

Dokumentation
zum Semesterprojekt „Verschenkbörse“
im Modul
Verteilte Systeme

Semester
Wintersemester 2016/ 2017

Gruppenmitglieder
Carsten Gedrat
Sarah Roehricht
Joachim Zerbig
Leonid Immel

Inhaltsverzeichnis

1. Planung.....	3
1.1 Umfang des Projekts	3
1.2 Scribbles.....	3-5
1.3 Eingesetzte Techniken.....	6
1.4 Einteilung der Aufgaben.....	6
2. Implementierung.....	7
2.1 Frontend-Team.....	7
2.2 Backend-Team.....	8-9
2.3 Screenshots.....	9-11
3. Projekt-Evaluation.....	11

1. Planung

1.1 Umfang des Projekts

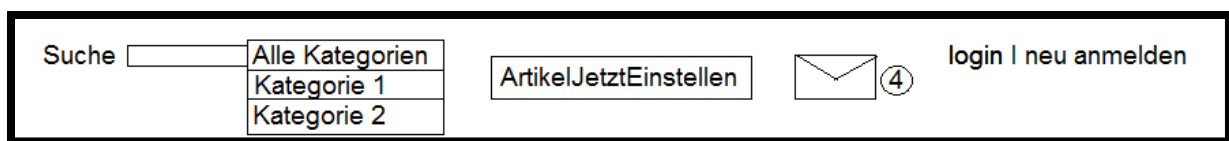
Zu Beginn musste festgelegt werden, welchen Umfang unser Projekt besitzen sollte. Hierzu zählte nicht nur der Funktionsumfang an sich, sondern auch die Bereitstellung des User-Interfaces. Dieses sollte schlank ausfallen, und den User intuitiv durch die Applikation führen.

Für unser Thema „Verschenkbörse“ wurden deshalb folgende Seiten angelegt:

Anmelden, Benutzerkonto anpassen, Benutzerkonto, eigene Artikel, eingestellter Artikel, erfolgreich registriert, neuen Artikel einstellen, Registrieren und Anzeigen der Suchergebnisse

1.2 Scribbles

Um sich einen Überblick zu verschaffen, wurden Scribbles angelegt. Auch wenn sich im Verlauf des Projekts die Oberfläche teils geändert hat, konnte so bereits eine erste Einschätzung über den Aufwand und Umfang gemacht werden. Hier eine Auswahl der Scribbles:



The header of the application is displayed within a black rectangular border. On the left, there is a search bar with the placeholder text 'Suche' and an empty input field. To the right of the search bar is a dropdown menu with three options: 'Alle Kategorien', 'Kategorie 1', and 'Kategorie 2'. Further right is a button labeled 'ArtikelJetztEinstellen'. To the right of the button is an envelope icon followed by a circle containing the number '4'. On the far right, there is a link that reads 'login | neu anmelden'.

Abbildung 1: Header der Applikation



The footer of the application is displayed within a black rectangular border. It contains two links: 'Impressum' on the left and 'AGB' on the right.

Abbildung 2: Footer der Applikation

Anrede	<input type="text"/>
Vorname	<input type="text"/>
Nachname	<input type="text"/>
Strasse	<input type="text"/>
HausNr	<input type="text"/>
PLZ	<input type="text"/>
Ort	<input type="text"/>
E-Mail	<input type="text"/>
Benutzername	<input type="text" value="steht fest, nicht veränderbar"/>
Passwort	<input type="password"/>
Passwort bestaetigen	<input type="password"/>
<input type="button" value="bestaetigen"/> <input type="button" value="abbrechen"/>	

Abbildung 3: Neu Anmelden

<div style="border: 1px solid black; width: 150px; height: 100px; display: flex; align-items: center; justify-content: center; margin: 10px;"> <div style="text-align: center;"> <h1 style="margin: 0;">Foto</h1> </div> </div>	<p><Titel> von <Benutzername></p>
	<p><PLZ> <Ort></p>
	<p><Straße><Nummer></p>
	<p><Beschreibung></p>
	<p>...</p> <p>...</p> <p>...</p> <p>...</p> <p>...</p>

Abbildung 4: Artikeldarstellung

A login form with a black border. It contains two text input fields: one for 'Benutzername' and one for 'Passwort'. Below the fields are two buttons: 'bestaetigen' and 'abbrechen'.

