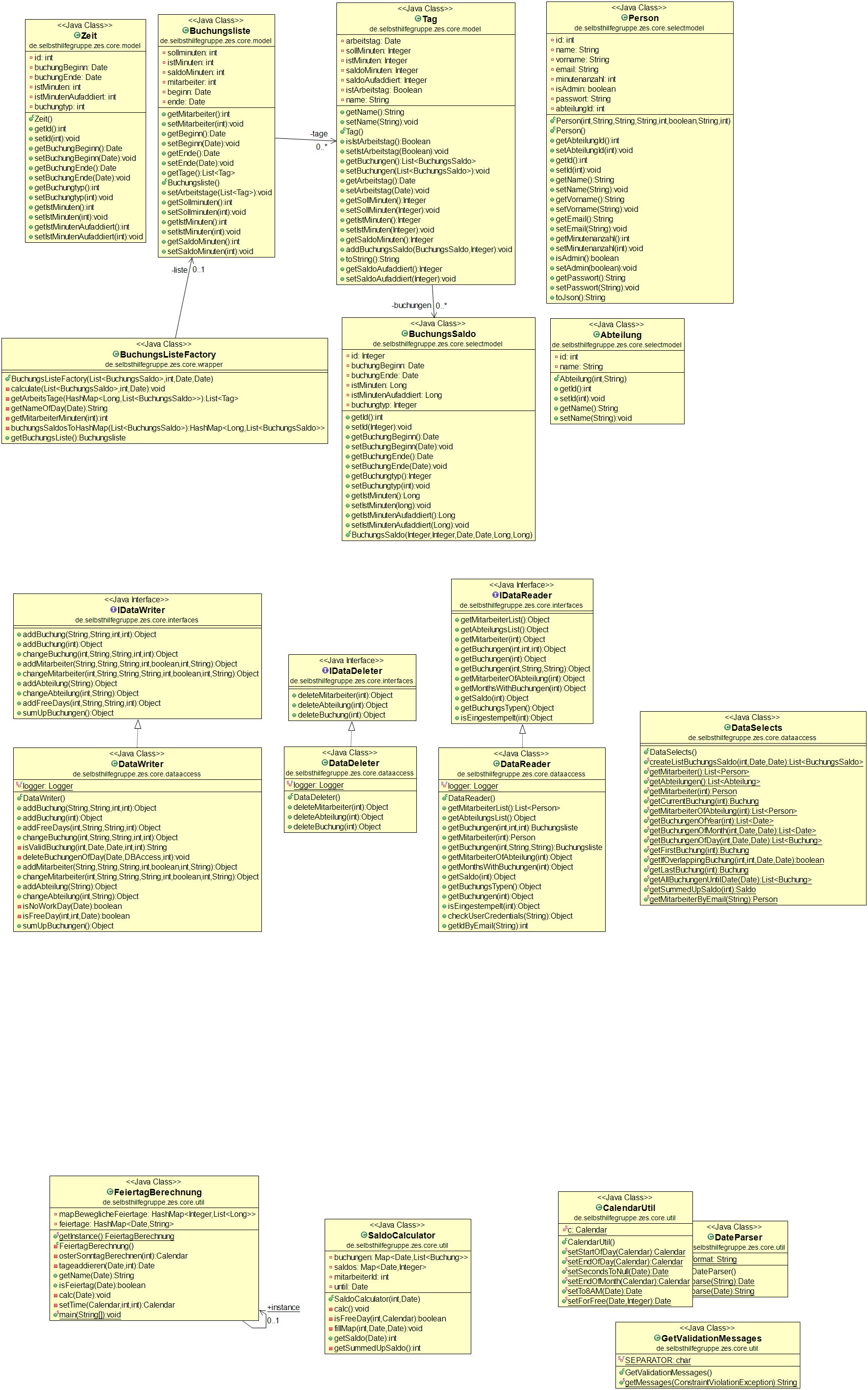
* + 1. Techniken
       1. Hibernate Query Language HQL

