Entwurfsdokumentation
Smart Building Solutions
-
Softwareprojekt Sommer 2021
Gruppe LMS2 UE1

Smart Building Solutions

Eddy Wu
Julius Daum
Lea Marie Schümann
Luca Anthony Schwarz
Natalie Kaufhold
Philipp Wieck
Till Kurzenberger
Yilmaz Atakan Kara

5. September 2021

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Dokumentaufbau . . . 1
1.2 Zweckbestimmung . . . 2
1.3 Entwicklungsumgebung . . . 2
1.3.1 Web und Backend . . . 2
1.3.2 Mobile Applikation . . . 3

2 Team-Aufteilung **5**

3 Komponentendiagramme **6**
3.1 Backend . . . 6
3.2 Web . . . 8
3.3 App . . . 10

4 Verteilungsdiagramm **11**

5 Klassendiagramme **13**
5.1 Backend . . . 13
5.1.1 Backend Datenmodell . . . 13
5.1.2 Backend Datenverarbeitung . . . 15
5.2 Web . . . 17
5.3 App . . . 19

6 Sequenzdiagramme **21**
6.1 Backend . . . 21
6.2 App . . . 22

Kapitel 1

Einleitung

1.1 Dokumentaufbau

In diesem Dokument wird die Implementierung der Software spezifiziert, wie sie im Pflichtenheft 2021 zum Projekt Smart Building Solutions vorgestellt wurde. Hierbei handelt es sich konkret um die **Weboberfläche,** die **mobile Applikation** und das entsprechende **Backend**, also den funktionalen Teil beider digitalen Produkte. Zu diesem Zweck wird zuerst in 1.2 der Einsatzbereich der Software erläutert und in 1.3 die Auswahl an Software oder Tools dargelegt, welche für die Implementierung verwendet werden.
Die **Teamaufteilung**, also der Zuständigkeitsbereich der einzelnen Entwickler, wird in Kapitel 2 festgehalten. In Kapitel 3 befinden sich die **Komponentendiagramme** des Backends 3.1, der Weboberfläche bzw. des Webservers 3.2 und der mobilen App 3.3 in dieser Reihenfolge. In Form einer Tabelle werden unterhalb der Abbildungen die jeweiligen Komponenten mit entsprechender Beschreibung näher erläutert. Anschließend folgt in Kapitel 4 ein **Verteilungsdiagramm**, welches die Elemente des Komponentendiagramms wieder aufgreift und das zukünftige Deployment des Systems widerspiegelt. Eine kurze Beschreibung darunter erläutert möglicherweise undeutliche Zusammenhänge. Kapitel 5 ist strukturiert in **Klassendiagramme** für das Backend, 5.1, Web 5.2 und die mobile-Applikation 5.3. Ergänzend verfügt jedes Klassendiagramm über eine Tabelle zur Beschreibung der konkreten Pfade und Intention der einzelnen Klassen. Abschließend werden die **Sequenzdiagramme** in Kapitel 6 aufgeführt und verfügen, wenn nötig, über eine schriftliche Beschreibung.

1.2 Zweckbestimmung

Smart Building Solutions hat als Ziel die entwickelte Software für Auftragnehmer und Auftraggeber im Baugewerbe zur Verfügung zu stellen. Die Software verwendet die vom Unternehmen bereitgestellten Vertrags- und Projektdaten, um den Status einzelner Projektbestandteile geeignet und den Anforderungen des Nutzers entsprechend in Form von Diagrammen und Statusbalken zu visualisieren.
Dabei soll für Mitarbeiter einzelner Organisationen eine Web-Oberfläche zur Verfügung stehen. Für Mitarbeiter, welche konkret am Bauprozess einzelner Leistungspositionen beteiligt sind, ist eine Oberfläche in Form einer mobilen Applikation bereitgestellt. Die mobile Anwendung verfügt über ausgewählte Funktionalitäten zur Rückmeldung und Visualisierung des Baufortschritts, sowie Statusänderungen der einzelnen Leistungspositionen. Änderungen an diesen Daten werden entsprechend mit der Webanwendung synchronisiert. Sie ist ausgelegt für die Nutzung auf einem geeigneten Endgerät unter Verwendung von Android 6 oder höher. Die Weboberfläche hingegen bietet im Webbrowser, bspw. Chrome oder Firefox, volle Funktionalität zur Darstellung von Diagrammen, die Möglichkeit Nutzer zu registrieren, Projekt- und Vertragsdaten einzusehen und den Status einzelner Leistungspositionen anzupassen.
Ein Systemadministrator, vom jeweiligen Bauunternehmen selbst ernannt, erhält die Rechte Mitarbeiter als Organisations-Administratoren auszuwählen. Zugriffsmöglichkeiten auf die einzelnen Funktionen von Webanwendung und mobiler Applikation sind entsprechend abhängig von der Position oder Benutzerrolle eines Mitarbeiters im konkreten Unternehmen und werden von einem zum Organisations-Administrator beauftragten Mitarbeiter des Unternehmens selbstständig vergeben. Die Registrierung einzelner Nutzer wird ebenfalls von diesem Mitarbeiter durchgeführt.

1.3 Entwicklungsumgebung

1.3.1 Web und Backend

Software	Version	URL
Eclipse	neueste (2021-06)	`https://www.eclipse.org/`
IntelliJ IDEA	neueste (2021. 2.1)	`https://www.jetbrains.com/de-de/idea/`
VSCode	neueste	`https://code.visualstudio.com/`
Java Development Kit	11.0.11	`https://www.oracle.com/de/java/technologies/javase-jdk11-downloads.html`
Gradle	7.1.1	`https://gradle.org/releases/`
Spring Boot	2.5.4	`https://mvnrepository.com/artifact/org.springframework.boot/spring-boot/2.5.4`
Spring Dependency Management	1.0.11	`https://plugins.gradle.org/plugin/io.spring.dependency-management`
Spring Boot Starter Data JPA	neueste (2.5.4)	`https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-data-jpa`
Spring Boot Starter Validation	neueste (2.5.4)	`https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-validation`

Weitergeführt auf der folgenden Seite

Spring Boot Starter Security	neueste (2.5.4)	https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-security
Spring Boot Starter Thymeleaf	neueste (2.5.4)	https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-thymeleaf
Spring Boot Starter Web	neueste (2.5.4)	https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-web
Thymeleaf Extras Springsecurity5	neueste (3.0.4.)	https://mvnrepository.com/artifact/org.thymeleaf.extras/thymeleaf-extras-springsecurity5
Spring Boot Devtools	neueste (2.5.4)	https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-devtools
H2 Database	neueste (1.4.200)	https://mvnrepository.com/artifact/com.h2database/h2
Spring Boot Starter Test	neueste (2.5.4)	https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-test
Spring Security Test	neueste	https://mvnrepository.com/artifact/org.springframework.security/spring-security-test
JUnit Jupiter API	5.7.2	https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api
Bootstrap	neueste (5.1.0)	https://getbootstrap.com/docs/5.1/getting-started/download/
Chart.js	neueste (3.5.1)	https://www.jsdelivr.com/package/npm/chart.js

Tabelle 1.1: Enwicklungsumgebung - Web-Oberfläche und Backend

1.3.2 Mobile Applikation

Software	Version	URL
Java Development Kit	11.0.11	https://www.oracle.com/de/java/technologies/javase-jdk11-downloads.html
Android Studio	neueste	https://developer.android.com/studio
Gradle	7.0.1	https://gradle.org/releases/
OkHttp	4.9.0	https://mvnrepository.com/artifact/com.squareup.okhttp3/okhttp
JUnit	ab Version 4	https://junit.org/junit4/
Espresso-Core	3.4.0	https://mvnrepository.com/artifact/androidx.test.espresso/espresso-core?repo=google

Weitergeführt auf der folgenden Seite

CameraX	1.0.1	`https://developer.android.com/training/camerax`
Room	2.3.0	`https://developer.android.com/jetpack/androidx/releases/room`

Tabelle 1.2: Enwicklungsumgebung - Mobile Applikation

Kapitel 2

Team-Aufteilung

Name	Zuständigkeit
Philipp	Android App
Yilmaz Atakan	Android App
Julius	Backend
Natalie	Backend
Eddy	Frontend: Web-Oberfläche
Lea Marie	Frontend: Web-Oberfläche
Luca Anthony	Frontend: Web-Oberfläche
Till	Frontend: Web-Oberfläche

Jedes Teammitglied ist für die Tests im jeweiligen Zuständigkeitsbereich verantwortlich.