Benutzername

Passwort

Abbildung 3: Login

A form titled 'Neuen Artikel einstellen' with a black border. It contains a large square area for a photo with a 'Foto hochladen' button. To the right are input fields for 'Titel', 'Beschreibung' (a text area), 'Ort', 'PLZ', and 'Kategorie' (a dropdown menu). At the bottom are 'bestaetigen' and 'abbrechen' buttons.

Neuen Artikel einstellen

Textarea

Dropdownmenu

Abbildung 4: Neuer Artikel Einstellen

1.3 Eingesetzte Techniken

Für das Semesterprojekt wurden Vorgaben festgeschrieben, welche Technologien zum Umsetzen des Projektes verwendet werden sollten, um die API zu realisieren. Hierzu zählte die No-SQL Datenbankbindung MongoDB. Mit Hilfe der Programmiersprache NodeJS und der Technik der objektrelationalen Abbildung Mongoose, wurden die erstellten Objekte in der Datenbank abgelegt.

Hierzu wurden Schemata des Nutzers und der Artikel erstellt. Die Befüllung der Datenbank mit User- und Artikeldaten erfolgt durch Formularübergaben, die über ein HTML User-Interface geschehen. Neben NodeJS wurde auch JavaScript und JQuery eingesetzt um einzelne Funktionen auf den bereitgestellten HTML-Websites zu realisieren.

Für das Stylen der HTML-Seiten wurde ein separates Cascading Stylesheet eingesetzt, welches über eine Verlinkung in die HTML-Seiten inkludiert wurde.

1.4 Einteilung der Aufgaben

Die Aufgaben wurden paarweise aufgeteilt. Sarah Roehricht und Carsten Gedrat realisierten den Backend-Bereich, während Leonid Immel und Joachim Zerbis sich um den Frontend-Bereich kümmerten. Auch wenn es größtenteils eine Trennung zwischen Front- und Backend gab, kam es immer wieder zu Überschneidungen. Sei es durch Funktionstests während der Implementierung oder Hilfestellungen in bestimmten Bereichen. Hierzu zählten Arbeiten am Server, den HTML-Seiten und dem Stylesheet, sowie der Passwortauthentifizierung und der Möglichkeit des Abspeicherns von Bildern.

2. Implementierung

2.1 Frontend-Team

Das Frontend-Team war zuständig für die Erstellung aller HTML-Seiten mit ihrem Header und Footer, den jeweiligen Inhalten wie Formulare zur Eingabe von Nutzerdaten oder Suchfunktion, den Verlinkungen, die über die Server.js realisiert wurden, sowie das Erstellen des CSS-Stylesheets. Des Weiteren wurde an Aufgabenbereichen gearbeitet, die aufgrund von Problemen oder effizienteren Lösungsansätzen zu einem späteren Zeitpunkt aufgegeben wurden. Hierzu zählte eine erste Version der Bildabspeicherung und ein erster Server

Des Weiteren wurde mit JavaScript eine Authentifizierung des Anmeldeformulars realisiert, welche die Nutzereingaben auf Korrektheit überprüft und bei Bedarf alert-Fenster aufruft, um den Benutzer auf die fehlerhaften Eingaben aufmerksam zu machen. Eine Anmeldung ist somit erst nach korrekter Eingabe der Felder möglich.

Es folgt die genaue Aufgabenverteilung:

Joachim Zerbig

- Erstellung der HTML-Seiten mit ihren Inhalten wie Formulare, Buttons, Links etc. auf Basis von UTF-8 Codierung
- Validierung der Eingabefelder bei Neuanmeldung (keine Ziffern in Namer, PLZ genau 5 Stellen lang), Hinweis durch alert()-Fensteraufrufe
- Erstellung und Einbindung des CSS (größtenteils obsolet geworden)
- Nutzung und Einbindung von CSS durch Bootstrap
- Erstellung des ersten Servers
- Arbeiten am Server (app.get Methoden mit erstellt, Weiterleitung auf HTML-Seiten ermöglicht)

Leonid Immel

- Erstellung der HTML-Seiten mit ihren Inhalten wie Formulare, Buttons, Links etc.
- Erstellung und Einbindung des CSS
- Abspeichern von Bildern als base64 (Rohversion)

