File Share Application
Konzeptbericht (Applikationsentwicklung)

Auftraggeber	Gibb Bern
Projektleiter	Romano Brentani
Autor	Yaël Wehrli, Romano Brentani, Von Werdt Thomas, Wiktor Kepczynski
Klassifizierung	*GEHEIM*
Status	*In Arbeit, Genehmigt*

Inhaltsverzeichnis

1 Einleitung

Im folgenden Dokument wird näher erläutert, wie das Projekt konkret umgesetzt werden soll und welche Anforderungen das Endprodukt abdecken wird.
Im ersten Teil wird beschreiben, in welchem Geschäftsumfeld die im Rahmen dieses Projektes zu realisierende Applikation eingesetzt werden könnte. Weiter werden die einzelnen Anforderungen in Form von möglichen in der Applikation auftretenden Anwendungsfälle dargelegt. Anschließend werden die Architektur sowie die eingesetzten Technologien der Applikation erläutert. Hierbei wird beschrieben, nach welchem Prinzip die Architektur aufgebaut und welche Architekturschicht mit welcher Technologie realisiert wird.
In einem letzten Teil werden weitere Schnittstellen der Applikation beschrieben, Testfälle festgelegt, um die einzelnen in die Applikation zu integrierenden Funktionalitäten zu testen und in Form eines Diagramms die in unterschiedlichen Phasen zu erledigenden Arbeitspakte abgebildet.

2 Systemanforderungen

Die Verwendung der Applikation beschränkt sich aufgrund ihrer Funktionalitäten und Anforderungen (vgl. Anwendungsfälle 3.2), nicht auf ein konkretes Geschäftsfeld, sondern kann sowohl von Unternehmen, schulischen Institutionen und Privatpersonen zugleich verwendet werden.
Das Bedürfnis, digitale Dokumente zu teilen, existiert sowohl unter Privatpersonen als auch in internen Unternehmenskreisen oder in öffentlichen und privaten Bildungsstätten.
Die Tatsache, dass digitale Dokumente auch zum Verkauf angeboten werden können, macht die Applikation insbesondere für Privatpersonen und Unternehmen attraktiv. Andererseits ermöglicht die Applikation, den Zugriff auf Dateien mithilfe von Gruppen einzuschränken. Schulen sowie andere öffentliche Bildungsinstitutionen zeigen ein erhöhtes Interesse an solchen Kollaborativen-Applikationen oder setzen diese bereits öfters ein.

3 Fachliche Entitätstypen

Die in der Applikation zentralste Beziehung besteht zwischen einer Person und der von ihr geteilten Dateien. Eine geteilte Datei kann immer einem Besitzer zugeordnet werden. Dateien ohne Besitzer existieren somit nicht. Einer Person steht es jedoch frei, mehrere Dateien hochzuladen. Aus diesem Grund kann eine Person mehrere Dateien besitzen.
Dateien weisen einen eindeutigen Dateityp auf. Umgekehrt kann ein Typ vermehrt in der Applikation durch Dateien vertreten werden.
Weiter können Dateien privat, das heißt, nur für eine bestimmte Zielgruppe oder Person einsehbar sein oder einen öffentlichen Status aufweisen, und somit für alle zugänglich sein.
Personen mit einem registrierten Account können Mitglieder einer Gruppe sein. Diese Gruppen können auf Dateien Zugriffsberechtigungen erhalten. Es können auch Zugriffsberechtigungen auf Dateien für Einzelpersonen vergeben werden.
Eine Gruppe kann von maximal einem Benutzer erstellt werden und weist anschließend somit einen Besitzer auf.
Jeder registrierte Benutzer ist berechtigt, Mitglied mehrerer Gruppen zu sein. Auch steht jedem Applikationsnutzer, welcher eine Benutzerregistration abgeschlossen hat, frei, so viele Gruppen zu erstellen, wie er das wünscht.
Das unten abgebildete Diagramm stellt die unter den in der Applikation vorhandenen Akteuren bestehenden Beziehungen im Detail dar.