Kapitel 3

Komponentendiagramme

3.1 Backend

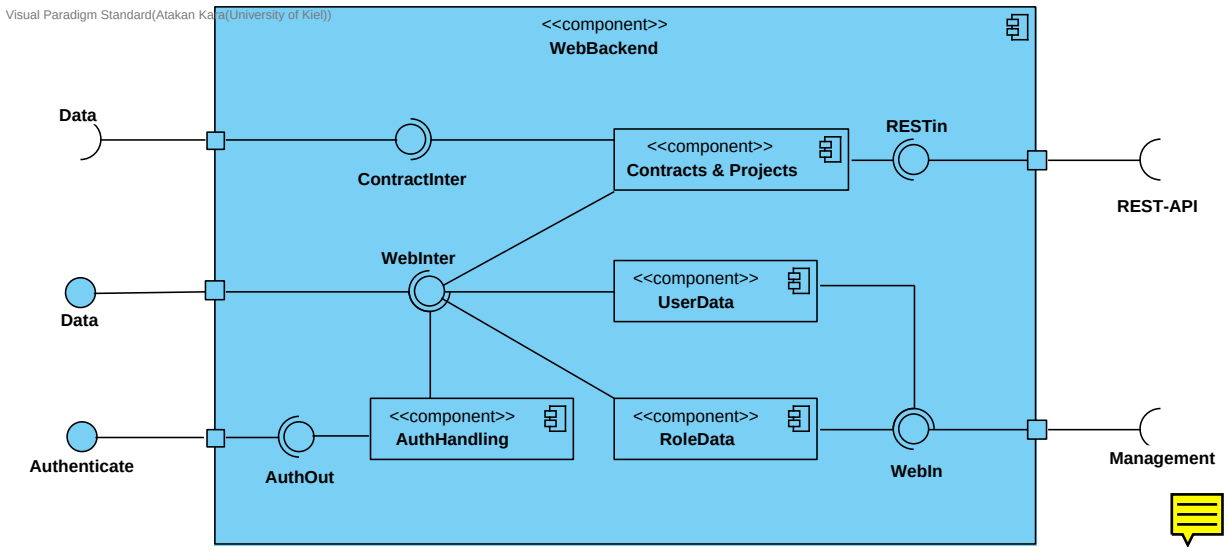

Abbildung 3.1: Komponentendiagramm - Backend

Komponente	Beschreibung
Authenticate	Gibt die Daten nach dem Authentifizierungsvorgang zurück.
AuthHandling	Verwaltet den Authentifizierungsvorgang.
Contracts & Projects	Speichert die Vertrags- und Projektdaten aus der **REST-API**.
Data	Das eingehende Interface empfängt die Anfragen des **WebServer**s für den Erhalt und die Veränderung bestimmter Vertrags- und Projektdaten. Das ausgehende Interface gibt, falls passende Berechtigungen bestehen, die abgefragten Daten, eine Bestätigung oder eine Fehlermeldung aus.
Management	Nimmt Daten vom **OrgAdmin** oder **SysAdmin** entgegen und übergibt diese über das Interface **WebIn** intern der **RoleData** und **UserDate**.
REST-API	Fragt die Daten von der **REST-API** ab und übergibt diese über das Interface **RESTin** der Komponente **Contracts & Projects**.
RoleData	Speichert die vom **OrgAdmin** erstellten Rollen und deren Berechtigungen ab.
UserData	Speichert Nutzerdaten, wie etwa Name, Passwort und assoziierte Rolle.