2.2 Backend-Team

Das Backend-Team befasste sich mit der Erstellung der Datenbank und ihrer Verbindung mit NodeJS. Es wurden Schemata für Nutzer und Artikel angelegt, sowie (ursprünglich) Funktionen zum flexiblen Speichern von Objekten erstellt, die nun jedoch obsolet geworden sind. Das Abspeichern von Artikeln wurde ebenso realisiert, wie die Möglichkeit des Abspeicherns von Bildern als base64. Des Weiteren wurden routes angelegt und das Anzeigen von Artikeln aus der Datenbank realisiert. Die jeweiligen Funktionen wurden in diverse JavaScript-Dateien ausgelagert. Die passport.js Datei beinhaltet beispielsweise die Authentifizierung der Benutzer auf Basis einer Registrierung mit Passwortverschlüsselung, sowie ein Login mit Benutzername und Passwort.

Zu weiteren Arbeiten zählte das Erstellen der Texte für AGB und Impressum, das Einbinden von Icons, sowie der JavaScript-Funktion Suchen. Hinzu kamen Arbeiten mit Embedded JavaScript für die Seiten „eigene Artikel“ und „profile“.

Es folgt die genaue Aufgabenverteilung:

Carsten Gedrat

- Erstellung der MongoDB,
- Verbindung Node JS, DB
- Schemata angelegt für User und Artikel,
- Funktionen zum flexiblen Speichern von Objekten erstellt(Wovon wir jetzt keine mehr benutzen),
- Artikel abspeichern realisiert,
- Bild als base64 abgespeichert und eingepflegt(Rohbau von Leonid Immel)
- routes mit Sarah Roehricht gepflegt und angelegt
- eigene Artikel anzeigen und aus der Datenbank auslesen mit Sarah Roehricht realisiert

Sarah Roehricht

- Authentifikation
 - Registrierung mit Passwortverschlüsselung
 - Login mit Benutzername und Passwort
 - passport.js
 - registrieren.html
 - Bootstrap eingebunden
 - Suchfunktion nach Artikeltitel
 - suchergebnisse.html
- routes.js 50/50 mit Carsten
 - eigeneArtikel
- Embedded JavaScript in:
 - eigeneArtikel.html
 - profile.html
- komplette Überarbeitung anmelden.html

- Strukturen ins Projekt eingefügt
 - Unterteilung views/js/config
- Texte in AGB und Impressum
- Icons eingebunden

2.3 Screenshots

Es folgen Screenshots der fertigen Applikation, sowie die Ordnerstruktur des Projekts, wie sie in Eclipse angelegt wurde.

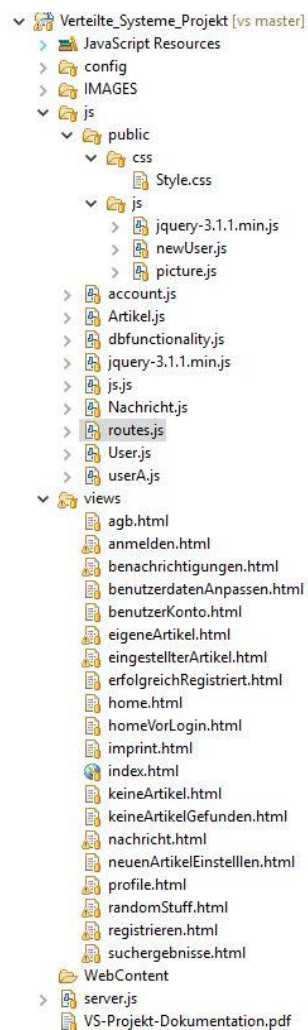


Abbildung 5: Projektstruktur, Server einzelstehend, Views der HTML Seiten separat, im Public Folder notwendige CSS und JS Dateien, die dem Client zur Verfügung stehen, Dateien nicht im Public Folder stehen dem Client nicht zur Verfügung, Views müssen über die REST-

```

var userSchema = mongoose.Schema({
  local      : {
    anrede    : String,
    vorname   : String,
    nachname  : String,
    Strasse   : String,
    Hausnummer : String,
    Plz       : String,
    Ort       : String,
    email     : String,
    benutzername : String,
    password  : String,
    _id       : String
  },
  facebook   : {
    id        : String,
    token     : String,
    email     : String,
    name      : String
  },
  twitter    : {
    id        : String,
    token     : String,
    displayName : String,
    username  : String
  },
  google     : {
    id        : String,
    token     : String,
    email     : String,