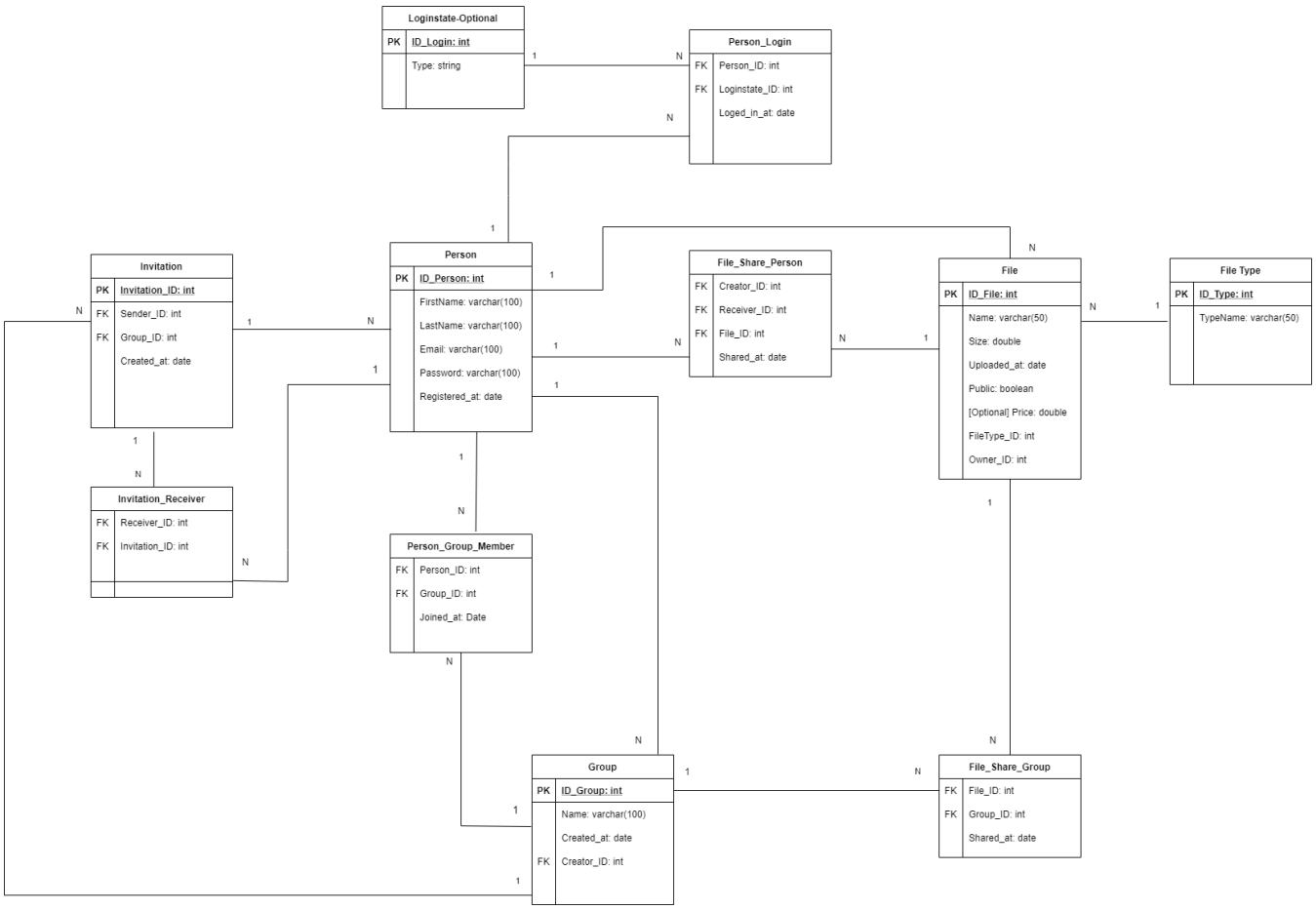

Abbildung 1: Entitätsdiagramm der fachlichen Entitäten

4 Anwendungsfälle / Product-Backlog

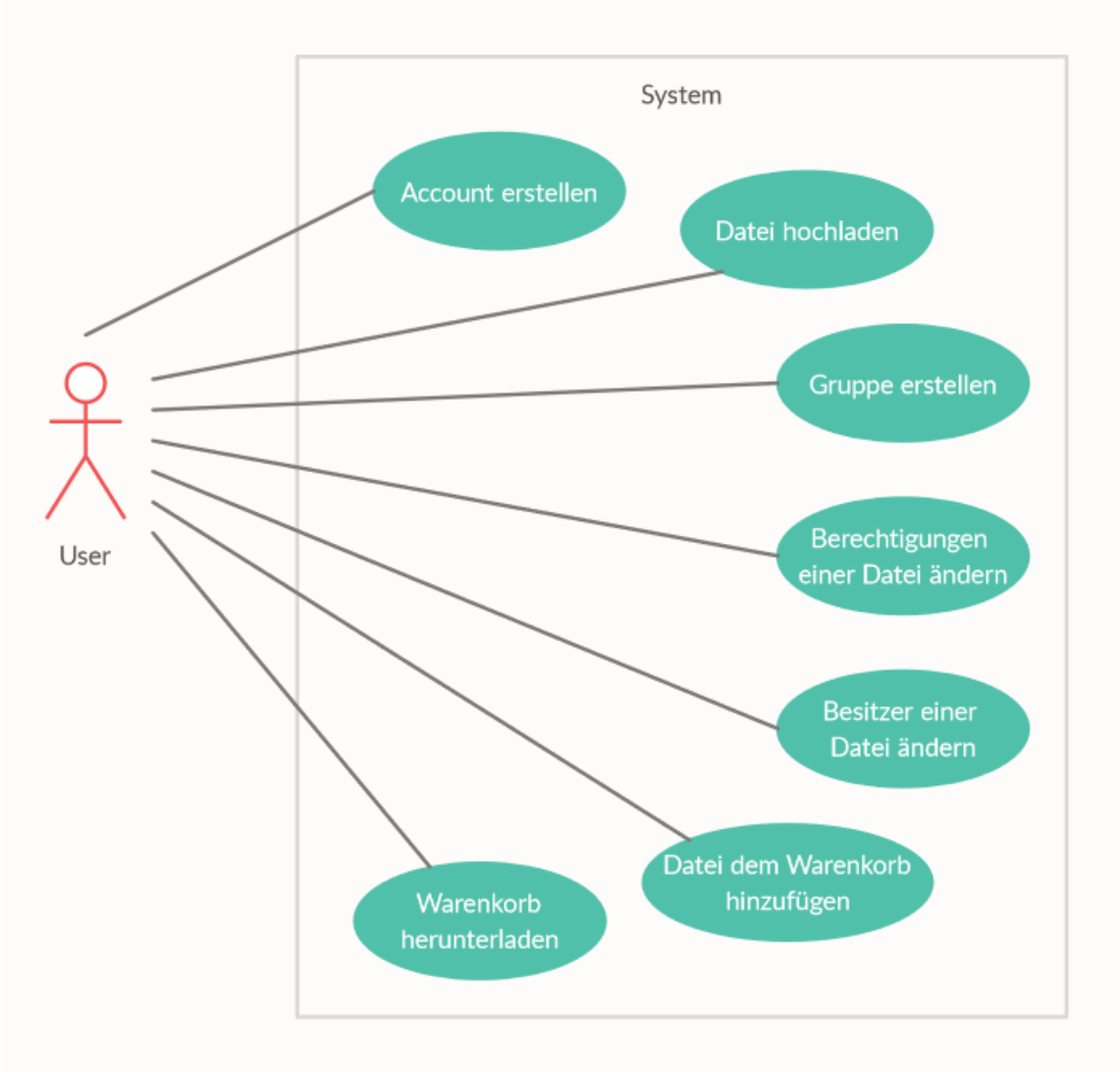

Abbildung 2: Use Case Diagramm

4.1 Use Cases

1) Anwendungsfall "Account erstellen”	
Kurzbeschreibung	Ein Benutzer erstellt einen Account, um die Anwendung nutzen zu können
Akteure	Benutzer
Vorbedingungen	Der Account existiert noch nicht.
Ablauf	1. Der Benutzer ruft die Webseite auf. 2. Das System präsentiert die Login-Seite, da der User nicht authentifiziert ist. 3. Der User klickt auf den “Ich habe noch keinen Account”-Link. 4. Das System navigiert zur Registrierungsseite. 5. Der User gibt den gewünschten Benutzernamen, seine Emailadresse und das Passwort, sowie eine Wiederholung dessen ein. 6. Der User klickt auf den “Registrieren”-Button.

