Abschlussprojekt ProPra 2

Daniel Warkus, Jacques Pasquier, Robin Kraft, Jeff Nosbusch, Franz Hoffmann,
Wolfgang Hinzmann, Oliver Nern, David Huseyin

Version 0.1, 11.02.2019

Table of Contents

Festlegen von Syntaxregeln

Zu Beginn des Praktikums haben wir uns auf eine einheitliche Syntax geeinigt. Im allgemeinen folgen wir dabei den bisher gelernten Coderichtlinien, die wir hier nochmal zusammenfassen:

- Namen und Kommentare in Englisch
- Tabs anstelle von 4 Leerzeichen
- { hinter die Methode und nicht in eine neue Zeile
- Variablennamen: fooBar
- Methodennamen: fooBar
- Konstanten: CON_VAR

Zusätzlich wird zu erst eine funktionierende Datei in den `master`-Branch gepushed, von dem alle zum arbeiten eigene Branches erstellen. Gemerged wird erst, sobald das bearbeitete Problem volständig funktioniert, getestet und dokumentiert wurde.

Planung

Unser Plan sieht es vor, 3 Self-contained-systems zu erstellen, die sich um die Verwaltung der Daten von Benutzern, Artikeln und Ausleihen kümmern und jeweils ihr eigenes Mapping ausführen. Im Hintergrund soll dazu eine `Postgres`-Datenbank laufen, auf die alle drei Systeme Zugriff haben und die die Daten speichert.

Realität

Unsere erste Planung ließen wir am Ende von Woche 2 fallen und stellten auf eine MVC-Architektur um, die die Kommunikation zwischen den Klassen erleichtert. Zudem haben wir auf eine `mariaDB` -Datenbank gewechselt.

Dokumentation

Artikel

Wir haben uns vorgestellt, dass die Anwendung `Article` die Artikel als Entitäten in die Datenbank einfügt. Ein einzelner Artikel enthält die Variablen `articleID`, `name`, `comment`, `PersonID`, `deposit`, `rent`, `available`, um die geforderten Daten speichern zu können. Dies wurde mit den entsprechenden Annotationen implementiert. Als nächstes wurde ein entsprechendes Repository `ArticleRepository` erstellt, dass das CrudRepository extended, um die Funktionen ebendieses für das Mapping anwenden zu können. Danach folgte ein Controller `ArticleController`, der das Mapping für die einzelnen Artikel übernimmt. Der Controller kann verschiede Ansichten erzeugen, unter Anderem eine Übersicht über alle hochgeladenen Artikel und eine Detailansicht ebendieser.

Zudem können Artikel bearbeitet werden. Dabei können lediglich die Attribute `name`, `comment`,

`deposit`, `rent` und `available` geändert werden, der Rest wird vom alten Artikel automatisch übernommen. Als erweiterte Features wurden dann noch ein paar Grundlagen implementiert, unter anderem das Editieren von Artikeln sowie das Löschen.

Desweiteren haben wir uns einem anspruchsvolleren Feature gewidmet: dem hinzufügen von Bildern. Diese werden beim Erstellen dem Artikel hochgeladen und können weder bearbeitet, noch gelöscht werden. Die Bilder sieht man dann in der Artikelübersicht. Um das zu vollbringen haben wir das Bild erstmal als MultipartFile dem Artikel nicht Persistent hinzugefügt, dann die wichtigen Informationen aus dem `File` ausgelesen und in einer `Image` Klasse dafür gespeichert. Dem Artikel weisen wir dann in einer `OnetoOne` Beziehung ein Image zu. In der Artikelansicht wird dann ein spezieller Imagecontroller aufgerufen, der die Bytes des Bildes zurückgibt.

Als nächstes implementierten wir eine Suchfunktion, die einen `Query` entgegennimmt und nach dem Vorkommen des Queries in den Namen und Beschreibungen der Artikel sucht. Alle auf diese Weise gefundenen Artikel werden dann auf einer Übersichtsseite ausgegeben. Die von uns implementierte Suche ist zudem Case-insensitive.