Tabelle 3.1: Tabelle - Komponentendiagramm-Backend

3.2 Web

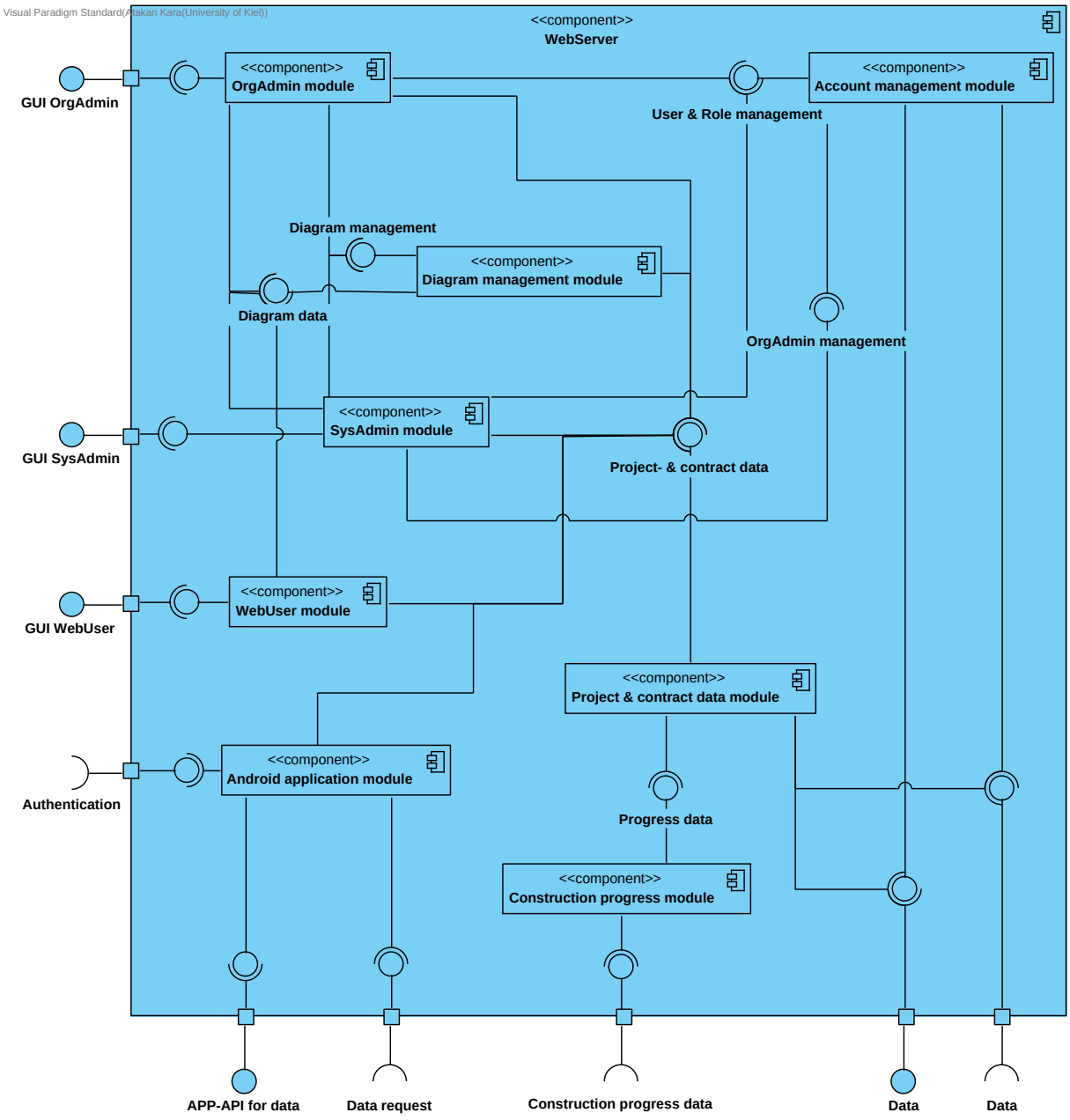

Abbildung 3.2: Komponentendiagramm - WebServer

Komponente	Beschreibung
Account management module	Anzeige und graphische Verwaltung von **OrgAdmin**s, **WebUser** und deren Rollen.
Android application module	Verwaltung der **App**. Leitet Authentifizierungs- und Datenanfragen weiter.
APP-API for data	Schnittstelle zur Datenübermittlung zur **App**.
Authentication	Schnittstelle zur Weiterleitung der Authentifizierungsanfrage des **AppUser**s zum **Backend**.
Construction progress module	Übermittlung von Baufortschrittsdaten.
Construction progress data	Eingang der Baufortschrittsdaten der App, wie z.B. Fotos und deren Beschreibungen.
Data	Ein- und Ausgang von Projekt-, Vertrags-, und Kontodaten zum und vom **Backend**.
Data request	Erhalt der Anfragen zur Datenübermittlung von der **App**.
Diagram management module	Verwaltung und Anzeige von Diagrammen.
Project & contract data module	Verwaltung und Weiterleitung von Projekt- und Vertragsdaten.
GUI OrgAdmin	Eingehende Daten und graphische Nutzeroberfläche des **OrgAdmin**s. Dieser muss eingeloggt sein.
GUI SysAdmin	Eingehende Daten und graphische Nutzeroberfläche des **SysAdmin**s. Dieser muss eingeloggt sein.
GUI WebUser	Eingehende Daten und graphische Nutzeroberfläche des **WebUser**s. Dieser muss eingeloggt sein.
OrgAdmin module	Vereinigt die Interaktionsmöglichkeiten des **OrgAdmin**s in einem zentralen Modul.
SysAdmin module	Vereinigt die Interaktionsmöglichkeiten des **SysAdmin**s in einem zentralen Modul.
WebUser module	Vereinigt die Interaktionsmöglichkeiten des **AppUser**s in einem zentralen Modul.

Tabelle 3.2: Tabelle - Komponentendiagramm-WebServer

3.3 App

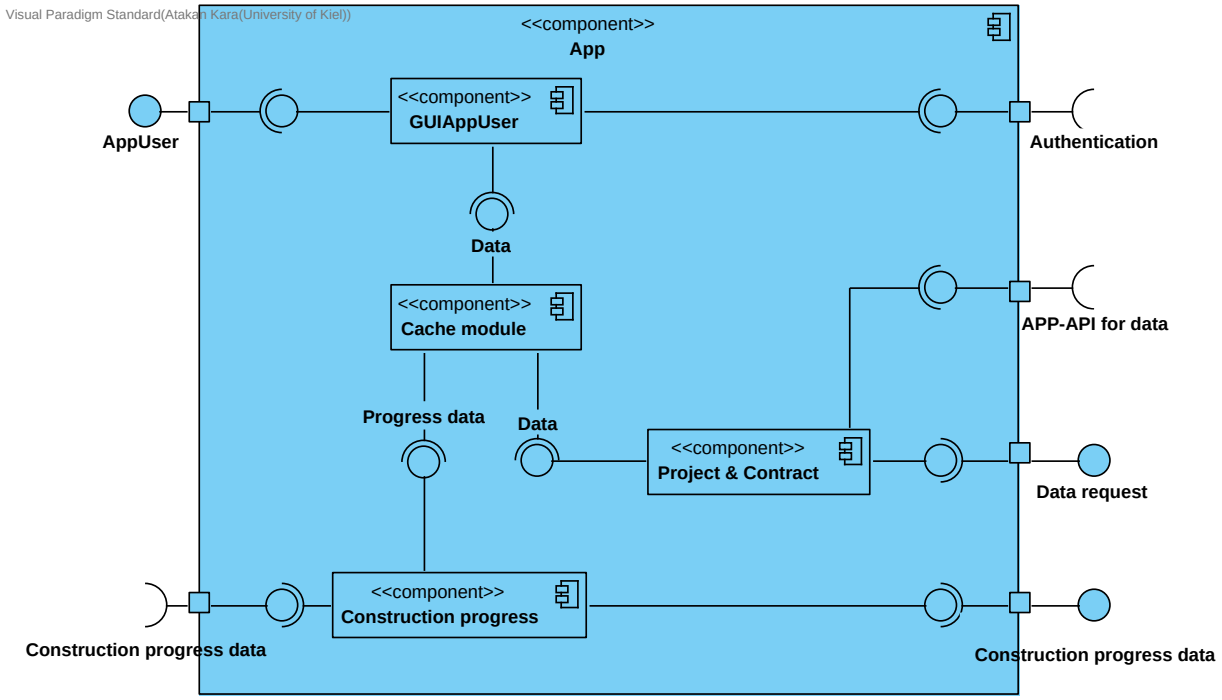

Abbildung 3.3: Komponentendiagramm - App

Komponente	Beschreibung
AppUser	Einsicht der Daten für den Nutzer der Applikation.
APP-API for data	Erhalt von Projekt- und Vertragsdaten vom **WebServer**.
Authentication	Übermittelt notwendige Daten zum **WebServer** zur Authentifizierung des **AppUser**s.
Cache module	Dient der Zwischenspeicherung der Projekt- und Vertragsdaten sowie des Baufortschritts.
Construction progress	Verwaltet den Baufortschritt.
Construction progress data	Das eingehende Interface erhält Daten zum Baufortschritt, wie z.B. Fotos und deren Beschreibungen. Diese fügt der **AppUser** hinzu. Das ausgehende Interface hingegen übermittelt diese an den **WebServer**.
Data request	Schnittstelle zur Anfrageübermittlung zum **WebServer**.
GUIAppUser	Modul zur Darstellung der graphischen Nutzeroberfläche der **App**.
Project & Contract	Verwaltet die Projekt- und Vertragsdaten.