	7. Das System validiert die Angaben. Ist die Validation erfolgreich, wird der Account erstellt. 8. Nach einer erfolgreichen Registrierung wird der User direkt angemeldet und zur Startseite weitergeleitet.
Resultat	In der Datenbank wurde der neue Account eingetragen.
Ausnahmen	• Die Validierung der Angaben ist nicht erfolgreich. (7) • Ein Account mit dem Benutzernamen existiert bereits. (7)

2) Anwendungsfall "Datei hochladen”	
Kurzbeschreibung	Der User lädt eine Datei hoch.
Akteure	User
Vorbedingungen	Der User hat einen Account und ist angemeldet.
Ablauf	1. Der User ruft die Webseite auf. 2. Das System präsentiert die Startseite. 3. Der User klickt auf “neue Datei hochladen”. 4. Das System lädt die “Datei hochladen”-Seite. 5. Der User wählt die Datei vom lokalen Gerät aus. 6. Der User gibt die gewünschten Berechtigungen für die Datei an. Wird die Datei privat geteilt, müssen noch die berechtigten Personen / Gruppen angegeben werden. 7. Er verleiht der Datei falls gewünscht einen Preis. 8. Der User klickt auf “Hochladen”. 9. Das System validiert die Datei und die Eingaben. Bei einer erfolgreichen Validierung wird die Datei hochgeladen. 10. Der Benutzer wird zur Startseite zurückgeleitet
Resultat	Es wurde ein neues Datei-Objekt mit der Datei und den dazugehörigen Angaben erstellt und in der Datenbank gespeichert. Die Datei befindet sich nun in der Dateiübersichtsliste des entsprechenden Benutzeraccounts. Ist die Datei nicht öffentlich, ist sie nur für die berechtigten Personen sichtbar.
Ausnahmen	• Der User hat nicht mehr genug Speicherplatz zur Verfügung. (9)

3) Anwendungsfall "Gruppe erstellen”	
Kurzbeschreibung	Der User erstellt eine neue Gruppe
Akteure	User
Vorbedingungen	Es existieren mindestens zwei User. Der Akteur ist angemeldet.
Ablauf	1. Der User ruft die Webseite auf. Das System präsentiert die Startseite. 2. Der User wechselt zur Gruppenseite. 3. Der User klickt auf „Gruppe erstellen“. 4. Das System lädt die „neue Gruppe“-Seite. 5. Der User gibt der Gruppe einen Namen. 6. Der User gibt die gewünschten Mitglieder ein. 7. Der User klickt auf “Gruppe erstellen”. 8. Das System validiert die Eingaben. Nach einer erfolgreichen Validierung wird die Gruppe mit den gewünschten Mitgliedern erstellt. 9. Der User wird zur Gruppenseite zurückgeleitet.
Resultat	In der Datenbank wird eine Gruppe nach Angaben des Users erstellt. Die Gruppe wird auf der Gruppenseite angezeigt.
Ausnahmen	• Eine Gruppe mit dem Namen existiert bereits (8).

4) Anwendungsfall "Zugriffsberechtigungen einer Datei ändern"	
Kurzbeschreibung	Der Benutzer ändert die Zugriffsberechtigungen einer seiner geteilten Dateien.
Akteure	User
Vorbedingungen	Der User hat einen Account und ist angemeldet und hat bereits eine Datei hochgeladen.
Ablauf	1. Der User ruft die Webseite auf. 2. Das System präsentiert die Startseite. 3. Der User wählt in der die gewünschte, ihm gehörende Datei aus. 4. Der User klickt auf „Berechtigungen ändern". 5. Das System präsentiert ein kleines Formular zum Ändern der Berechtigungen. 6. Der User kann die Datei von privat auf öffentlich ändern oder umgekehrt. Bei einer privaten Datei kann er die berechtigten Personen / Gruppen auswählen. 7. Der User klickt auf „Speichern".
Resultat	Das System passt den Status und die Zugriffsberechtigungen in der Datenbank an.
Ausnahmen	• Die Datei gehört nicht dem Benutzer und kann somit nicht in ihrer Zugriffsberechtigungen verändert werden. (5)

5) Anwendungsfall "Besitzer einer Datei ändern"	
Kurzbeschreibung	Der Benutzer weist eine ihm gehörende Datei einem neuen Besitzer zu.
Akteure	User
Vorbedingungen	Der User hat einen Account, ist angemeldet und hat bereits eine Datei hochgeladen. Nebst dem Akteur existiert ein zweiter User.
Ablauf	1. Der User ruft die Webseite auf. 2. Das System präsentiert die Startseite. 3. Der User wählt in der die gewünschte, ihm gehörende Datei aus. 4. Der User klickt auf „Besitzer wechseln". 5. Das System präsentiert ein kleines Formular zum Wechseln des Besitzers. 6. Der User wählt den neuen Besitzer aus über ein Suchfeld aus. 7. Der User klickt auf „Speichern".
Resultat	Das System wechselt den Besitzer der Datei in der Datenbank. Wenn nötig werden die Zugriffsberechtigungen angepasst.
Ausnahmen	• Keine Ausnahmen, der Besitzer kann immer gewechselt werden

6) Anwendungsfall „Datei dem Warenkorb hinzufügen”	
Kurzbeschreibung	Der User fügt eine Datei seinem Warenkorb hinzu.
Akteure	User
Vorbedingungen	Der User hat einen Account und ist angemeldet.
Ablauf	*1.* Der User ruft die Webseite auf. *2.* Das System präsentiert die Startseite. Der User gibt im Suchfeld einen Dateinamen ein und klickt auf „Suchen“. *3.* Das System zeigt alle Dateien, die den Suchbegriff im Namen enthalten. *4.* Der User klickt auf „Dem Warenkorb hinzufügen“. *5.* Das System prüft, ob die Datei nicht bereits im Warenkorb existiert. *6.* Nach einer erfolgreichen Überprüfung wird die Datei dem Warenkorb hinzugefügt.
Resultat	Der Warenkorb wurde um die entsprechend ausgewählte Datei ergänzt.
Ausnahmen	• Im Warenkorb befinden sich bereits zehn Dateien (6)

