Tatyana Merlo – Natalie Stalder – Nadja Stadelmann

Travelling I

M133 – Webapplikation realisieren

# Einleitung

Bei einer Diskussion über unsere bisherigen Reiseziele ist uns aufgefallen, dass es schwierig sein kann, sich an alle Orte zu erinnern, welche man bisher besucht hat. Daher kam uns die Idee, dass dies in einem Webprogramm gut ersichtlich dargestellt werden kann. Ausserdem wären die Daten so von überall her abrufbar (angenommen wir stellen die Seite online).

## Systemanforderungen

Wir testen die Anwendung in Google Chrome und in Firefox. Ob die Seite in anderen Browsern ebenfalls funktioniert können wir daher nicht garantieren.

Der Webserver muss ein PHP-Interpreter installiert haben, die Browser der Clients muss JavaScript-fähig sein.

Ausserdem muss eine Verbindung vom Webserver an einen Datenbankserver vorhanden sein, damit die Daten abgespeichert und abgerufen werden können.

## Dokument

### Aufbau

* Analyse
  + Beschrieb und erste Aufsplittung in einzelne Teilbereiche
* Vorgehen
  + Abhängigkeiten, Aufgabenübersicht
* Design
  + Detaillierte Analyse DB-Design
  + Detaillierte Analyse Programmstruktur
* Implementation
* Testing
  + Testcases basieren auf UseCases

Inhalt

[Einleitung 1](#_Toc497118050)

[Systemanforderungen 1](#_Toc497118051)

[Dokument 1](#_Toc497118052)

[Aufbau 1](#_Toc497118053)

[Vision 3](#_Toc497118054)

[Analyse 4](#_Toc497118055)

[Anforderungen 4](#_Toc497118056)

[Must have 4](#_Toc497118057)

[Nice to have 4](#_Toc497118058)

[Use Cases 4](#_Toc497118059)

[Design 8](#_Toc497118060)

[DB 8](#_Toc497118061)

[Tabellenbeschrieb 8](#_Toc497118062)

[User 8](#_Toc497118063)

[Location 8](#_Toc497118064)

[Post 8](#_Toc497118065)

[Picture 8](#_Toc497118066)

[Visited 8](#_Toc497118067)

[Designanpassungen 8](#_Toc497118068)

[Programm 8](#_Toc497118069)

[Model 8](#_Toc497118070)

[View 9](#_Toc497118071)

[Controller 9](#_Toc497118072)

[Designanpassungen 9](#_Toc497118073)

[Implementation 10](#_Toc497118074)

[Implementierung 10](#_Toc497118075)

[Testing 0](#_Toc497118076)

# Vision

Man bereist die Welt und möchte möglichst viele Orte sehen. Doch sich dabei auf einen Schlag an alle Orte zu erinnern, die besten Fotos an einem zentralen Ort lagern und allenfalls Freunden eine Hotel- oder Restaurantempfehlung liefern – das fällt oftmals schwer.

Für all diese Probleme bringen wir mit dem Travelblog ‘Travelling I’ die Lösung: Auf einer Karte kann eingetragen werden, an welchem Ort man bereits gewesen ist. Das entsprechende Land wird dabei farbig eingefärbt und die Stadt mittels eines Punktes markiert. So erhält man eine Übersicht über seine vergangenen Reisen, die man überall auf der Welt über das Internet abrufen kann!

Zu jedem Ort kann auch ein Blogeintrag geschrieben werden, um tolle Hotels, Restaurants oder Ausflugsziele festzuhalten. Dazu kann man seine schönsten, lustigsten oder emotionalsten Fotos hochladen, um immer wieder in diesen Erinnerungen zu schwelgen.

Damit diese Daten nicht einfach öffentlich preisgegeben werden, muss man sich dazu zuerst registrieren und jeweils einloggen. So werden die Ferienerlebnisse vor unerwünschten Blicken geschützt.

In weiteren Schritten stellen wir uns vor, dass sich die User Restaurants oder vorhandene Aktivitäten an den spezifischen Orten anzeigen lassen können. Da das Wetter für Ausflüge und das Gepäck ebenfalls wichtig ist, wird auch dieses eingebunden.

Um die Suche durch seine Einträge zu erleichtern, werden später Kategorien eingebunden. So kann der User zum Beispiel die Kategorie ‘Hotel’ angeben und später ganz einfach die entsprechend markierten Einträge filtern.