Die Artikel sind so erstellt, dass der User, dem der Artikel gehört, die Möglichkeit hat, den Artikel zu löschen oder zu bearbeiten. Durch die Änderungen der Aufgabe hat er da auch die Möglichkeit, den Kaufpreis zu setzen und die Kauffunktion zu aktivieren (Was er optional auch beim Einstellen des Artikels machen kann). Ein User, dem ein Artikel nicht gehört, kann dieses statt dessen Ausleihen (sofern er verfügbar ist) oder Kaufen, wenn dies freigeschaltet ist.

In der Artikelübersicht sieht man das Artikelbild oder ein Standardbild, sowie einen Preis und einen Standort (Stadt). Durch einen Klick auf den Artikel kommt man auf die oben erwähnte Detailansicht des Artikels.

User

Man kann sich einloggen, indem man seinen Username in das dafür vorgesehen Textfeld schreibt und den `Einloggen` Button betätigt. Die einzige Vorraussetzung dafür ist, dass bereits ein User mit diesem Usernamen und Passwort in der Datenbank vorhanden ist. Ansonsten kann man den `Registrieren` Button betätigen, dann wird man zur Registrieren-Seite weitergeleitet. Hier kann man nun einen ganz neuen User erstellen, indem man alle Textfelder mit korrekten Eingaben ausfüllt und den entsprechenden Button betätigt. Nach dem Einloggen bzw. Registrieren, kann man den `Profil Bearbeiten` Button betätigen, um auf eine Seite weitergeleitet zu werden, auf der man seine Profilinformationen einsehen und bearbeiten kann. Mit einem Klick auf `Aktualisieren` werden die Änderungen in die Datenbank aufgenommen.

Testing (User)

Wir haben viele Tests geschrieben und haben eine Overall line Coverage von über 85% erreicht. Es werden alle Methoden in den Controllern getestet. Wir erreichen keine 100% Coverage, weil wir das Repository gemockt haben und in der Methode ins Repository gespeichert wird und in der gleichen Methode wieder ausgelesen wenn wir die Methode für diesen Username mocken wird er schon in der vorherigen Methode abgefangen.

Ausleihe

Eine Ausleihe ist eine eigene Entität namens `Lending`, diese beinhaltet die `lendingId`, die `lendingPerson` für die ausleihende Person und die `lendedArticle`, um die Artikel in der Datenbank zu finden, weil diese die `id` von der `ownerPerson` besitzt, die den Artikel anbietet. Anfangs bekommt man eine Anfrage von der Artikelübersicht, und wird auf `/lendingRequest` weitergeleitet. Hier kann ebenfalls auf `/saleRequest` weitergeleitet werden. Bei beiden wird als Pathvariable die `id` von dem Artikel mitgeben. Bei einem Verkauf wird nur nach einem Kommentar gefragt und dieser wird dem Besitzer des Artikels auf der `overview` Seite angezeigt. Wenn der Request abgeschickt wird, wird man auf die `inquiry` Seite weitergeleitet, wo man informiert wird, dass die Anfrage bearbeitet wird.

Für die Ausleihe muss man das `startDate` und das `endDate` eingeben wie lange die Ausleihe gehen soll. Ebenfalls kann man einen `requestComment` eingeben um sich für den Artikel zu bewerben. Bei beiden Fällen wird eine neue Entität `Lending` angelegt und beim akzeptieren wird die Reservation im `Lending` gespeichert. diese werden als `isRequest` markiert und dem Besitzer in der `overview` zum entscheiden angezeigt. In beiden Fällen wird der Artikel auf `isAvailable == false` gesetzt. Falls der Besitzer sich dagegen entscheidet, den Artikel zu verkaufen, wird ein boolean `isDummy` auf `true` gesetzt, damit der kaufenden oder leihenden Person auf der `overview` eine Information angezeigt wird, dass diese Anfrage abgelehnt wurde. Wenn die anfragende Person auf den `Bestätigen` Button klickt, wird das `Lending` aus der Datenbank gelöscht. Wenn die Verkaufanfrage bestätigt wird, wird eine `Transaction` zwischen dem Käufer und dem Bestizer erzeugt und in der Datenbank gespeichert. Dabei wird der Artikel rausgelöscht und ein DummyArtikel für die Transaktionen angelegt. Wird ein Leihanfrage von dem Besitzer angenommen in der `overview` Seite, dann wird der Leihenden Person die Ausleihe in `OverviewLending` angezeigt.