7) Anwendungsfall "Warenkorb herunterladen”	
Kurzbeschreibung	Der User lädt seinen Warenkorb herunter.
Akteure	User
Vorbedingungen	Der User hat einen Account und ist angemeldet. Der Warenkorb ist nicht leer.
Ablauf	1. Der User ruft die Webseite auf. 2. Das System präsentiert die Startseite. 3. Der User klickt auf den Warenkorb. 4. Das System präsentiert die Warenkorbseite. 5. Der User klickt auf Warenkorb herunterladen. 6. Das System fügt alle Dateien im Warenkorb in einem Zip zusammen und lädt alle Dateien auf das lokale Gerät herunter. 7. Das System präsentiert die Rechnung
Resultat	Der Inhalt des Warenkorbs des Users wurde gezippt und beim User Lokal heruntergeladen.
Ausnahmen	• Der Applikationsnutzer verfügt über nicht mehr genug Speicherplatz auf seinem lokalen Gerät. (6)

5 Benutzerschnittstelle

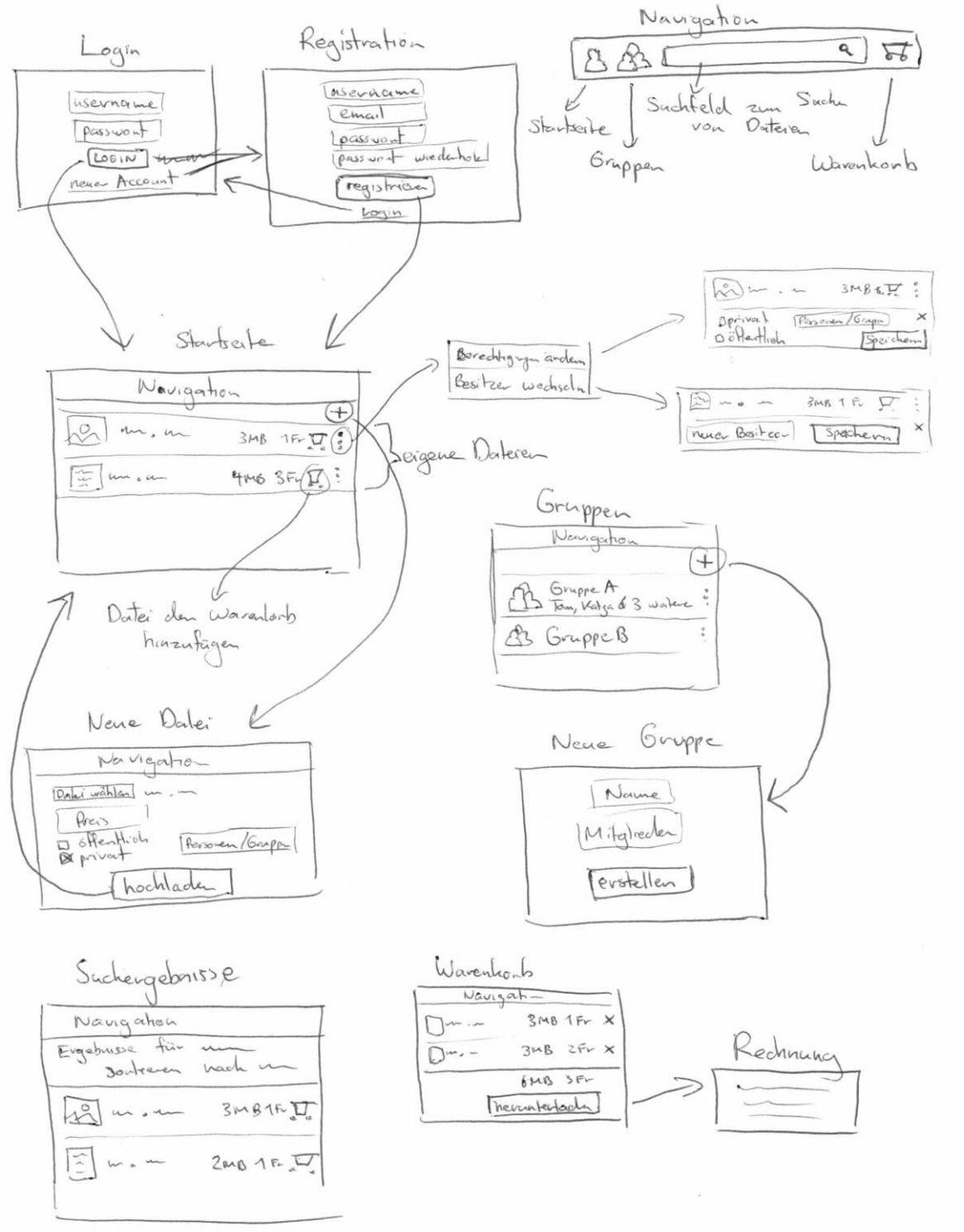

Abbildung 3: Skizze der Benutzerschnittstelle

6 Systemarchitektur

6.1 Gliederung der Lösung

Die Applikation wird als Server/Client Anwendung realisiert. Für die Server-und-Clientanwendung werden zwei unterschiedliche Projekte aufgesetzt, welche unabhängig voneinander auf verteilten Systemen lauffähig sind und betrieben werden können. Auf Serverseite wird das auf der Programmiersprache Python basierte Webframework Django verwendet. Für die Clientanwendung (grafische Benutzerschnittstelle) wird das AngularJS Framework eingesetzt. AngularJS verwendet TypeScript als Programmiersprache.
Die Applikation selbst wird in die in untenstehender Abbildung ersichtlichen Schichten aufgeteilt. Die Präsentationsschicht berücksichtigt für die Modellierung der Interaktion mit dem Benutzer die Aspekte des Model-View-Controller-Ansatzes. Die Applikationsschicht greift mithilfe von Controllerklassen der jeweiligen Entitäten auf die in der Datenbank vorhandenen Geschäftsdaten zu. Für die Speicherung der Objektzustände in der Persistenz Schicht wird eine relationale Datenbank verwendet, welche mithilfe des MySQL Managementsystems verwaltet wird.
Da die in der Applikationen benötigten Entitäten weiter oben im Entitätsdiagramm dargestellt wurden, werden die einzelnen Objektklassen (Models) nicht im Klassendiagramm dargestellt.

Abbildung zwei zeigt den Aufbau der gewählten Architektur in ihren jeweiligen einzelnen Schichten ohne auf die spezifischen Anwendungsfälle einzugehen.