Tabelle 3.3: Tabelle - Komponentendiagramm-App

Kapitel 4

Verteilungsdiagramm

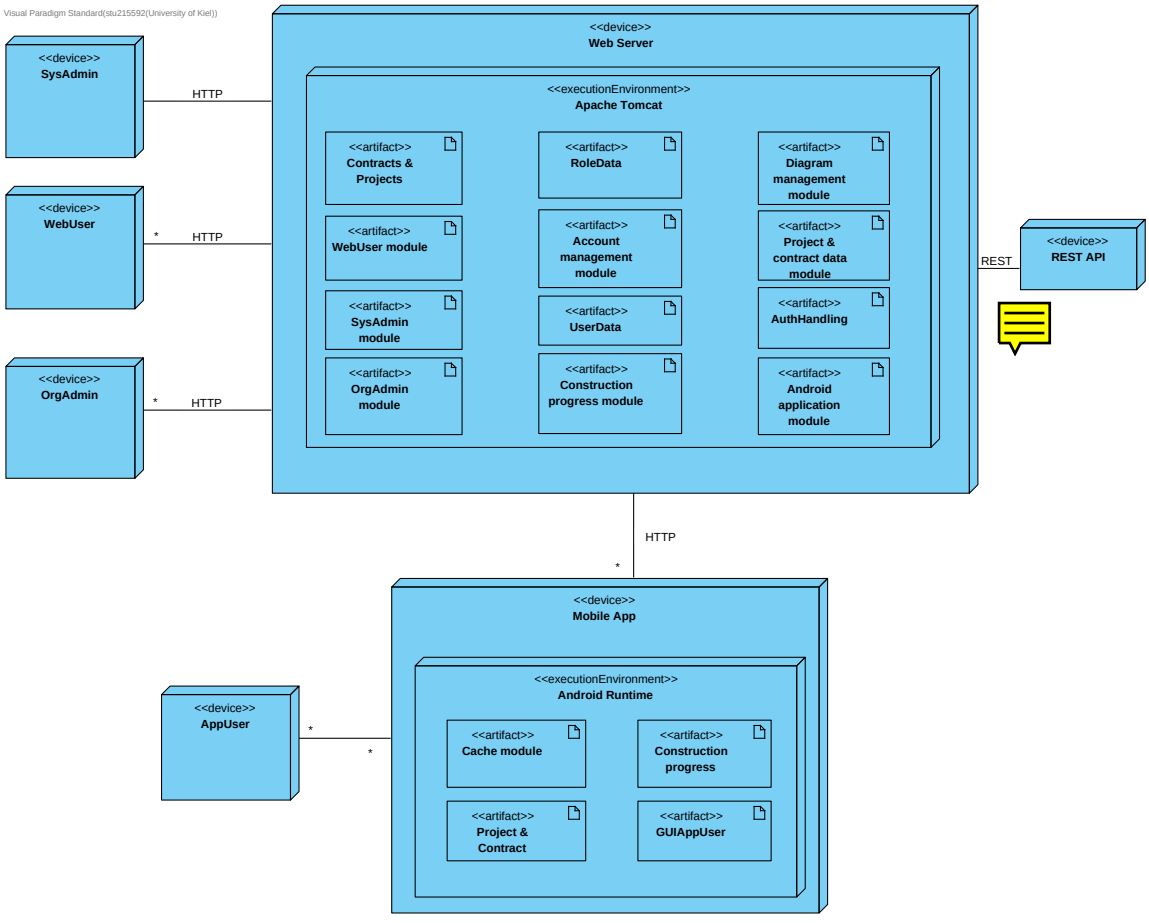

Abbildung 4.1: Verteilungsdiagramm

Die Applikation (Mobile App) kommuniziert mit dem Webserver (Web Server) mittels HTTP, um Projekt- und Vertragsdaten zu erhalten. Diese Schnittstelle ist unter dem Namen **App-API for data** jeweils im Komponentendiagramm von mobiler Applikation und Webserver eingezeichnet. Hierfür authentifiziert sich die Applikation per **POST-Request**, indem sie ein Tupel bestehend aus Benutzername und Passwort verschickt. Der Webserver antwortet mit einem **JSON Web Token** (**JWT**), welches in den folgenden Nachrichten jeweils mitgesendet wird. Dieser Vorgang stellt sicher, dass Ressourcen nur von Benutzern mit den entsprechenden Zugriffsrechten vom Server abgefragt werden können. Auf diese Weise bezieht die Applikation nun also Projekt- und Vertragsdaten vom Webserver, indem sie **GET-Request**s an den Server versendet. Darüber hinaus erfolgt das Aktualisieren des Status einer Leistungsposition ebenfalls per **POST-Request** an den Server.