Dort kann Sie einen Konflikt erzeugen oder den Artikel zurückgeben. Wenn der Artikel zurückgegeben wird, wird ein Flag `isReturn` auf `true` gesetzt und der Besitzer von dem Artikel wird auf der `overview` Seite eine Anfrage angezeigt ob er den Artikel zurücknehmen will oder den Zustand des Artikels nicht für gut genug empfindet. Beim akzeptieren des Artikels, wird der Betrag ausgerechnet und in eine `Transaction` gespeichert und die Reservation die anfangs angelegt wurde wird gelöscht und danach wird diese auch über Propay aufgelöst. Beim ablehnen wird ein Konflikt erzeugt und diese kann von den Admins erledigt werden. Im `OverviewLending` wird auch ausgerechnet wie lange man den Artikel noch ausleihen darf oder zeigt einen Warning Text an wenn der Artikel zu lange ausgeliehen ist. In der `overview` wird dann ein pop-up geöffnet wenn ein Artikel zu lange ausgeliehen ist. Auf der Seite `PropayOverview` Wird das Guthaben des Accounts angezeigt mit allen `Reservations` und dort ist ein Feld zum aufladen des Kontos.

Falls Propay nicht geöffnet sein sollte, wird man auf eine Fehlerseite geleitet, die sagt das "Propay derzeit nicht erreichbar ist". Unter `conflictPage` Kann man sich alle Ausleihen angucken an denen man beteiligt ist und diese derzeit einen Konflikt hat. Falls man zu wenig Geld hat, während man sich etwas ausleiht oder bezahlt, wird man auf die `povertyPage` weitergeleitet. Zum befüllen der Html Seiten benutzen wir `Representation`-Klassen, die Listen von `lendings`, `requests` oder allen anderen benötigten Listen. In der Klasse `ApiProccessor` haben wir alle Methoden die mit Propay kommunizieren oder kontrollieren ob genügend Geld vorhanden ist auf den jeweiligen Propay-konten. In der `PostProccessor` werden alle Entscheidungen richtig vernetzt und alle Entitäten werden dort rausgelöscht und angelegt.

Änderungen vom Marketing

Durch die Änderungen durch die Marketingabteilung war nicht allzuviel zutun. Wir haben dem Artikel zwei Attribute hinzugefügt. Einen Boolean, der angibt ob das Produkt zum Verkauf ist, sowie aber auch ein Kaufpreis. Dieser ist Standartmäßig auf 0, was aber nicht relevant ist, da der Preis auch nicht angezeigt wird, wenn der Artikel als Nicht-verkaufbar eingestellt wird. Des weiteren haben wir die Edit Funktion der Artikel so abgeändert, dass man jetzt im Nachhinein den Verkaufshaken noch einsetzen kann, sowie auch den Preis ändern darf. Dazu kam dann ein Button für verkaufsbereite Artikel, der ein extra Lending Mapping aufruft für den Verkauf. Das Mapping läuft Analog zum Ausleihprozess. Die einzige Änderung ist, dass wir das Artikel dann erst kopieren mit den wichtigen Attributen, dann wird das original aus der Datenbank gelöscht. Die Kope ohne persönliche Daten behalten wir als Inaktiv für unsere `Lending/Verkaufshistorie`.