```

Abbildung 6: User Schema welches für Facebook, Twitter und Google Anbindung ausgelegt ist, und modular verwendet werden kann.

```

//Home
app.get('/home',function(req,res){
  console.log("Got a GET request for the home");
  res.render('home.html', { message: req.flash('signupMessage') });
});

//HomeVorLogin
app.get('/homeVorLogin',function(req,res){
  console.log("Got a GET request for the homeVorLogin");
  res.render('homeVorLogin.html', { message: req.flash('signupMessage') });
});

//Impressum
app.get('/imprint',function(req,res){
  console.log("Got a GET request for the imprint");
  res.render('imprint.html', { message: req.flash('signupMessage') });
});

//NeuenArtikelEinstellen
app.get('/neuenArtikelEinstellen',function(req,res){
  console.log("Got a GET request for the neuenArtikelEinstellen");

  if (req.isAuthenticated()){
    res.render('neuenArtikelEinstellen.html', { message: req.flash('signupMessage') });
  }else{
    res.render('index.html', { message: req.flash('signupMessage') });
  }
});

```

Abbildung 7: Ausschnitt der routes.js, welches die Get Requests verarbeiten, und die entsprechenden .html Dateien rendern, bei Datensensiblen Get Requests wird überprüft ob der User eingeloggt ist, und eine Session hat.

```

<form method="post" action="/article/new">

<label for="titel">Titel</label>
<input type="text" name="article[titel]">

<label for="Beschreibung">Beschreibung</label>
<input type="text" name="article[beschreibung]">
<label for="Bild">Bild</label>
<input id="inputFileToLoad" type="file" onchange="encodeImageFileAsURL();" />
<label for="Base64">Base64</label>
<input type="text" name="article[foto]" id="base64t" disabled><br>
<input type="submit" value="Einstellen">

```

Abbildung 8: HTML Post Form, hier als Beispiel für den Artikel, sendet entsprechende Daten mit einem post request an /article/new, welcher vom Backend verarbeitet wird.

3. Projekt-Evaluation

Die zunächst strikt vorgesehene Trennung zwischen Backend- und Frontend-Team war nicht einzuhalten, da es immer wieder zu Überschneidungen kam, oder Probleme gelöst werden mussten.

Problemlos verlief die Erstellung der HTML-Seiten und die Einbindung dieser durch den Server. Ursprünglich sollten die Header und Footer der HTML-Dokumente dynamisch eingebunden werden. Nach mehreren Fehlschlägen der Einbindung wurde aufgrund von Zeitdruck und anderer Prioritäten darauf zurückgegriffen, die Header und Footer fest in die HTML-Dokumente zu integrieren. Ein ähnliches Problem ergab sich zunächst auch durch den Versuch der Einbindung der CSS-Links. Dieses Problem konnte schließlich gelöst werden und die Lösung dieses Problems hätte wohl auch zur Lösung des dynamischen Bindens beigeführt, doch aufgrund von Zeitdruck und dem Vorziehen der Implementierungen weiterer Features hinten angestellt und schließlich nicht mehr realisierbar.

Ein weiteres Problem ergab sich aus dem Blockieren der benötigten Ports im Hochschulnetzwerk. Hierdurch war ein Testen an der HS selbst nicht möglich. Hinzukamen Probleme durch das Team selbst. Ein Umzug eines Teammitglieds hatte zur Folge, dass es keinen Zugang zum Internet außerhalb der HS besaß. Durch das bereits erwähnte Problem der blockierten Ports ergaben sich hierdurch neue Probleme, was zu Kommunikationsschwierigkeiten führte.

Um die Effizienz zu steigern, wurde das Projekt auf GitHub hochgeladen und die Kommunikation durch Skype realisiert. Somit konnte ein schneller und effizienter Informationsfluss zwischen den Teammitgliedern erfolgen und Fragen schneller geklärt werden.