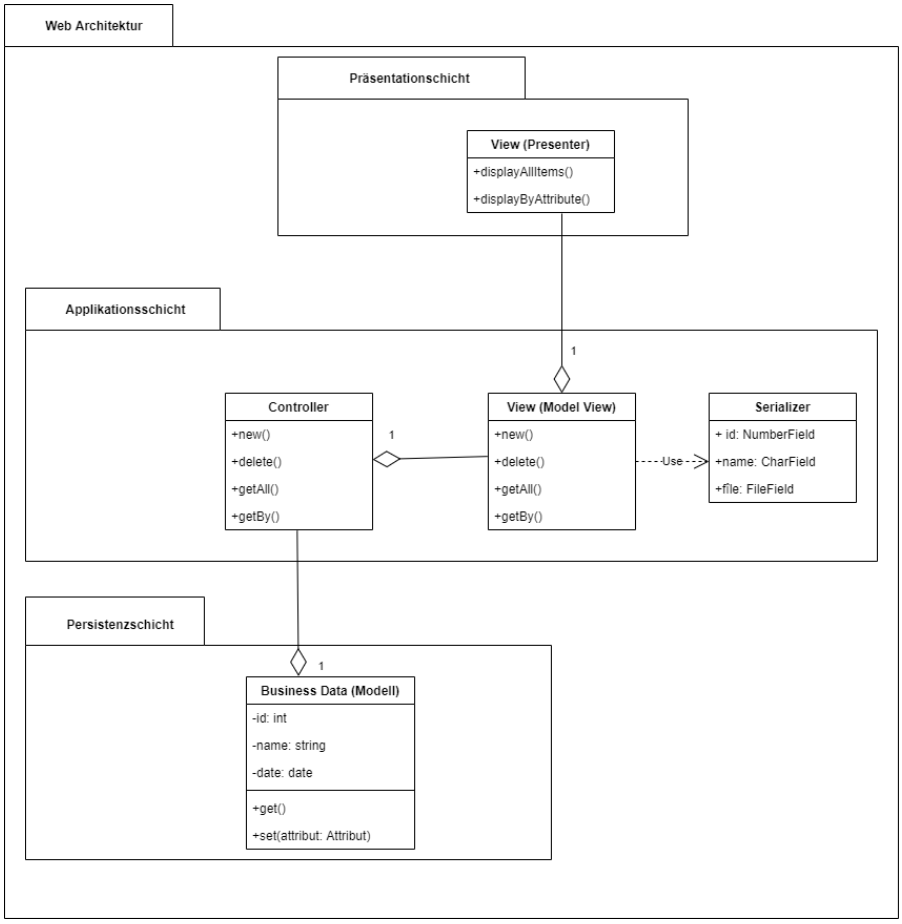

Abbildung 4: Schematische Darstellung der Gesamtarchitektur

Die Gesamtarchitektur gliedert sich in folgende Bausteine:

6.2 Schichten

6.2.1 Persistenz Schicht

Die Persistenz Schicht dient zur persistenten, also zu einer über einen längeren Zeitraum hinweg andauernde Speicherung von Daten in einer relationalen Datenbank. Der Grund, warum die Daten persistent gespeichert werden müssen, hängt damit zusammen, dass der Zustand der Datenobjekte nach dem Beenden und erneuten Aufstarten der Applikation wieder abrufbar sein muss. Mithilfe einer relationalen Datenbank können ebenfalls die oben beschriebenen Beziehungen gespeichert werden und somit die unter den jeweiligen durch Entitätsobjekte repräsentierten Akteuren existierenden Interaktionen implementiert werden.

6.2.2 Präsentationsschicht

Die sich am Model-View-Controller-Ansatz (MVC) orientierende Architektur in der Präsentationsschicht, ermöglicht es, Interaktionen mit dem Benutzer über die grafische Benutzerschnittstelle sauber von der Speicherung, Veränderung und der Darstellung der Objekte zu trennen. Das Modell in der Applikation legt die Struktur sowie der Zustand der gespeicherten Objekte, oder Entitäten, welche zu einem späteren Zeitpunkt dargestellt werden, fest. Das für die Entwicklung dieser Applikation eingesetzte Framework Django ermöglicht es, über das Modell direkt mit der Persistenz Schicht (Datenbank) zu kommunizieren und Daten in die Datenbank hineinzuschreiben oder auszulesen. Im Unterschied zum klassischen MVC-Ansatz sind die jeweiligen Views der Applikationsschicht nicht für die eigentliche Darstellung der Objekte auf dem Endgerät zuständig, sondern für dessen Aufbereitung. Diese Klassen werden oft auch Model Views genannt. Sobald der Server über eine Http-Anfrage kontaktiert wird, erhalten die Model Views über den entsprechenden Controller die vom Benutzer angeforderten Daten und überführen sie in eine für die Übertragung passende Form. Die Konvertierung, oft wird in diesem Zusammenhang auch von Serialisierung gesprochen, wird durch die in den zu den jeweiligen Views gehörenden Serialisierungsklassen durchgeführt. Die Deserialisierung überführt die Repräsentation der Daten wieder in den ursprünglichen Objektzustand, wie er vom Modell festgelegt wird. Die Deserialisierung erfolgt dann, wenn der Client Daten an den Server sendet und diese über den Controller in die Datenbank geschrieben werden sollen.
Nach der Übergabe der Daten an den Client durch die Model Views, können diese in der gewünschten Form dargestellt werden. Die tatsächliche Darstellung der Entitätsobjekte in Form von HTML-Quellcode erfolgt also durch eine von der Applikationsschicht unabhängiger höheren Schicht (vgl. Abbildung 3 oben).
Die Applikation wird so gebaut, dass jede Ressource, damit sind die durch das Modell repräsentierten Daten (Entitäten) gemeint, welche vom Benutzer angefordert werden können, eine eigene Model View-Klasse erhält. Wie oben bereits erwähnt, existiert zu jeder View eine passende Controller Klasse. Die Controller kommunizieren direkt über das Modell mit der Datenbank und können die gespeicherten Daten anfordern, löschen oder verändern sowie neue anlegen.