Kapitel 5

Klassendiagramme

5.1 Backend

5.1.1 Backend Datenmodell

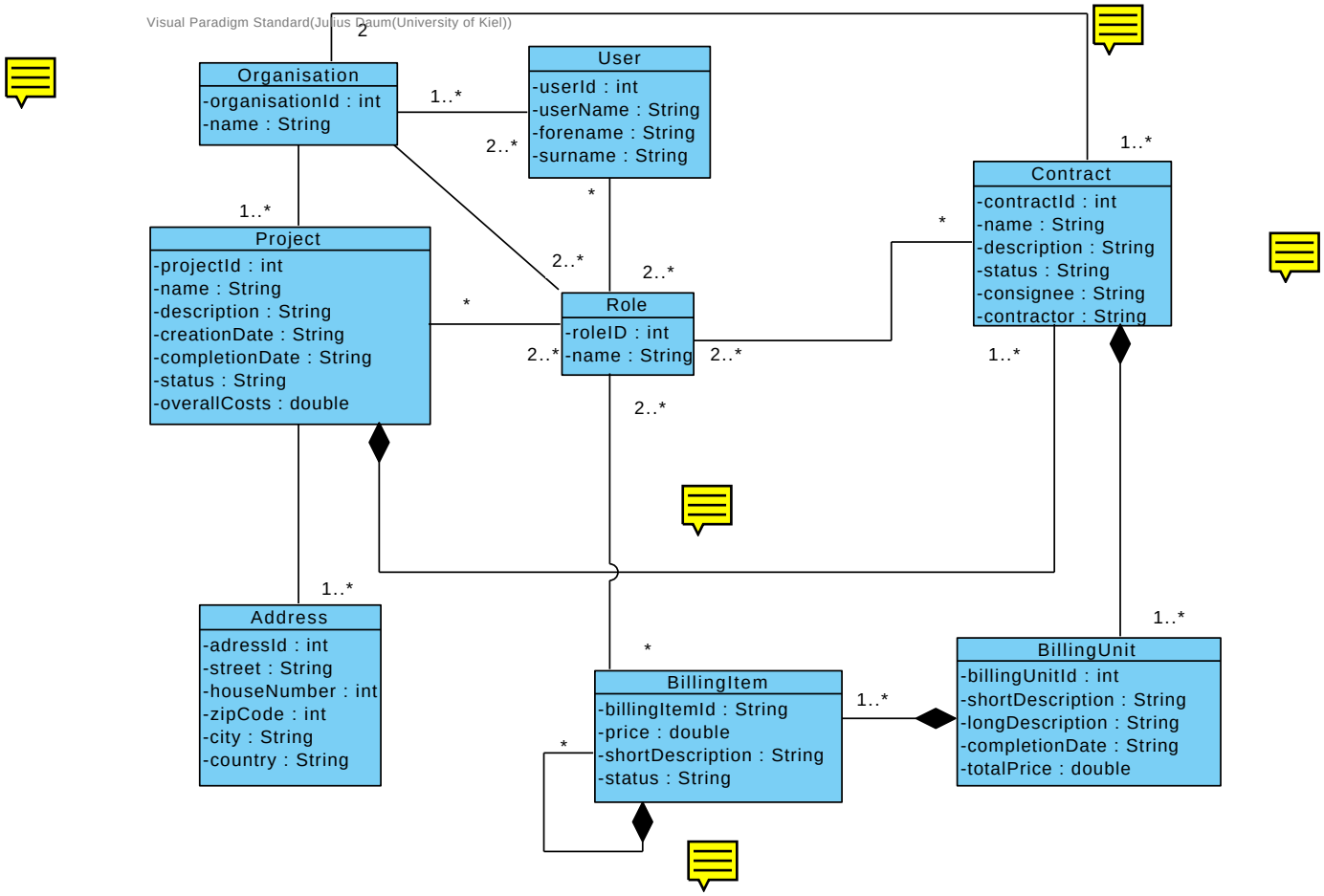

Abbildung 5.1: Klassendiagramm - Backend - Datenmodell

Dieses Klassendiagramm enthält die Entitäten der Datenbank, jede Klasse entspricht einer Entität. Die Klassen **BillingItem**, **Contract**, **User**, **Organisation** und **Project** stehen jeweils mit mindestens 2 Instanzen der Klasse **Role** in Beziehung, da der **SysAdmin** Mitglied jeder Organisation ist und somit alle Projekte, User, Verträge und Leistungspositionen einsehen kann. Zusätzlich existiert zu jeder Organisation mindestens ein **OrgAdmin**, der ebenfalls auf alle seiner Organisation zugeordneten Verträge, Projekte und Leistungspositionen zugreifen kann.

Klassenname	**Aufgabe**
Address	Die Adresse des Projekts.
BillingItem	Die Leistungsposition eines Vertrages.
BillingUnit	Eine Gruppierung von Leistungspositionen.
Contract	Ein Vertrag, welcher zwischen zwei Parteien geschlossen wird.
Organisation	Eine Gruppierung von **User**n.
Project	Ein Projekt mindestens einer Organisation, welches mehrere Verträge enhalten kann.
Role	Nutzerrollen mit verschiedenen Rechten.
User	Mitarbeiter mindestens einer Organisation.

Tabelle 5.1: Klassenbeschreibung - Backend - Datenmodell

5.1.2 Backend Datenverarbeitung

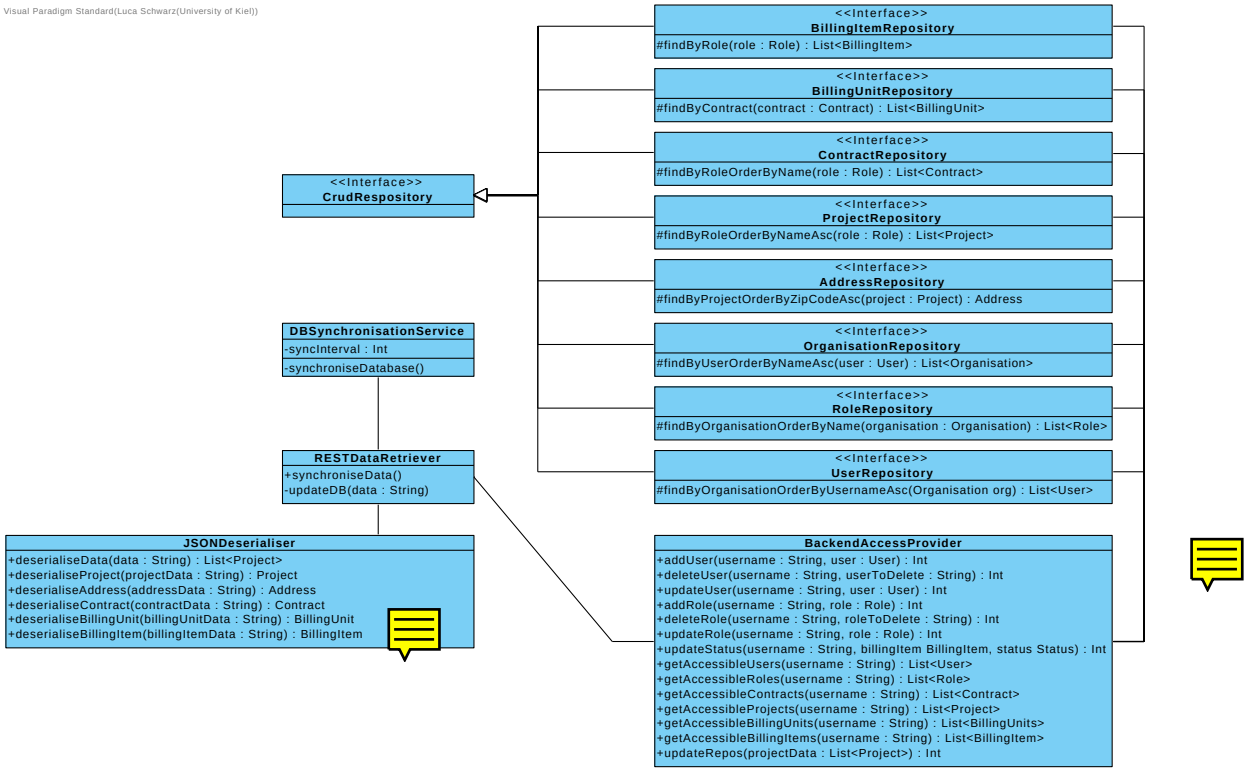

Abbildung 5.2: Klassendiagramm - Backend - Datenverarbeitung

Klassenname	Aufgabe
CrudRepository	Teil des Spring Frameworks. Wird hier genutzt, um die Datenbank in Form von erbenden Repositories darzustellen.
*Repository	Verwaltet die jeweilige Datenmodellklasse. Bietet spezielle Zugriffsfunktionen für eine einfachere Nutzung.
BackendAccessProvider	Bietet die eigentliche Funktionalität des Backends innerhalb der Server-Applikation an. Über diese Klasse wird der gesicherte Zugriff auf die gespeicherten Daten sichergestellt und die einzelnen Daten-Repositories werden vor dem Nutzer verborgen. Alle Methoden verlangen einen Nutzernamen als Parameter, um festzustellen, welche Daten zurückgegeben werden dürfen. Dafür werden intern die Rollen verwendet. So soll ein Nutzer z.B. nur Zugriff auf Verträge bekommen, für die er auch über eine entsprechende Rolle verfügt. Ansonsten werden leere Listen und auch Fehlercodes zurückgegeben, welche dann z.B. vom Frontend entsprechend verarbeitet werden können. Die Controller-Klassen des Frontends haben folglich Zugriff auf den **BackendAccessProvider**.
DBSynchronisationService	Dienst, welcher nach einem Zeitintervall eine Synchronisation zwischen der Datenbank von adesso und der lokalen veranlasst. Die eigentliche Synchronisation führt die Klasse **RESTDataRetriever** durch.
RESTDataRetriever	Fragt die Daten über die **REST-API** von adesso ab und deserialisiert diese anschließend. Die so erhaltenen Klassen werden abschließend in die Datenbank über den **BackendAccessProvider** eingepflegt.
JSONDeserialiser	Konvertiert **JSON**-Strings in die entsprechenden Modellklassen. Dies findet Verwendung, wenn die Daten über die **REST-API** von adesso abgefragt werden.

