Dokumentation
„UNI-Buch“

Ein Projekt im Rahmen der Veranstaltung
Webbasierte Anwendungen 2 - Verteilte Systeme

Gruppe 02: Fabian Fuchs, Clara Meschkat, Eugen Scherer
Betreuer: Prof. Dr. Kristian Fischer, Sheree Saßmannshausen, Sascha Lemke

Aufgabenstellung und Ziel

Der Workshop soll die zentralen Konzepte der Datenmodellierung, synchrone sowie möglichst asynchrone Interaktion durch praktische Umsetzung mit aktuellen Werkzeugen auf nodeJS - Basis vertiefen, die unten stehender Referenzarchitektur entspricht. Die Abgabe des Codes erfolgt mittels GitHub, sodass die Commit - Operationen der einzelnen Teammitglieder zu den jeweiligen Diensten nachvollziehbar sind.

Die Aufgaben sind folgende:

1. Vom Web Service bereitgestellte Dienste in Form einer Tabelle spezifizieren, die zur Implementation des Dienstanbieters benötigt werden.
2. Fertigstellung des Web Services mit den zuvor spezifizierten Ressourcen der REST-Architektur
3. Drei Use-Cases für den Dienstnutzer des Projektes definieren
4. Realisierung des Dienstnutzers

Problemszenario

Hans-Peter braucht für seinen Betriebssystemkurs ein Fachbuch, welches ihm im Laden zu teuer ist. Währenddessen mistet Dirk sein Zimmer aus und findet einige Bücher aus seinem Studium, die in seiner Wohnung zu viel Platz wegnehmen. Dirk hat allerdings keine Lust seine Bücher zu verschicken, sondern möchte sie gerne lokal an der Hochschule loswerden, da er sich dort täglich aufhält.

Lösungsansatz

Dirk kann nun seine alten Fachbücher über UNIBuch bei seiner Hochschule einstellen, und andere Studierende können sehen welche Bücher dort momentan angeboten werden und mit dem Anbieter Kontakt aufnehmen.

Projektdefinition

Die Anwendung „UNIBuch“ ermöglicht es Studenten lokal an ihrem jeweiligen Campus ihre gebrauchten Fachbücher an andere Studenten zu verkaufen. Der Benutzer muss sich mit seiner E-Mail Adresse auf der UNIBuch Seite registrieren um Angebote einzustellen. Er kann über die Suchfunktion nach bestimmten Büchern suchen oder seiner Wunschliste hinzufügen. Auf der Startseite befindet sich außerdem eine Auswahl der beliebtesten Bücher. Wenn der Benutzer ein passendes Angebot gefunden hat kann er per Email Kontakt mit den Verkäufer aufnehmen um genauere Details des Verkaufs zu klären.

Dienstnutzer

Die Anwendung soll Angaben wie Fakultät, Buchname, Semester, Modulnamen speichern, verwalten und filtern können, so dass nur relevante Informationen zu der Buchsuche angezeigt werden. Des Weiteren sollen Informationen wie die Buchbeschreibung, Cover und ISBN von der Google Books API zur Verfügung gestellt werden.

Dienstanbieter

Die Angebote werden in einer Datenbank gespeichert. Außerdem verwaltet der Dienstanbieter die Informationen der Nutzer.

Anforderungen

- der User muss Artikel anlegen können
- der User muss seine Artikel verwalten können
- der User muss bestimmte Artikel aufrufen können
- der User muss seine Suchergebnisse filtern können
- der User muss Artikel mit einer Volltext-Suche finden können
- der User muss Artikel mit Hilfe von vordefinierten Kategorien finden können
- der User muss seine Kontaktdaten für Interessenten hinterlegen können

Use Cases

Use-Case 1: Buch einstellen

Der User kann ein Buch auf der Plattform einstellen. Dazu muss er mindestens den Buchtitel angeben, weitere Buchinformationen werden von der Google Books API bereitgestellt. Optional kann noch eine Artikelbeschreibung, der Zustand, und/oder der gewünschten Preis angegeben werden.