6.2.3 Geschäftslogik

Der Business Layer, oftmals ist damit die Geschäftslogik der Software gemeint, wird durch die bereits oben vorgestellten Entitäten repräsentiert, welche direkt Zugriff auf die Geschäftsdaten (Entitäten) haben. Auch die Use Cases (vgl. 2.3), stellen anwendungsspezifische Geschäftsregeln dar und beschreiben, zu welchem Sinn und Zweck des Systems existiert. Das unten abgebildete Klassendiagramm stellt in den Controllerklassen der jeweiligen Entitäten und Model Views die zentralen Use Case Objekte in Form von Funktionen dar, welche die anwendungsspezifischen Geschäftsregeln implementieren.
Die Geschäftslogik gliedert sich in folgende Teile:

Benutzerverwaltung

Die Benutzerverwaltung kontrolliert alle mit der Authentifizierung, (Sessions, Login) oder Registrierung korrelierenden Prozesse. Die Benutzerverwaltung regelt auch das Löschen und das Bearbeiten eines bereits existierenden Accounts.

Dateienverwaltung

Eine sehr bedeutende und zentrale Geschäftsregel ist die Dateiverwaltung durch den Benutzer. Das umschliesst das Hochladen und Entfernen einer Datei, sowie weitere Aspekte wie die Vergabe von Zugriffsberechtigungen auf eine Datei.

Gruppenverwaltung

Die Gruppenverwaltung umfasst das Erstellen, Bearbeiten und Löschen von Gruppen.

Berechtigungsverwaltung
Die Berechtigungsverwaltung bietet die Möglichkeit, Zugriffsberechtigungen auf Dateien an weitere Personen und Gruppen zu vergeben.

6.2.4 Pakete

Wie dem Klassendiagramm zu entnehmen ist, teilt sich die Applikation unter Berücksichtigung der Geschäftsregeln und die in direktem Zusammenhang stehenden Funktionalitäten (Use Cases) in folgende Pakete:

Invitation API

Umfasst die Funktionalität, Personen in Gruppen einzuladen und diesen somit die Berechtigung zu verleihen, Einsicht in die in der Gruppe enthaltenen Dateien zu gewähren.

Group API

Beinhaltet die Funktionalitäten der Gruppenverwaltung

File API

Regelt die Dateiverwaltung und die Zugriffsberechtigungsvergabe auf Dateien.

Person API

Alle Funktionalitäten zur Verwaltung von Benutzerdaten, werden in diesem Paket geregelt.

Share API

Enthält die Funktionalität für die Vergabe von Zugriffsberechtigungen für Personen und Gruppen auf Dateien.

Authentication API

Dieses Paket enthält die Regeln zur Authentifizierung von Benutzern, welche über ein Benutzerkonto verfügen.