Tabelle 5.2: Klassenbeschreibung - Backend - Datenverarbeitung

5.2 Web

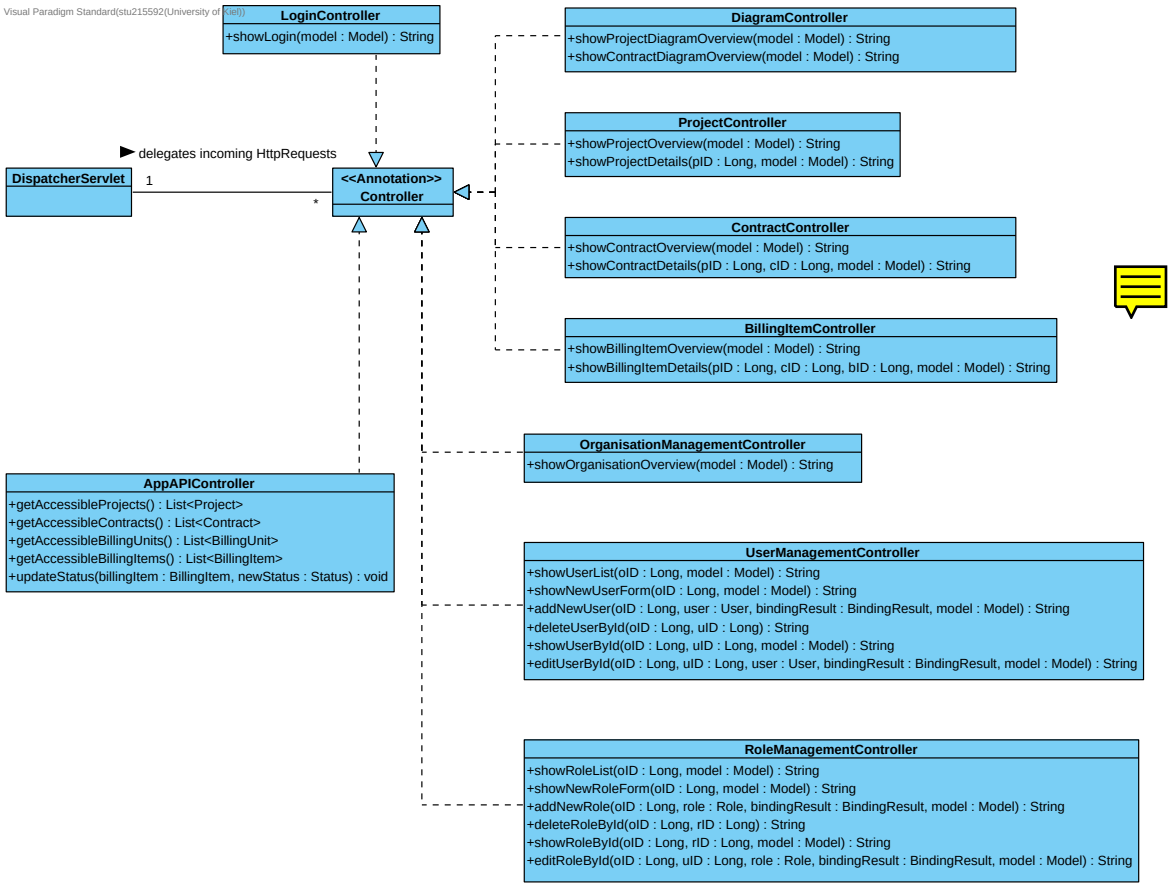

Abbildung 5.3: Klassendiagramm - Web

Controller verarbeiten HTTP-Requests zu bestimmten Pfaden. Die Pfade werden von je einem **Controller** nach Gebiet gruppiert verarbeitet. Diese werden in der folgenden Tabelle unter Aufgabe aufgeführt. Die Identifikationsnummern oID (Organisation), uID (Nutzer), rID (Rolle), pID (Projekt), cID (Vertrag) und bID (Leistungsposition) sind in einigen Pfaden direkt integriert.

Klassenname	Aufgabe
DispatcherServlet	Teil des Spring Frameworks, leitet die HTTP-Requests an den jeweils zuständigen **Controller** weiter.
LoginController	/login → Login-Seite
OrganisationManagementController	/organisation_overview → Management von Organisationen und deren **OrgAdmin**s, nur der **SysAdmin** hat hierauf Zugriff

Weitergeführt auf der folgenden Seite

UserManagementController	/organisation/{oID}/user_management → Management der **WebUser** einer Organisation /organisation/{oID}/user_management/user_new → Hinzufügen eines **WebUser**s zu einer Organisation /organisation/{oID}/user_management/user/{uID} /user_edit → Bearbeiten eines **WebUser**s einer Organisation
RoleManagementController	/organisation/{oID}/role_management → Management der Rollen einer Organisation /organisation/{oID}/role_management/role_new → Hinzufügen einer Rolle zu einer Organisation /organisation/{oID}/role_management/role/{rID} /role_edit → Bearbeiten einer Rolle einer Organisation
ProjectController	/project_overview → Zeigt alle Projekte an, für welche der **WebUser** die nötigen Berechtigungen hat /project/{pID}/show → Zeigt die Verträge des Projekts an, für welche der **WebUser** die nötigen Berechtigungen hat
ContractController	/contract_overview → Zeigt alle Verträge an, für welche der **WebUser** die nötigen Berechtigungen hat /project/{pID}/contract/{cID}/show → Zeigt die Leistungspositionen des Vertrags an, für welche der **WebUser** die nötigen Berechtigungen hat
BillingItemController	/billing_item_overview → Zeigt alle Leistungspositionen an, für welche der **WebUser** die nötigen Berechtigungen hat /project/{pID}/contract/{cID}/billing_item/{bID}/show → Zeigt Details zur Leistungsposition an, falls der **WebUser** die nötigen Berechtigungen hat
DiagramController	/project_diagram_overview → Zeigt alle Diagramme zu Projekten an, für welche der **WebUser** die nötigen Berechtigungen hat /contract_diagram_overview → Zeigt alle Diagramme zu Verträgen an, für welche der **WebUser** die nötigen Berechtigungen hat

Tabelle 5.3: Klassenbeschreibung - Web

5.3 App

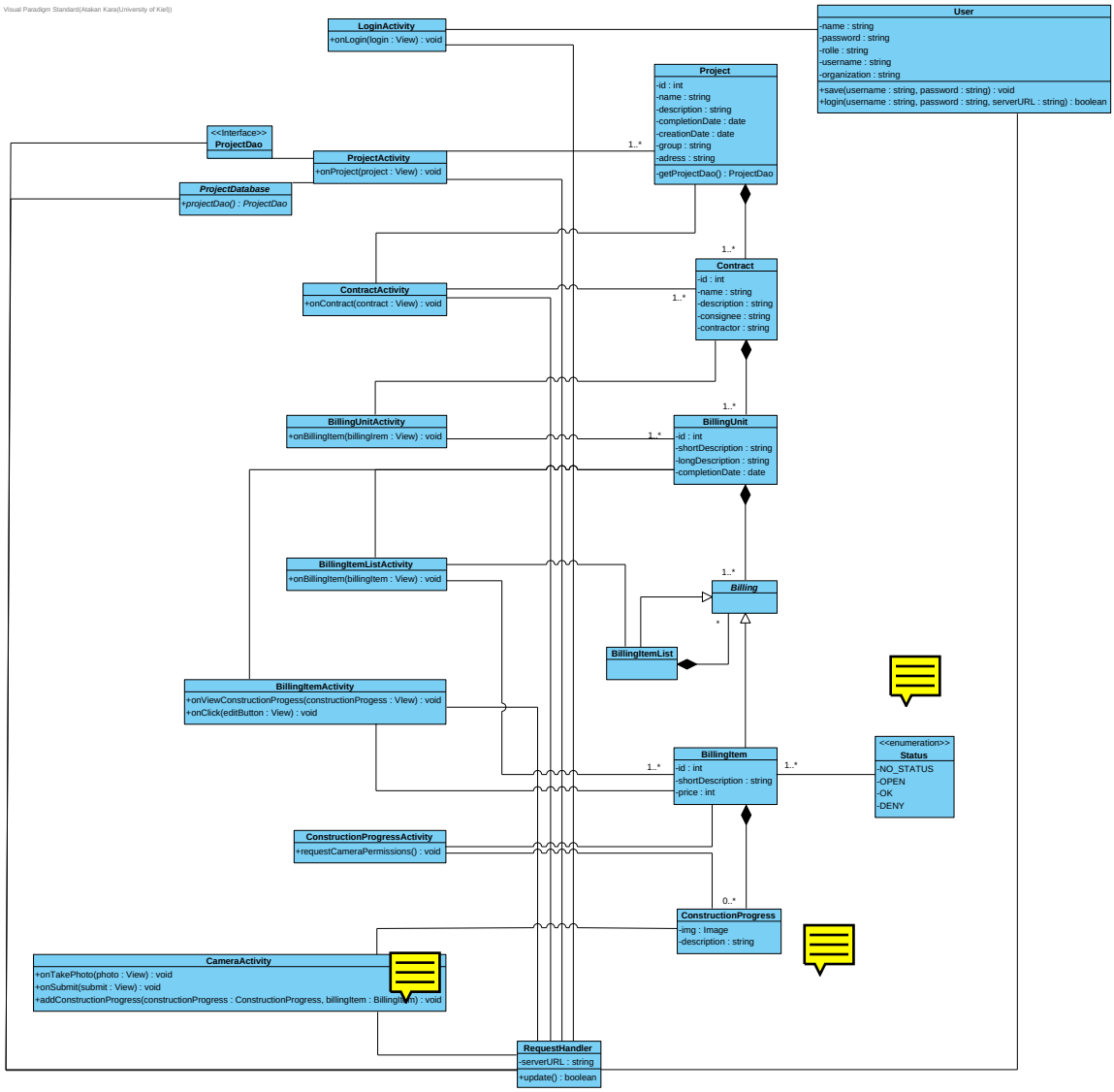

Abbildung 5.4: Klassendiagramm - App

Alle Activity-Klassen enthalten neben den angegebenen auch die folgenden Methoden: onLogout(), onBack(), onCreate(), onStart(), onResume(), onPause(), onStop(), onRestart(), onDestroy(). Außerdem wird mittels der abstrakten Klasse Billing das Composite-Strukturmuster verwendet.

Klassenname	Aufgabe
Project	Bauplan eines Projekts mit den jeweiligen Attributen.
Contract	Bauplan eines Vertrages mit den jeweiligen Attributen.
BillingUnit	Bauplan einer Leistungseinheit mit den jeweiligen Attributen.
ConstructionProgress	Bauplan einer Baufortschritts-Klasse mit den jeweiligen Attributen.
LoginActivity	Verwaltung des Login-Bildschirms.
ProjectActivity	Verwaltung des Projekt-Bildschirms.
ContractActivity	Verwaltung des Vertags-Bildschirms.
BillingUnitActivity	Verwaltung des Leistungspositioneinheits-Bildschirms.
BillingItemListActivity	Verwaltung des Leistunspositionlisten-Bildschirms.
BillingItemActivity	Verwaltung des Leistunspositions-Bildschirms.
ConstructionProgressActivity	Verwaltung des Baufortschritts-Bildschirms.
CameraActivity	Verwaltung des Kamera-Bildschirms.
RequestHandler	Verwaltung der Netzwerkanfragen aller Klassen.
ProjectDao	Datenzugriffsobjekt mit Anfragen zur Interaktion mit der Projektdatenbank.
ProjectDatabase	Room-Datenbank für Projekte.
Status	Aufzählung, welche den Status modelliert.
User	Nutzer der App.
Billing	Abstrakte Klasse, dient dem Composite-Strukturmuster.
BillingItemList	Kompositum, welches **BillingItem**s enthält.
BillingItem	Blatt des Kompositums, Bauplan einer Leistungsposition mit den jeweiligen Attributen.

Tabelle 5.4: Klassenbeschreibung - App

Kapitel 6

Sequenzdiagramme

6.1 Backend

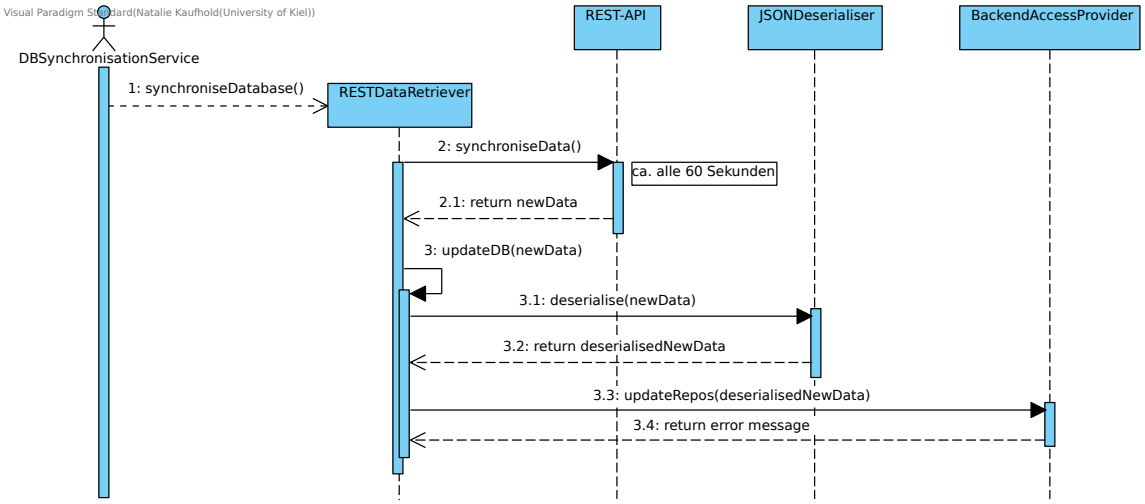

Abbildung 6.1: Sequenzdiagramm - Backend

Die neuen Daten werden von der **REST-API** als **JSON** zur Verfügung gestellt. Um die Daten aus der **JSON**-Datei in die Datenbank zu übertragen, muss diese Datei in seine Einzelteile zerlegt werden: Dafür wird eine Instanz des **JSONDeserialiser** verwendet, um die Daten etwa in eine Liste von Projekten zu konvertieren. Diese wird dann von einer Instanz des **BackendAccessProviders** in die Datenbank über die von Spring gestellten Repositories übertragen. Die **REST-API** wird vom Kunden gestellt und verwaltet.