Precondition	Der User muss sich Registriert haben
Success End Condition	Der User hat erfolgreich ein Buch auf der Plattform eingestellt
Failed End Condition	Der User konnte das Buch nicht auf der Plattform einstellen
Primary Actor	User
Trigger	Der User hat ungenutzte Bücher Zuhause

Beschreibung:

1. Der User wählt die Funktion "neuen Artikel einstellen"
2. Die Seite zur Erstellung eines Artikels öffnet sich
3. Der User füllt die erforderlichen Felder aus und bestätigt seine Eingabe
4. Der Artikel wird mit der Datenbank der Google Books API abgeglichen und die restlichen Informationen wie Cover, Zusammenfassung etc. werden ergänzt
5. System speichert den Artikel

Use-Case 2 : Buchsuche

Der User kann ein Buch suchen. Zur Suche kann er den Titel oder die ISBN Nummer des gesuchten Buches verwenden. Des weiteren kann nach Kategorien gefiltert werden.

Precondition	Der User muss sich Registriert haben
Success End Condition	Der User hat erfolgreich ein Buch gefunden

Failed End Condition	Der User konnte das Buch nicht finden
Primary Actor	User
Trigger	Der User braucht ein bestimmtes Buch

Beschreibung:
1. Der User wählt die Funktion "Büchersuche"
2. Die Seite zur Büchersuche mit Such-Eingabefeld öffnet sich
3. Der User füllt das erforderlichen Felder aus und bestätigt seine Eingabe
4. Der User schränkt seine Suche mit Hilfe von Filterfunktionen ein
5. Die Seite mit den gefundenen Büchern wird in Listenform ausgegeben
6. Der User wählt das gesuchte Angebot

Use-Case 3 : Wishlist

Der User kann eine Wishlist anlegen und dort favorisierte Angebote von Büchern speichern, oder auch wieder entfernen.

Precondition	Der User muss sich Registriert haben
Success End Condition	Der User hat erfolgreich ein Buch zu seiner Wishlist hinzugefügt oder entfernt
Failed End Condition	Der User konnte das Buch nicht zu seiner Wishlist hizufügen oder es entfernen
Primary Actor	User
Trigger	Der User möchte ein bestimmtes Angebot für einen späteren Moment speichern

Beschreibung - Hinzufügen:
1. Der User navigiert zu einer Artikelseite
2. Der User nutzt auf das "Herz-Symbol" um den Artikel zu seiner Wishlist hinzuzufügen

Beschreibung - Entfernen:

1. Der User navigiert zu seinem Profil
2. Der User nutzt das Wishlist-Icon um auf die die Wishlist-Seite weitergeleitet zu werden
3. Der User wählt das Angebot, dass er von seiner Wishlist entfernen möchte
4. Der user nutzt die Funktion "von der Wishlist entfernen" um den Artikel von der Wishlist zu entfernen

Use-Case 4 : Registrieren

Ein User kann sich auf der Plattform registrieren.

Precondition	Es muss eine Datenbank existieren um Benutzer zu speichern
Success End Condition	Der User hat sich erfolgreich registriert
Failed End Condition	Der User konnte sich nicht erfolgreich registrieren
Primary Actor	User , Datenbank
Trigger	Der User braucht ein bestimmtes Buch und möchte sich auf der Plattform umsehen

Beschreibung:

1. Der User nutzt die Registrations-funktion
2. Der User füllt das Registrations-formular aus
3. Der User bestätigt seine Eingabe
4. Das System speichert die Userdaten in der Datenbank
5. Das System gibt dem User eine Registrationsbestätigung zurück

Proof of Concept

Verwendung der Google Books API zur Realisierung des Projekts

Was Ist Google Books?
"Google Bücher" oder "Google Books" funktioniert wie die Web- und Bildersuche von Google. Hier durchsucht man allerdings nicht das Web, sondern digitalisierte Bücher. Die Auswahl reicht von Belletristik und Sachbüchern bis hin zu wissenschaftlichen Werken sowie Lehr- und Fachliteratur.

Verwendung innerhalb des Projekts

Google Books liefert uns alle notwendigen Informationen zu dem Buch. Wir bekommen die Bibliografischen Informationen, wie Titel, Autor, Verlag, ISBN und Seitenzahl. Zudem kann auch das Buchcover und die Kurzbeschreibung des Buches eingebunden werden. Um einen besseren Überblick über das Buch zu erhalten, können auch Rezension von anderen Nutzern eingeblendet werden.
Google Books stellt auch Ausschnitte aus den Büchern zur Verfügung, dadurch kann sich der Nutzer kurz in das Buch einlesen und beurteilen ob Ihm das Angebotene weiterhilft und er dieses Kaufen möchte.
Durch die Verwendung der Google Books API ist es für die Nutzer einfacher ein Buch einzustellen. Der Nutzer benötigt nur die ISBN des Buches. Möchte der Nutzer ein Buch einstellen wird er nach der ISBN und dem Verkaufspreis gefragt. Nach Eingabe dieser Daten wird automatisch ein neues Angebot mit allen Informationen zu dem Buch erstell.