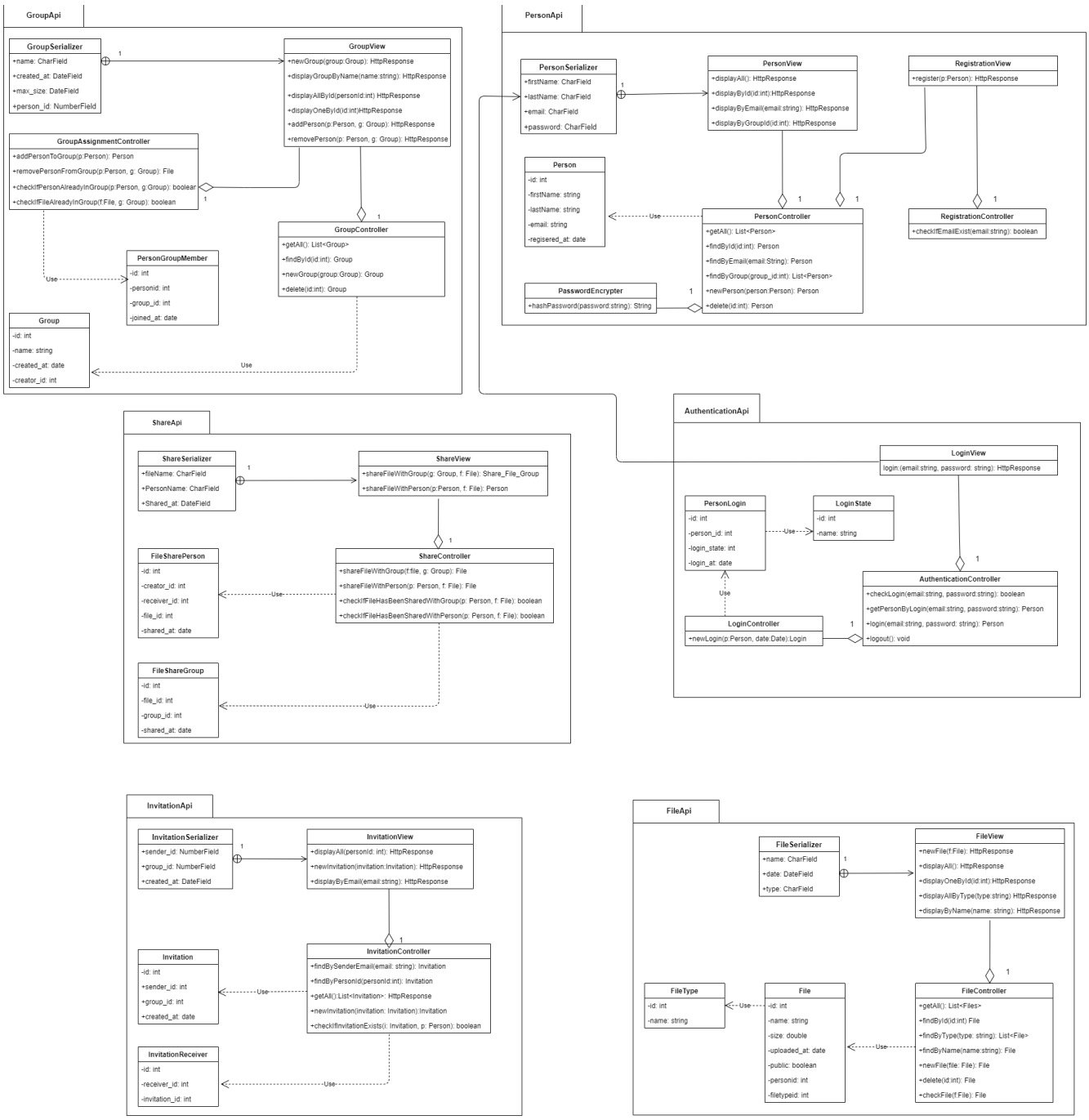

Abbildung 5: Klassendiagramm

7 Schnittstellen

7.1 REST Schnittstelle

Die Kommunikationsschnittstelle zwischen der Sever-und-Client-Anwendung erfolgt über eine sogenannte Rest API-Schnittstelle. Eine Rest API basiert auf dem HTTP-Protokoll und dient dem zustandslosen Datenaustausch zwischen verteilten Systemen.
Wird mit der Clientanwendung, bspw. durch einen Applikationsnutzer über die grafische Benutzerschnittstelle ein Event ausgelöst und Daten angefordert, so wird der Server über eine HTTP-Verbindung (bidirektional) mit einer URL (Uniform Ressource Language) kontaktiert. Das ist der sogenannte Http-Request (Anfrage). Über die URL identifiziert der Server, welche Operation auf dem Modell verlangt und welche Ressource(n) vom Client angefordert, verändert oder gelöscht werden sollen. Der Server konvertiert die Daten in eines für die Übertragung passendes Format, in diesem Fall ist dies das JSON-Format und sendet die Ressource zurück an den Client (Response).
Wie oben bereits erwähnt, erfolgt jede Anfrage eines Clients an den Server zustandslos. Der Server speichert also keine spezifischen Daten des Gerätes, welches Daten anfordert oder speichern möchte.
Somit müssen bei jeder Anfrage, um Ressourcen zu speichern, bestehende zu verändern oder existierende abzufangen, alle dafür nötigen Informationen neu gesendet werden.
Sendet die Clientanwendung in umgekehrter Reihenfolge Daten an den Server, beispielsweise ist das der Fall, wenn eine neue Ressource persistent gespeichert werden soll, so werden auch hier die Daten im JSON-Format übertragen und anschliessend auf dem Server in die vom Modell repräsentierte Datenstruktur konvertiert.
Die JavaScript Object Notation (JSON) ist ein von der Programmiersprache unabhängiges Datenaustauschformat.
Untenstehende Abbildung visualisiert den Kommunikationsprozess schematisch.

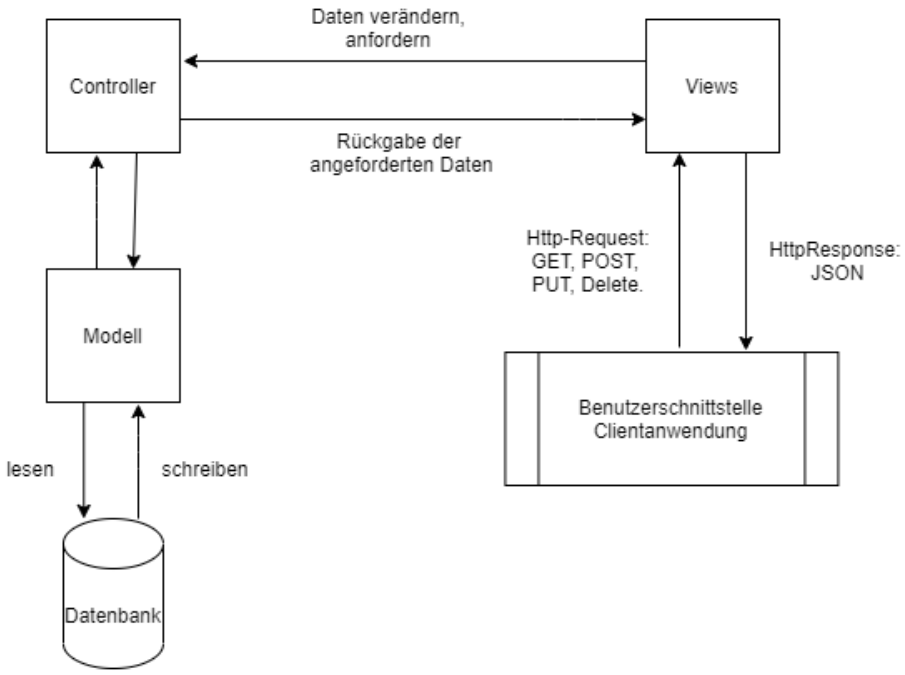

Abbildung 6: Kommunikation zwischen Client, Server und Datenbank

8 Qualitätssicherung

8.1 Testfälle der Use Cases

Die in folgender Tabelle aufgelisteten Testfälle decken die oben beschriebenen Anwendungsfälle, vgl. Anwendungsfälle 4.1, ab.
Teilweise beziehen sich die Testfälle nicht direkt auf die im oberen Teil beschriebenen Use Cases, sondern auf weitere, unter Berücksichtigung der aktuellen Anwendungsfälle sich abspalteten Use Cases. Teilweise werden Testfälle dargelegt, welche die Negation oder Restriktionen von Use Cases darstellen.

Nr.	***Abgedeckter Anwendungsfall oder User Stories***	***Beschreibung***
1	*Neues Benutzerkonto erstellen (Use Case 1)*	*Ein Testbenutzer, welcher noch kein Konto besitzt, registriert sich beim System mit einer Emailadresse und Passwort und kann sich anschliessend mit diesen Informationen anmelden.*
2	*Der Benutzer lädt eine Datei öffentlich hoch (Use Case 2)*	*Da die Datei als öffentlich gekennzeichnet und geteilt wurde, kann sie nun von jeder Person konsultiert werden und befindet sich somit in der öffentlichen Dateiablage.*
3	*Der Benutzer lädt eine private Datei hoch (Use Case 2)*	*Da die Datei als privates Element hochgeladen wurde, befindet sie sich nicht in der öffentlichen Dateiablage der Webseite. Nur der Besitzer kann die hochgeladene Datei in seiner privaten Dateiablage einsehen.*
4	*Vergabe von Zugriffsberechtigungen auf eine Datei an eine einzelne Person (Use Case 4)*	*Wird eine Datei durch einen Applikationsnutzer einer anderen Person zugewiesen, so generiert das System eine Meldung, welche den Benutzer informiert, dass er seine Dateizuweisung erhalten hat. Sobald die Zuweisung erfolgreich war, kann die entsprechende Person die Datei einsehen oder herunterladen.*
5	*Eine Gruppe wird für den Zugriff auf eine Datei berechtigt. (Use Case 4)*	*Alle in der Gruppe enthaltenen Mitglieder haben nun Einsicht in die Datei und können sie auf ihr lokales Gerät herunterladen.*
6	*Eine Gruppe kann nicht zweimal auf die gleiche Datei berechtigt werden (Use Case 4)*	*Wird ein Versuch unternommen, eine bereits in der Gruppe existierende Datei erneut hinzuzufügen, so wird dies blockiert und das System generiert eine Meldung, welche den Benutzer informiert.*
7	*Änderungen des Status einer Datei von öffentlich nach privat (Use Case 4)*	*Die von der Änderung der Zugriffsberechtigungen betroffene Datei kann in der Öffentlichen Dateiablage nicht mehr aufgesucht werden. Der Besitzer der Datei ist der einzige Akteur, wel-*