6.2 App

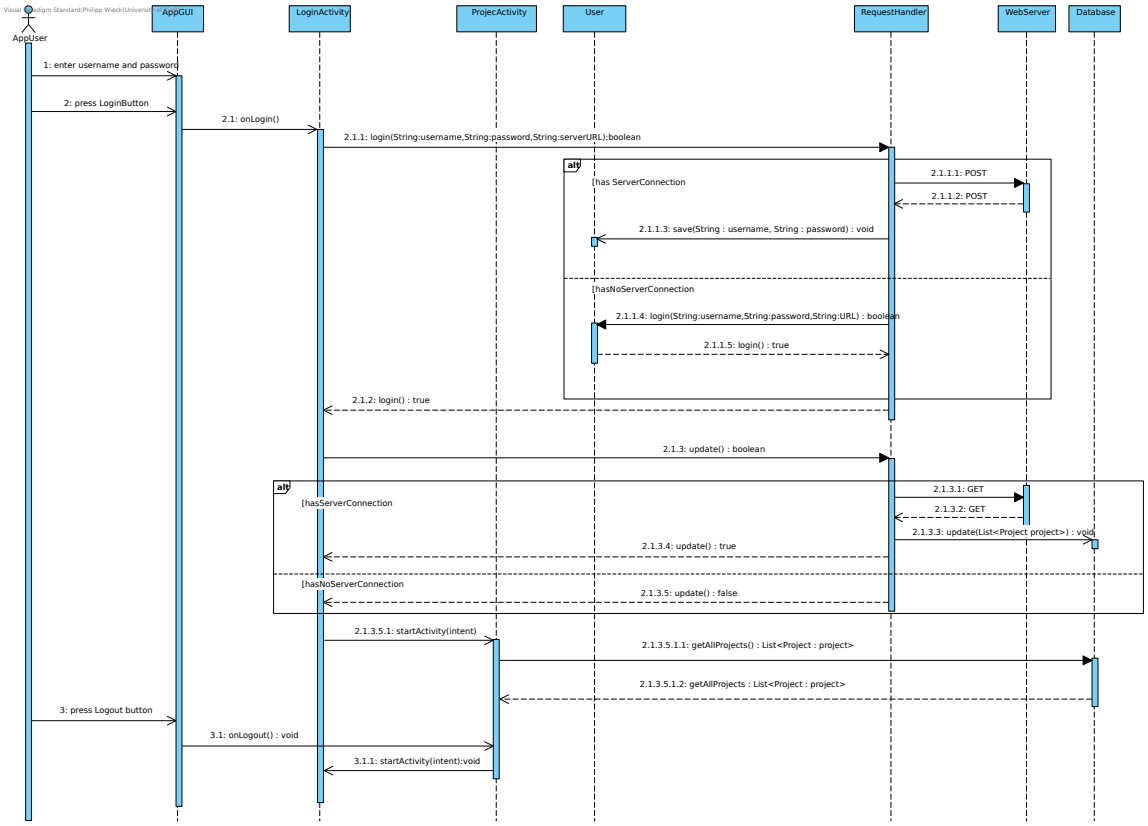

Abbildung 6.2: Login und Logout - App

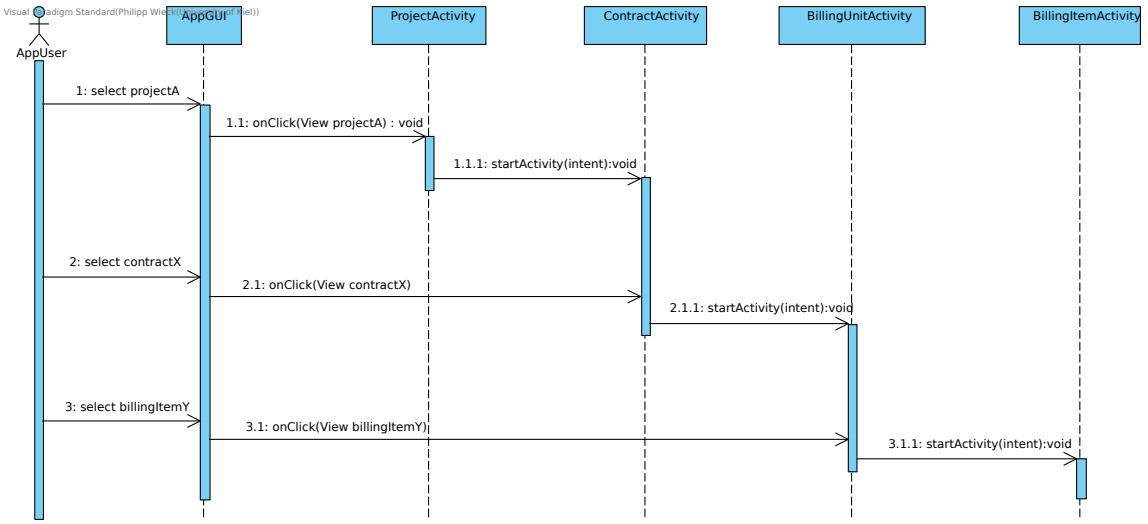

Abbildung 6.3: Auswählen einer Leistungsposition - App

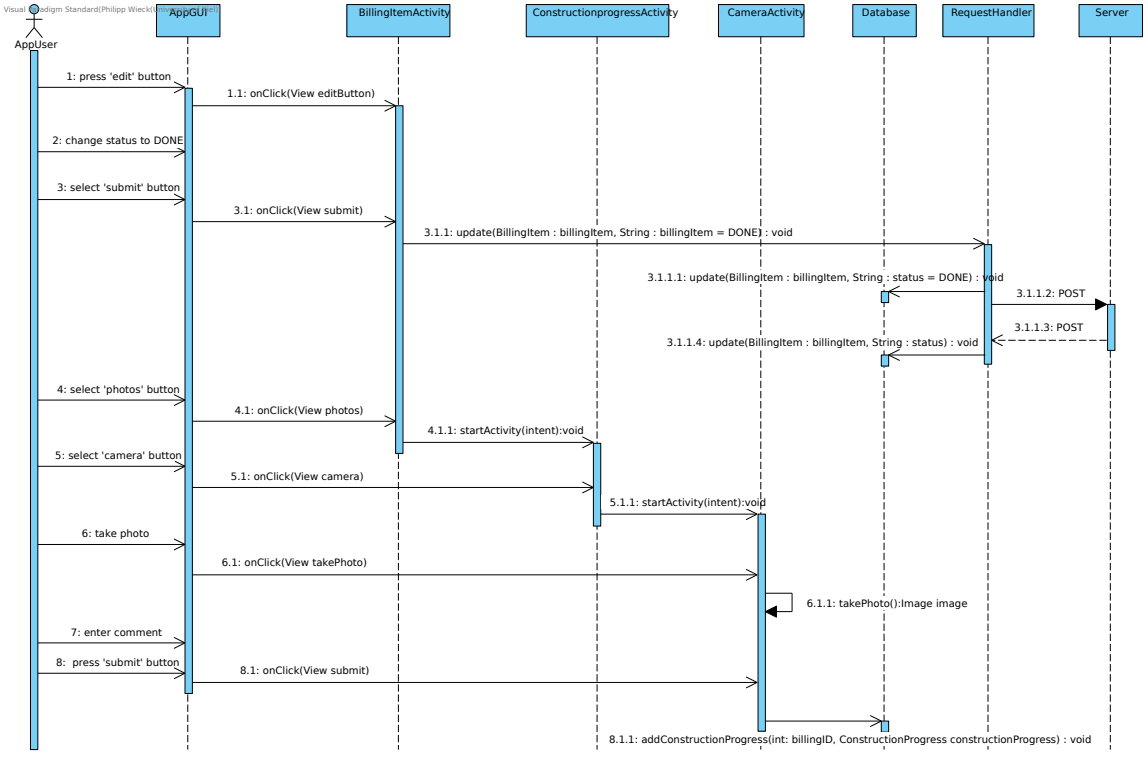

Abbildung 6.4: Verändern des Status und Hinzufügen eines Fotos inkl. Kommentar - App