REST Spezifikation

Definition der REST Ressourcen die für den Dienstnutzer benötigt werden

Primärressourcen unserer Anwendung sind *books* und *user*. Diese bilden gleichzeitig auch Listenressourcen, über *GET /books* oder *GET /users* wird die Repräsentation alle Bücher/User übermittelt.
Mit einem *POST (POST /books, POST /users)* auf die Listenressource wird ein neuer Eintrag in der Liste der Primärressource angelegt.
Soll eine Entität einer Primärressource verändert werden, z.B das die Artikelbeschreibung eines Buches geändert werden soll, geschieht dies über *PUT /book/:isbn* wobei der *isbn* Parameter der URI der *isbn* des Buches entspricht. Das gleiche Prinzip trifft auch auf den *user* zu.
Mit einem *DELETE* auf eine spezielle Primärressource wird diese entfernt. Sie ist dann nicht mehr über ihre URI *books/:isbn* aufrufbar.
Mithilfe von Queries können wir Bücher suchen und gegebenfalls nach Kategorien filtern.

Ressource	Methode	Semantik	content-type(req)	content-type(res)
/users	GET	Listet alle Benutzer auf	text/plain	application/ json
/users	POST	Anlegen eines neuen Benutzers	application/ json	application/ json
/books	GET	Listet alle Bücher auf	text/plain	application/ json
/books	POST	Anlegen eines neuen Buchs	application/ json	application/ json
/books/?query=search+term	GET	Durchsuche die Listen-ressource nach dem bestimmten Buch	application/ json	application/ json

Ressource	Methode	Semantik	content-type(req)	content-type(res)
/books/?category=filter	GET	Filtert die Listenressource nach Büchern gewünschter Kategorie	application/json	application/json
/user/{ID}	GET	Zurückgeben von Information über Benutzer	application/json	application/json
/user/{ID}	PUT	Aktualisiere Benutzer	application/json	application/json
/user/{ID}	DELETE	Löscht einen bestimmten Benutzer	application/json	application/json
/user/{ID}/book	POST	Anlegen eines neuen Buches	application/json	application/json
/user/{ID}/book/{ID}	GET	Bestimmtes Buch abrufen	application/json	application/json
/user/{ID}/book/{ID}	PUT	Buch aktualisieren	application/json	application/json
/user/{ID}/book/{ID}	DELETE	Lösche ein bestimmtes Buch	application/json	application/json
/user/wishlist/book	POST	Buch auf Wishlist setzen	application/json	application/json
/user/wishlist	GET	Wishlist Abrufen	application/json	application/json
/user/wishlist/book/{ID}	DELETE	Bestimmtes Buch von der Wishlist nehmen	application/json	application/json
/category	GET	Kategorien Abrufen	application/json	application/json

HTTP Status Codes

Diese Codes können von uns verwendet werden

Code	Nachricht	Bedeutung
200	OK	Die Anfrage wurde erfolgreich bearbeitet und das Ergebnis der Anfrage wird in der Antwort übertragen
204	no Content	Die Anfrage wurde erfolgreich durchgeführt, die Antwort enthält jedoch bewusst keine Daten
205	reset Content	Die Anfrage wurde erfolgreich durchgeführt; der Client soll das Dokument neu aufbauen und Formulareingaben zurücksetzen
207	multi-Status	Die Antwort enthält ein XML-Dokument, das mehrere Statuscodes zu unabhängig voneinander durchgeführten Operationen enthält
304	not modified	Der Inhalt der angeforderten Ressource hat sich seit der letzten Abfrage des Clients nicht verändert und wird deshalb nicht übertragen
305	use proxy	Die angeforderte Ressource ist nur über einen Proxy erreichbar
400	bad request	Die angeforderte Ressource ist nur über einen Proxy erreichbar
401	unautorized	Die Anfrage kann nicht ohne gültige Authentifizierung durchgeführt werden
403	forbidden	Die Anfrage wurde mangels Berechtigung des Clients nicht durchgeführt