Chat

Der Chat ist erreichbar über `/messages`. Dies eine Seite mit dem Nachrichtenverlauf aller Nachrichten. Diese Übersicht ist für jeden Nutzer anders, da der Chat ein Eins-zu-Eins Chat ist. Um Nachrichten zu verschicken geht man auf einen Artikel und klickt dann auf die "Nachricht senden" Schaltfläche. Der Emfänger und der Sender werden automatisch festgelegt sodass der Sender nur den Inhalt der Nachricht schreiben muss. Die Nachrichten, die angezeigt werden, werden durch eine Filtermethode gefiltert (FROM, TO) die durch die Nachrichtendatenbank streamt. Auf der Übersichtsseite hat der Nutzer die Möglichkeit, auf Nachrichten zu antworten oder diese zu löschen. Beide Möglichkeiten leiten auf je eine eigene Webseite (`/messages/delete/{ID}`, `/messages/answer/{ID}`), welche dann die betreffende Nachricht findet die beantwortet oder gelöscht werden soll. Falls eine Nachricht gelöscht wird, wird die Nachricht für beide Benutzer gelöscht, da die Nachrichten sich eine Datenbank teilen. Beim beantworten der Nachricht wird automatisch der Sender und Empfänger festgelegt, sodass der Benutzer nur noch den Inhalt der Nachricht schreiben muss.

Konflikt

Es gibt zwei Möglichkeiten einen Konflikt zu erzeugen, der Benutzter welcher sich ein Gerät ausleiht kann während der Leih-Periode einen Konflikt erstellen und der Verleiher kann bei der Rückgabe falls der rückgegebenen Artikel beschädigt ist ebenfalls einen Konflikt erstellen. Wenn der Leiher zurückgeben will bekommt der Verleiher sofort nach erfolgreichem Login die Möglichkeit den Artikel anzunehmen oder einen Konflikt zu erstellen. Bei dem Erstellen eines Konfliktes wird eine Email mit der Nachricht an alle RhinoShare-Admins gesendet welche dann über den Konflikt entscheiden können. Entscheidet sich der Admin der bei einem Fall für den Verleiher so kann dieser die Kaution behalten, enscheidet sich der Admin für den Leiher so erhält dieser seine Kaution zurück.

Tests wurden geschrieben für die Filtermethode die diverse Beziehungen von den Chatnachrichten testet (zb: onetoone, onetotwo, etc..).

Administration

Die Verwaltung von Benutzern, Artikeln und Konflikten ist nur den Benutzern mit der Rolle **ROLE_ADMIN** möglich. Hierzu muss sich einfach mit einem Admin-Account eingeloggt werden. Alternativ kann der Bereich über den Tab mit dem Benutzernamen im Dropdown unter **Administration** erreicht werden.

Die Administration ist in verschiedene Sektionen unterteilt

Konfliktlösung

In der Konfliktlösung findet man als Administrator alle Konflikte zwischen Benutzern. Es kann entschieden werden, wer die Kaution im Endeffekt bekommt. Vorher sollte eine Konversation mit den Beteiligten stattfinden.

Benutzerverwaltung

DIe Benutzerverwaltung ermöglicht es, alle Benutzerprofile zu editieren und neue zu erstellen. Die Bearbeitung erfolgt über den Button **Bearbeiten**. Der neue Benutzer lässt sich über **Neuer Benutzer** anlegen. Der Button **Löschen** löscht den jeweiligen Account.

Artikel und Ausleihen

Die Beiden Tabs **Artikel** und **Ausleihen** verschaffen dem Administrator einen Überblick über die jeweilige Kategorie. Einzelne Elemente können mit einem Klick auf **Löschen** gelöscht werden.

IMPORTANT	Vorsicht! Artikel lassen sich nur löschen, wenn keine Ausleihe darauf existiert. Benutzer lassen sich nur löschen, wenn keine Artikel auf dessen Namen gespeichert sind.