Doch bei den vergangenen Reisen soll es nicht bleiben! Pläne zu schmieden für neue Ferien ist ebenfalls spannend und erfreulich. Damit dies auch locker vom Hocker geht, werden den Usern entsprechende Flüge und Hotels angezeigt, die für sie geeignet sein können.

# Analyse

## Anforderungen

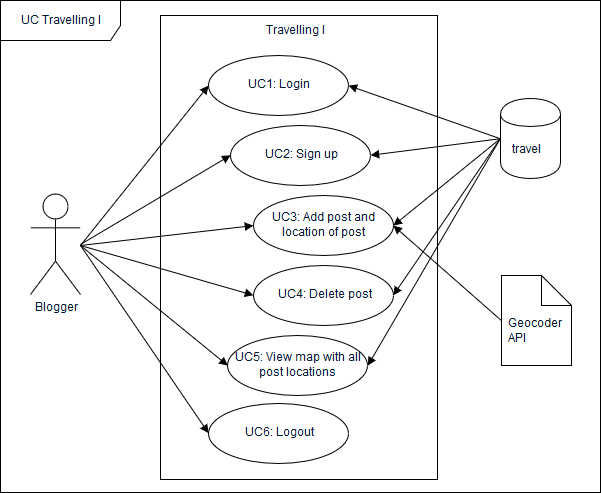
### Must have

1. Login / Registrierung
2. Karte – wo warst du?
   1. Nadel auf Stadt
   2. Blogeintrag mit Fotos

## Nice to have

1. Google Maps einbindung
   1. Restaurants in der jeweiligen Stadt suchen
2. Wetter anzeige
   1. Name der Stadt eingeben 🡪 entsprechendes Wetter angeben
3. Vorhandene Aktivitäten anzeigen lassen
   1. Europaweit / Weltweit
   2. Tripadvisor API?
4. Blogmässige Notizen erweitern z.B. Kategorien
   1. Restaurant
   2. Hotels
   3. Festivals
   4. Einkaufscentern
   5. ….
5. Plan für nächste Reise(n)
   1. gute Flüge und Hotels anzeigen lassen

## Use Cases



|  |  |
| --- | --- |
| **Identifikation** | **UC1: Login** |
| **Beschreibung** | Ein Blogger logt sich ein. |
| **Primärer Actor** | Blogger |
| **Vorbedingungen** | Der Blogger muss bereits einen Account haben (Sign up) |
| **Trigger** | Blogger wählt "Login" |
| **Standardablauf** | 1. Link von Travelling I eingeben  2. Benutzernamen und Passwort eingeben und "Login" klicken  3. Blogger wird eingeloggt, wenn der Benutzername und das Passwort mit den in der "travel"-Datenbank abgespeicherten Werten übereinstimmt  4. Die Sessionvariable "username" wird gesetzt |
| **Alternative Abläufe** | 2a. Falscher (nicht in Datenbank erfasster) Benutzernamen eingegeben 🡪 Blogger kann sich nicht einloggen, Login-Seite wird erneut – mit einer Fehlermeldung - angezeigt  2b. Blogger gibt falsches Passwort an 🡪 Es wird ebenfalls eine Fehlermeldung ausgegeben und die Login-Seite neu geladen (der Benutzername bleibt)  2c. kein Input in den Feldern |

|  |  |
| --- | --- |
| **Identifikation** | **UC2: Sign up** |
| **Beschreibung** | Ein Blogger erstellt einen Account. |
| **Primärer Actor** | Blogger |
| **Vorbedingungen** | - |
| **Trigger** | Blogger wählt "Sign up" |
| **Standardablauf** | 1. Link von Travelling I eingeben  2. Auf der Login-Seite auf die Sign-Up-Seite wechseln: "Sign up here" klicken  3. Benutzernamen, Passwort sowie eine Passwortbestätigung eingeben und "Sign up" wählen  4. Bloggerkonto wird erstellt und in der Datenbank "travel" abgespeichert  5. Die Sessionvariable "username" wird gesetzt  6. Der Blogger gelangt auf die Startseite im Logged-In-Bereich |
| **Alternative Abläufe** | 3a. Bereits in Datenbank "travel" erfasster Benutzernamen eingegeben 🡪 Bloggerkonto kann nicht erstellt werden 🡪 Sign-Up-Seite wird erneut geladen mit einer Fehlermeldung  2b. Blogger gibt nicht übereinstimmende Passwortbestätigung ein 🡪 Es wird ebenfalls eine Fehlermeldung ausgegeben und die Login-Seite neu geladen (der Benutzername bleibt)  2c. Blogger sendet ein leeres Formular ab 🡪 Fehlermeldung und der Blogger bleibt auf der Sign-Up-Seite |