Code	Nachricht	Bedeutung
404	not found	Die angeforderte Ressource wurde nicht gefunden
416	Requested range not satisfiable	Der angeforderte Teil einer Ressource war ungültig oder steht auf dem Server nicht zur Verfügung
423	locked	Die angeforderte Ressource ist zurzeit gesperrt
429	to many requests	Der Client hat zu viele Anfragen in einem bestimmten Zeitraum gesendet
500	internal Server error	Dies ist ein „Sammel-Statuscode“ für unerwartete Serverfehler
502	bad Gateway	Der Server konnte seine Funktion als Gateway oder Proxy nicht erfüllen, weil er seinerseits eine ungültige Antwort erhalten hat
503	Service unavailable	Der Server steht temporär nicht zur Verfügung, zum Beispiel wegen Überlastung oder Wartungsarbeiten
504	Gateway Time-out	Der Server konnte seine Funktion als Gateway oder Proxy nicht erfüllen, weil er innerhalb einer festgelegten Zeitspanne keine Antwort von seinerseits benutzten Servern oder Diensten erhalten hat

Topic Modellierung

Die Publish/Subscribe Kommunikation wird über faye realisiert.

User

- Publish
  - */users/{id}* publiziert wenn ein User gelöscht oder geändert wird
  - */users/{id}/wishlist* publiziert Änderungen an der Wishlist

- Subscribe
  - */books* abonniert wenn ein neues Buch eingestellt wurde
  - */books/:isbn* abonniert wenn ein Buch mit gewünschter ISBN eingestellt wurde

Books

- Publish
  - */books* publiziert wenn ein neues Buch eingestellt wird
  - */books/:isbn* publiziert wenn bestimmtes Buch eingestellt wurde
- Subscribe
  - */users/{id}/wishlist* abonniert Änderungen an der Wishlist eines Users

Dienstgeber

Der Dienstgeber ist verantwortlich für die gesamte Datenlogik. Die Daten werden sowohl in-Memory, also im Arbeitsspeicher, als auch in einer seperaten *JSON* Datei abgespeichert und zur Verfügung gestellt. Zentrale Gegenstände in unserem System sind der *user* und die *books*. Diese bilden die **Primärressourcen** unserer Anwendung. Beide sind auch als **Listenressourcen** verfügbar, sodass die Liste aller *user* und *books* repräsentiert werden kann. Ein User verfügt gegebenfalls über eine *wishlist*. Diese wird als **Subressource** dem Nutzer zugeordnet. Mehr dazu in unserer REST-Modellierung.

- stellt die Plattform
- erstellt Benutzer
- erstellt Buchseite
- löscht Buchseiten (wenn das Buch verkauft ist)
- erstellt eine Liste mit allen Angebotenen Büchern einer Kategorie

Dienstnutzer

Der Dienstnutzer übernimmt die direkte Kommunikation mit dem Client. Er ist verantwortlich für das Rendern der Website und alle zusätzlichen Anfragen. Er ist außerdem zuständig für die Benutzerauthentifizierung. Zur Abfrage von Daten stellt er Anfragen an die API, nimmt eventuell noch Berechnungen auf diesen Daten vor, und leitet sie dann an den Client weiter.

- bezieht Daten von Dienstgeber
- benutzt Google Books API
- sucht Bücher
- stellt Leseproben zur Verfügung

Beschreibung der Anwendungslogik

Die Logik des Dienstnutzers besteht im Wesentlichen darin, als Vermittler zwischen Dienstgeber und dem Browser zu fungieren. Dabei werden anwendungsrelevante Daten vom Server verarbeitet, wie z.B. wenn ein *POST* - Befehl auf eine Ressource gesendet und daraufhin an den Dienstgeber weitergeleitet wird. Ursprünglich wurde überlegt verschiedene Funktionalitäten in die Anwendungslogik zu implementieren. Diese Idee wurde jedoch schließlich verworfen, da der damit zusammenhängende Aufwand zu groß geworden wäre.

Vorgehensweise und Irrwege

Im Folgenden soll der Arbeitsprozess im Verlauf des Projektes beschreiben und aufgetretene Schwierigkeiten aufgezeigt werden.
Das Vorgehen orientierte sich größtenteils an den Meilensteinen des Workshops. Aufgaben die während der Workshop Termine nicht erledigt

werden konnten, wurden außerhalb der Termine erarbeitet.
Der Prozess zur Erstellung und Bearbeitung des Projekts kann in die folgenden Phasen eingeteilt werden:

Die Ideenfindung
für die Realisierung des Projekts benötigten wir eine passende Idee die den Anforderungen entsprechen würde. Das System sollte nach den Prinzipien von REST (Representational State Transfer) entwickelt werden und über einen Dienstgeber und einen Dienstnutzer verfügen. Letzten Endes haben wir uns für ein Bücher Verkaufsportal für Studenten entscheiden.