		cher die Datei in seiner privaten Dateiablage konsultieren und gegebenenfalls herunterladen kann.
8	*Besitzer einer Datei wechseln (Use Case 5)*	*Wird der Besitzer einer Datei geändert, so hat der neue Besitzer nun die Möglichkeit, Zugriffsberechtigungen auf die Datei an Gruppen oder Einzelpersonen zu vergeben*
9	*Hinzufügen einer Datei in den Warenkorb (Use Case 6)*	*Bei einem erfolgreichen Hinzufügen informiert das System der aktuelle Benutzer passend auf die erfolgreiche Speicherung der Datei im Warenkorb hin. Der Warenkorb wurde um die entsprechende Datei ergänzt.*
10	*Blockierung der Warenkorbergänzung um eine Datei (Use Case 6)*	*Befinden sich im Warenkorb bereits zehn Dateien, so wird das Abspeichern einer neuen Datei im Warenkorb vom System verhindert.*
11	*Datei kann nicht doppelt im Warenkorb vorhanden sein. (Use Case 6)*	*Wird dem Warenkorb die gleiche Datei erneut hinzugefügt, so generiert das System eine Meldung und blockiert das Speichern der Datei im Warenkorb*
12	*Dateien im Warenkorb auf das lokale Gerät herunterladen (Use Case 7)*	*Nachdem der Benutzer seine im Warenkorb enthaltenen Dateien heruntergeladen hat, müssen sich diese anschliessend im Ordner „Downloads“ des lokalen Gerätes befinden.*
13	*Entfernen der Dateien im Warenkorb (Use Case 7)*	*Alle im Warenkorb enthaltenen Dateien werden aus diesem entfernt, sobald der Benutzer die entsprechenden Dokumente auf sein lokales Gerät heruntergeladen hat.*

9 Projektplanung

Der aktuelle Zeitplan, welcher wir im Projektinitialisierungsantrag erstellt haben, ist bisher gut aufgegangen. Die Phasen waren richtig definiert und die Zeit wurde ordnungsgemäss eingehalten. In den weiteren Phasen verfolgen wir wie bisher unseren Zeitplan. Ab der Woche 43 werden wir mit der Realisierung des Projektes beginnen.

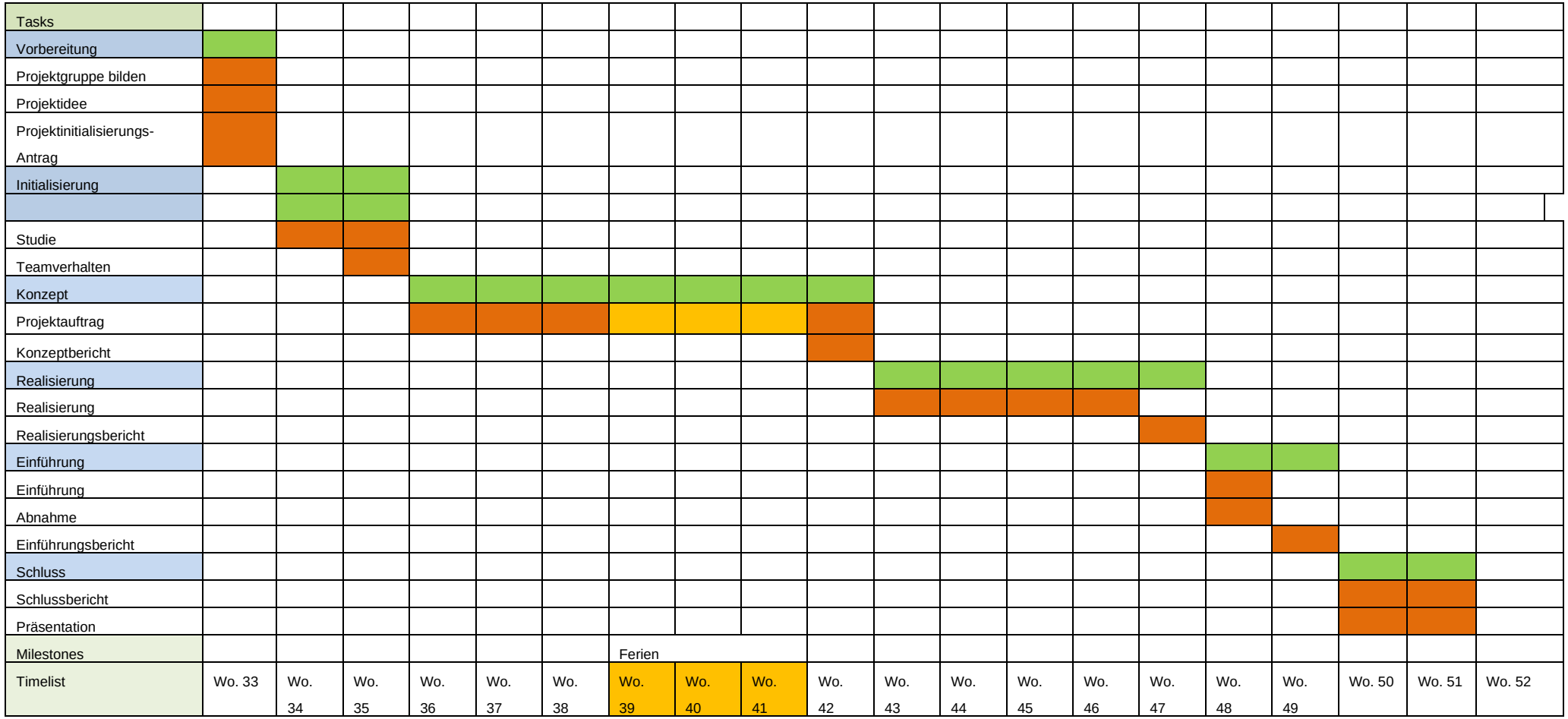

Abbildung 7: Gantdiagramm Projektplanung

Weiterhin gelten die während der Initialisierungsphase analysierten Risiken.

10 Abbildungsverzeichnis