|  |  |
| --- | --- |
| **Identifikation** | **UC3: Add post and location of post** |
| **Beschreibung** | Ein Blogger erstellt einen Post zu einem bestimmen Ort auf der Welt. |
| **Primärer Actor** | Blogger |
| **Vorbedingungen** | - |
| **Trigger** | Blogger klickt auf "Add Post" |
| **Standardablauf** | 1. Einloggen  2. In "Home" eine Ortschaft aussuchen und "Add Location" klicken  3. Posttitel, Posttext und die Bilder auswählen  4. "Submit" wählen  5. Post wird erstellt, in der Datenbank "travel" abgespeichert und als Standort auf der Blogger spezifischen Weltkarte markiert |
| **Alternative Abläufe** | 2a. Die ausgewählte Location hat bereits einen Posteintrag 🡪 eine Fehlermeldung wird angezeigt  2b. Die Bilder, die hochzuladen sind, sind nicht im richtigen Format/ in der richtigen Grösse etc. 🡪 Es wird ebenfalls eine Fehlermeldung ausgegeben, der Post wird ohne Bilder resp. mit den Bildern abgespeichert, welche hochgeladen werden konnten |

|  |  |
| --- | --- |
| **Identifikation** | **UC4: Delete Post** |
| **Beschreibung** | Ein Blogger löscht einer seiner Posts. |
| **Primärer Actor** | Blogger |
| **Vorbedingungen** | - |
| **Trigger** | Blogger wählt für bestimmten Post "Delete Post". |
| **Standardablauf** | 1. Blogger geht auf seinen Blog und wählt den "Delete"-Button, welche hinter jedem Post angezeigt wird.  2. Der Post wird gelöscht |
| **Alternative Abläufe** | 2a. Der Posteintrag kann nicht gelöscht werden 🡪 eine Fehlermeldung erscheint. |

|  |  |
| --- | --- |
| **Identifikation** | **UC5: View map with all post locations** |
| **Beschreibung** | Die Home-Seite jedes Bloggerkontos zeigt eine Karte mit all den Ortschaften, zu welchen der Blogger einen Post hat. |
| **Primärer Actor** | Blogger |
| **Vorbedingungen** | - |
| **Trigger** | Der Blogger loggt sich ein  Ein neuer Benutzer erstellt ein Konto  Der Blogger wählt im Logged-In-Menu "Home" |
| **Standardablauf** | 1. In der Navigation "Home" wählen  2. Die Karte wird geladen mit allen und alle in der "travel"- Datenbank abgespeicherten Ortschaften markiert  3. Blogger wird eingeloggt |
| **Alternative Abläufe** | 2a. Der Blogger loggt sich ein und landet auf der Startseite "Home"  2b. Der Blogger erstellt ein Konto, wird gleich eingeloggt und landet auf der Startseite "Home" |

|  |  |
| --- | --- |
| **Identifikation** | **UC6: Logout** |
| **Beschreibung** | Ein Blogger logt sich aus. |
| **Primärer Actor** | Blogger |
| **Vorbedingungen** | - |
| **Trigger** | Blogger wählt im Logged-In-Bereich die Aktion "Logout" |
| **Standardablauf** | 1. Eingeloggt sein  2. "Logout" wählen  3. Blogger wird ausgeloggt |
| **Alternative Abläufe** | Mit anderen besprechen |

# Design

## DB

### Tabellenbeschrieb

Hier Anpassungen vornehmen sobald Taty die neue DB Struktur online hat.

#### User

Der User braucht einen Usernamen sowie das Passwort. Der Username ist unique definiert. Das Passwort ist mittels BCrypt verschlüsselt – das heisst, es benötigt eine Länge von CHAR(60). Ausserdem wird der Salt CHAR(60) abgespeichert, er besteht aus einer Zahl.

#### Location

Hier werden die besuchten Orte abgespeichert. Wir speichern den Namen der Stadt, sowie die Längen- und Breitengrade in dieser Tabelle ab, da Google Maps die Nadeln mittels der Koordinaten setzt.

Formate: Längengrad Breitengrad

-000.0000 -00.0000

VARCHAR(45) VARCHAR(45)

Index: Unique über Longitude und Latitude.