Dienstgeber und Dienstnutzer
Auf die Projektidee folgt die Implementierung. Die Definition der erforderlichen Ressourcen des Systems und deren Implementierung als auch Nutzung wurden zuerst umgesetzt. Dies wurde beispielsweise durch die Erstellung von Tabellen für die REST-Ressourcen und ihre HTTP-Methoden sowie deren Code-Umsetzung verwirklicht. Im Anschluss konnte schließlich der Dienstnutzer realisiert werden.

REST-Konformität
Da die REST-Prinzipien als Vorgabe für die Realisierung galten, wurden diese als Orientierung genutzt.

Client Umsetzung
Um dem System eine Basis zu liefern, haben wir uns in der Realisierungsphase für eine Webseite entschieden. So können sämtliche Funktionen präsentiert und ausgeführt werden. Diese galt es zu erstellen und zu gestalten. Während der Umsetzung gab es einige Irrwege und Probleme, die zu bewältigen waren.
Eine Schwierigkeit bestand darin, mit den neuen Prinzipen und Techniken umzugehen. Beispielsweise hatten wir zu Beginn große Probleme damit die Datenhaltung zu implementieren. Aufgrund dieser und weiterer Probleme befand sich das System in ständiger Weiterentwicklung. Wegen von unzureichenden Informationen benötigte diese Implementierung sehr viel Zeit, was zur Schmälerung der umgesetzten Funktionalitäten zu einem bestimmten Zeitpunkt führte.

Fazit

Im Folgenden soll das Projekt kritisch reflektiert werden. Dabei ist zu sagen, dass das Projekt zum derzeitigen Zeitpunkt noch nicht abgeschlossen ist. Es werden nachträglich noch Funktionen implementiert und die Dokumentation weitergeführt sowie vervollständigt. Wie bereits oben unter dem Unterpunkt *Vorgehensweise und Irrwege* erwähnt, haben sich die Implementierung der Datenhaltung so wie die der Routen Auslagerung auf einen Webserver leicht verzögert. Dies war hauptsächlich auf die Unerfahrenheit der Teammitglieder und der verwirrenden Nummerierung der Screencasts zurückzuführen. Obwohl zu Beginn des Projekts geplant war eine Vorschläge-Option basierend auf den Artikeln auf der Wishlist zu implementieren, hat sich diese jedoch als zu Zeitaufwändig erwiesen und wurde deshalb verworfen.Deshalb wurde diese durch eine Vorschläge-Option basierend auf einem Counter ersetzt, welcher dem User die beliebtesten Artikel aufzeigt.
In den Semesterferien werden wir noch weiter am Projekt arbeiten und dieses erweitern und verfeinern. Folgende Implementierungen sind noch für die Ferienzeit geplant In den Semesterferien werden wir noch weiter am Projekt arbeiten und dieses erweitern und verfeinern. Folgende Implementierungen sind noch für die Ferienzeit geplant:

- Die Detailseite für die Einzelnen Bücher ist bisher nur in JSON verfügbar, soll aber noch realisiert werden. Auf der Detailseite des Buches findet man dann die Informationen zum Buch, sowie die Informationen vom Verkäufer (Buch Zustand, Preis etc..)
- Auf der Buchseite wird Angezeigt, ob dieses gerade Verfügbar oder bereits Verkauft ist. (Online/Offline)
- Eine Wishlist wird erstellt und der Benutzer bekommt eine Meldung, sobald ein von Ihm gewünschtes Buch wieder verfügbar ist (Weiterentwicklung von Pub/Sub mit faye)
- Die Buttons (recommend/delete) auf der Seite werden lauffähig sein
- Die Anwendungslogik weiterentwickeln:
    - Bücher sollen entweder bewertbar sein, und der Dienstnutzer errechtnet die durchschnittlichen Buchbewertungen, oder
    - Bücher erhalten einen ‘recommendedCounter’ und das System sortiert, welche Bücher am häufigsten Empfohlen wurden

Arbeitsmatrix

Aktivität	Fabian Fuchs	Clara Meschkat	Eugen Scherer
Projektidee	33,33 %	33,33 %	33,33 %
Exposé	40 %	30 %	30 %
Wiki	30 %	40 %	30 %
Implementierung des Dienstgebers	25 %	25 %	50 %
Implementierung des Dienstnutzers	40 %	30 %	30 %
Dokumentation	30 %	40 %	30 %
Zusammenfassung:	33,33 %	33,33 %	33,33 %