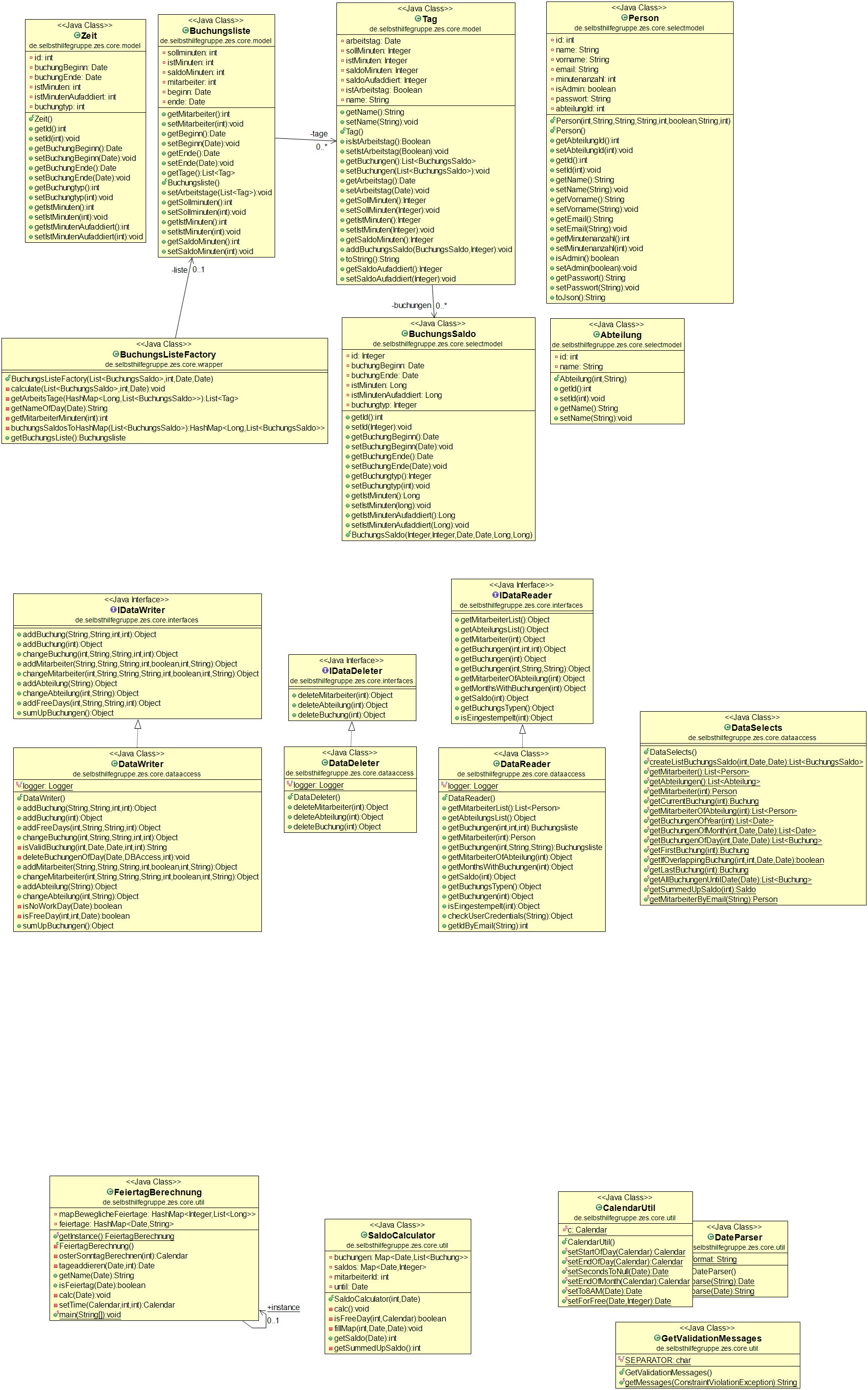
Für Abfragen aus der Datenbank wird die Hibernate Query Language verwendet. Dafür werden die Objekte und der Datenbankzugriff (de.selbsthilfegruppe.zes.db.DBAccess) aus der Datenbankkomponente (zes\_db) verwendet.

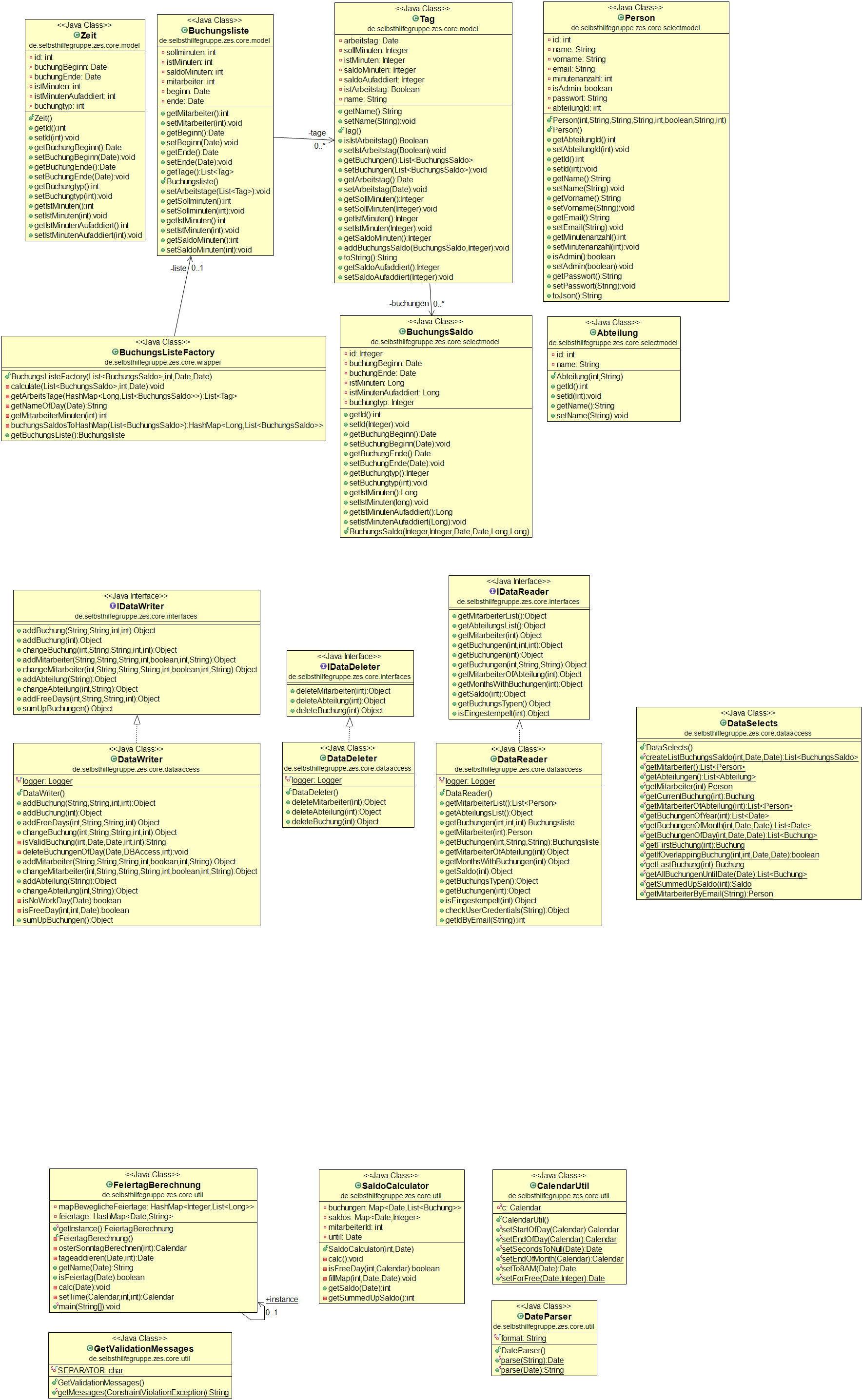
* + - 1. Apache Log4j
      2. Authentifizierung

Die Buisness Logic übernimmt außerdem die Authentifizierung der Nutzer. Der Webservice leitet den Base64 encodierten String an zes\_core weiter. Der encodierte String enthält die E-Mail und das Passwort des Users. Dieser String wird dann decodiert und der Nutzer mit angegebener E-Mail wird aus der Datenbank geladen. Das übergebene Passwort wird mit SHA-265 gehashed und dann mit dem Passwort in des geladenen Nutzers verglichen. Wenn die beiden Hash Werte übereinstimmen ist der Nutzer erfolgreich authentifiziert und der Wert true wird dem Webservice zurückgeliefert.

* + 1. Aufbau
       1. Models
       2. Schnittstelle zum Webservice
       3. Utils







* + 1. Beschreibung
    2. Sprachen

Für die Programmierung wurde ausschließlich Java verwendet.

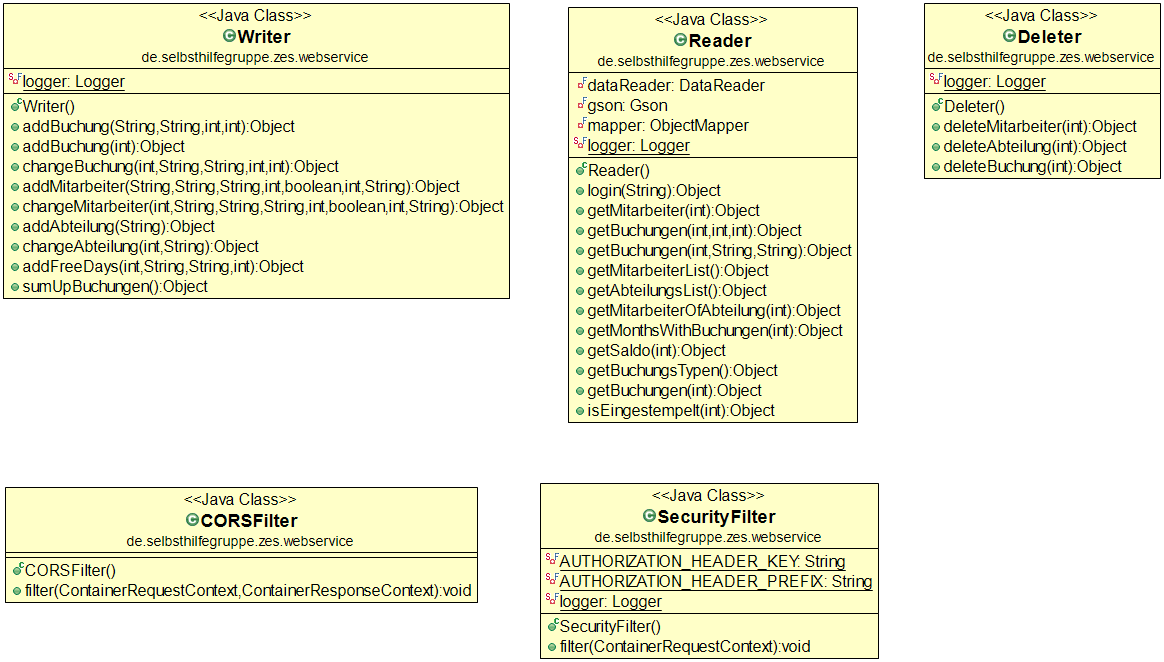
* + 1. Bibliotheken
    2. Probleme
  1. WebService
* Verbindung zwischen verschiedenen Clients und Business Logic
* Rest-WebService
* Get-Methoden (Mitarbeiterliste, Zeitübersicht, …)
* Post-Methoden (Mitarbeiter anlegen, Mitarbeiter ändern, Buchung anlegen, …)
  + 1. Techniken

→ JAX-RS

→ Tomcat 8

* + 1. Aufbau

Der Webservice hat die Aufgabe eine Webschnittstelle für die Webanwendung bereitzustellen. Das Projekt benutzt JAX-RS zur Bereitstellung eines Rest-Services. Die Klassen DataReader, DataWriter, DataDeleter implementieren die entsprechenden Interfaces aus dem zes\_core Projekt um eine einheitliche Zugriffsstruktur zu garantieren. Die Methoden werden daraufhin über entsprechende URLs publiziert und mit den benötigten JAX-RS Annotationen versehen. Außerdem sind zwei Filter implementiert. Zum einen wird in der Klasse „AuthenticationFilter“ die Authentifizierung der Applikation behandelt. Jeder Request, der sich an den Rest Service richtet,  wird von diesem Filter abgefangen und auf einen Header mit dem Namen „Authentication“ durchsucht. Sollte dieser Header nicht gefunden werden wird eine Response mit Fehler 401 („Unauthorized“) an den Sender zurückgeschickt. Sollte dieser Header vorhanden sein wird der Inhalt des Headers an den Core zur Validierung weitergegeben. Die Klasse „HeaderFilter“ fügt jeder Response, welche vom Rest-Service versendet werden soll, Header hinzu, die somit nicht in jeder Klasse implementiert werden müssen.



* + 1. Sprachen

- Java

- JSON

* + 1. Bibliotheken

- Gson

- Log4j

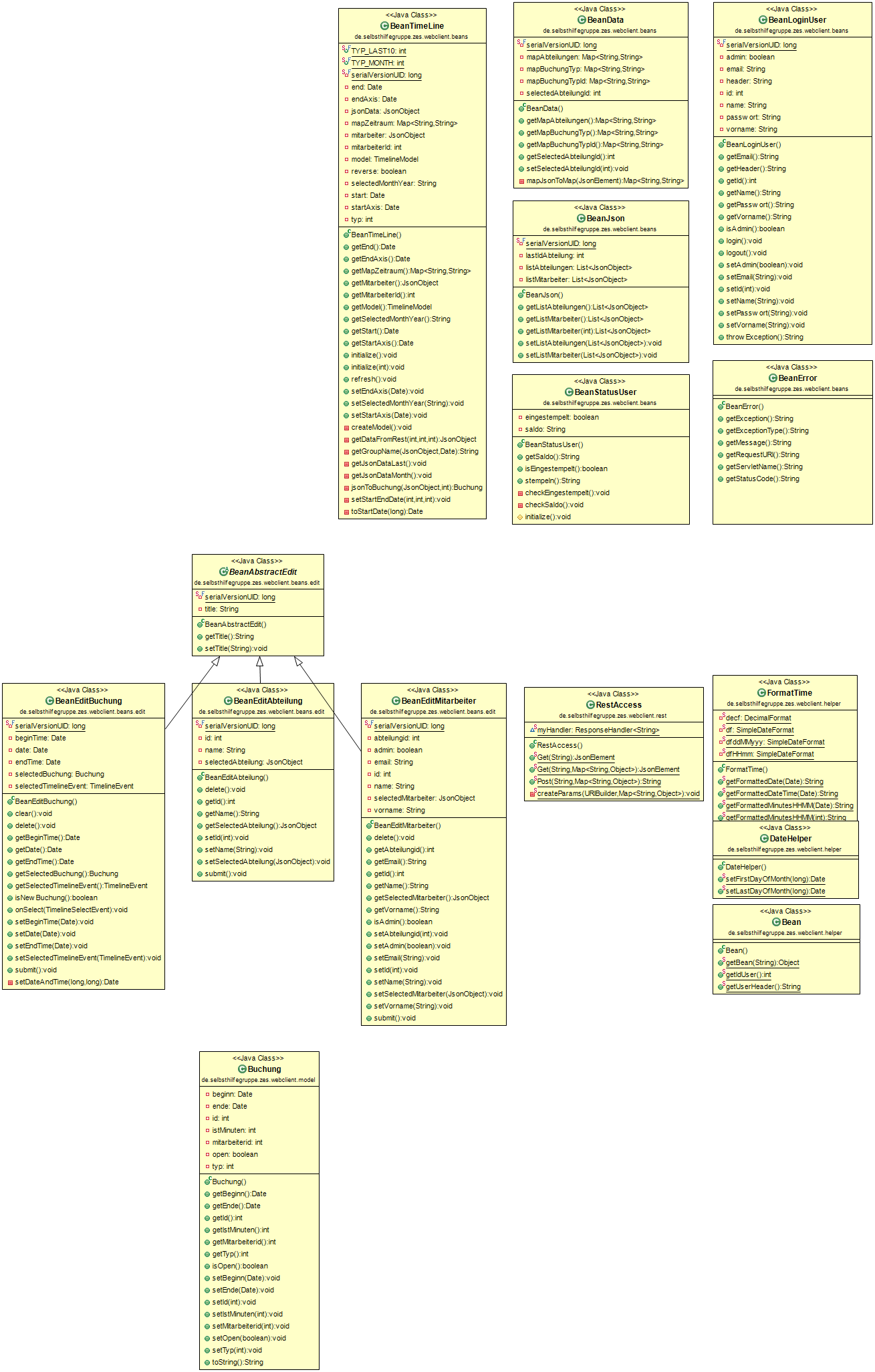
- Jersey-Core

- Jackson

* + 1. Probleme

Bei der Authentifizierung mit der externen Schnittstelle gab es Probleme. Ursprünglich war ein Spring Boot Projekt geplant welches eine Token basierte Authentifizierung umsetzt. Allerdings hat das Verbinden von zes\_core mit dem Webservice unter Verwendung von Spring nicht funktioniert. Daraufhin wurde das Projekt zes\_webservice als normales Webprojekt mit JAX-RS Unterstützung implementiert.

* 1. Web-Applikation
* Login --> 2 Rollen (Mitarbeiter, Admin)
* Je nach Rolle verschiedene Funktionen (siehe Anforderungen) vorhanden
  + 1. Techniken
    2. Aufbau



* + 1. Beschreibung
    2. Sprachen
    3. Bibliotheken
    4. Probleme

1. Umsetzung