#### Post

Im Post wird das Datum, der Titel und ein Text eingespeichert. Der Text ist nicht zwingend, damit ein User auch Bilder ohne Text hochladen kann.

#### Picture

Der Bildername, welcher angezeigt wird, wird gespeichert. Weiter wird der Filename gespeichert, damit das passende Bild geladen werden kann.

#### Visited

Zwischentabelle zwischen dem User und der Location. Mehrere User können dieselbe Stadt besuchen und ein User kann mehrere Städte besuchen. Weiter ist hier der Fremdschlüssel zum zugehörigen Post eingespeichert.

### Designanpassungen

## Programm

In unserer Filestruktur achten wir sehr darauf, dass MVC-Prinzip einzuhalten. Die verschiedenen Programmiersprachen möchten wir ebenfalls möglichst voneinander abgrenzen. So soll in unseren HTML-Files kein PHP oder JS ersichtlich sein.

### Model

Das eigentliche Datenmodell ist unsere Datenbank ‘travel’. Dennoch müssen wir natürlich auch in unserer Applikation etwas implementieren, was die Daten aus der Datenbank holt und für die Weiterverarbeitung zur Verfügung stellt. Dies machen wir über die Library ‘Doctrine’. Das Konzept dahinter ist, dass eine DB-Tabelle genau eine Klasse, beziehungsweise eine Entity im PHP-Programm besitzt. Sie besteht aus Klassenvariablen mit Getter- und Setter-Methoden, wobei jede Variable für ein DB-Attribut steht.

### View

Unsere Views sind allesamt als HTML-Dateien abgespeichert. Zwei Templates beinhalten die Grundstruktur der Views. Eines ist für die eingeloggte Ansicht gedacht, das andere (ohne Navigation) wird für die Aktionen ausserhalb des eingeloggten Bereiches verwendet. Sprich, das Login und das SignUp erhalten ein etwas anderes Design als beispielsweise der Blog eines Users.

Allgemein ist das Layout sehr schlicht gehalten. Es soll nicht von den spannenden Reisen der User ablenken, schon gar nicht, da er seine eigenen Bilder hochladen kann.

### Controller

Der Einstiegspunkt der Applikation ist die Datei ‘index.php’, welche auf die ‘route.php’ verweist. Darin sind alle Controller gehandelt. Die abstrakte Klasse ‘Controller’ wird von allen spezifischen Controller extended. Jeder handelt einige Funktionen in einem speziellen Zuständigkeitsbereich. Der LoginController ist zum Beispiel für die Anzeige des Loginformulars zuständig. Gleichzeitig checkt er die Usereingaben und logt den User bei korrekten Eingaben ein.

Die gesamte Logik ist also in den Controller eingebaut, welche die Usereingaben entgegennehmen, die entsprechenden Funktionen ausüben und danach die korrekten Views anzeigen.

In der URL sind die jeweiligen Controller und die verwendeten Methoden ersichtlich.

### Designanpassungen

//Filestruktur einfügen?

# Implementation

## Implementierung

# Testing

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Testfall Nummer | Use Case | Testbeschrieb | Erwartetes Resultat | Tatsächliches Resultat | Tester | Testdatum | Status |
| **1** | **UC1** | **Login** |  |  |  |  |  |
| 1.1 |  | Ein Blogger loggt sich mit in der DB vorhandenem Usernamen und Passwort ein | Der Blogger wird eingeloggt, er gelangt auf seine Home-Seite | Der Blogger wird eingeloggt, er gelangt auf seine Home-Seite | Natalie | 30.10.2017 | OK |
| 1.2 |  | Ein Blogger will sich mit nicht in der DB vorhandenen Usernamen einloggen. | Der Blogger gelangt erneut auf die Login-Seite, die ihm die Fehlermeldung "Invalid Credentials." ausgibt. | Der Blogger gelangt erneut auf die Login-Seite, die ihm die Fehlermeldung "Invalid Credentials." ausgibt. | Natalie | 30.10.2017 | OK |
| 1.3 |  | Ein Blogger will sich mit in DB vorhandenem Usernamen und falschem Passwort einloggen. | Der Blogger gelangt erneut auf die Login-Seite, die ihm die Fehlermeldung "Invalid Password." ausgibt. | Der Blogger gelangt erneut auf die Login-Seite, die ihm die Fehlermeldung "Invalid Password." ausgibt. |  |  |  |
| **2** | **UC2** | **Sign Up** |  |  |  |  |  |
| 2.1 |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |