



Fachhochschule Köln
Cologne University of Applied Sciences

FACHHOCHSCHULE KÖLN FAKULTÄT FÜR INFORMATIK UND
INGENIEURWISSENSCHAFTEN

ENTWICKLUNGSPROJEKT INTERAKTIVE SYSTEME

Meilenstein 4

Campus Gummersbach
im Studiengang
Medieninformatik

Betreut von:

Prof. Dr. Kristian Fischer
Prof. Dr. Gerhard Hartmann
Robert Gabriel, B. Sc.

ausgearbeitet von:

DERYA ERGUEL
SINEM KAYA

26. Juni 2015

Inhaltsverzeichnis

1 Datenstrukturen	2
1.1 XML-Dateien	3
1.2 XML-Schemata	5
1.3 Entity-Relationship-Modellierung	8
2 WBA-Modellierung	9
2.1 Ressourcen	9
2.1.1 User	9
2.1.2 Profil	9
2.1.3 Karte	9
2.1.4 Nachricht	10
2.1.5 Wetter	10
2.1.6 Route	11
2.1.7 Berichte	11
2.2 Ressourcentabelle	13
2.3 Pseudocode	16
3 Prototyp UI	21
3.1 Gestaltungslösungen	21
3.1.1 Design Principles	21
3.1.2 Paperbased-Prototyping	23
3.2 Fazit zu der Gestaltungslösung	36
4 Projektplan	37
5 Literaturverzeichnis	39
6 Anhang	40
6.1 Iteration	40
6.1.1 Szenarien	40
6.1.2 Funktionale Anforderungen	42
6.1.3 Organisatorische Anforderungen	42
6.1.4 Qualitative Anforderungen	43
6.1.5 Prototyp	43

1 Datenstrukturen

In diesem Kapitel wird die Strukturierung der Daten ausführlich besprochen, die notwendig sind, um die technische Umsetzung zu realisieren und das Repräsentieren der Daten gerecht realisieren zu können.

Ein Benutzer gibt verschiedene Daten von sich preis. Ausserdem enthalten die Karten und Routen Informationen und Daten, genauso wie die Wetterfunktion. Um welche Daten es sich hierbei handelt soll die folgende Tabelle darstellen:

Funktion	Daten
Benutzer	Benutzer ID Vorname, Nachname E-Mail Bild Passwort
Karte	Geodaten (Longitude, Latitude) Standort Anfangsposition Zielposition Umgebungsdaten Zeit Reitstil Bericht
Wetter	Ort Temperatur Niederschlag Windgeschwindigkeit
Nachrichten	NachrichtenID Textnachricht

Tabelle .1: Daten innerhalb der Anwendung

1.1 XML-Dateien

Da eine Android-Applikation entwickelt werden soll, ist die Repräsentation der Daten im Zusammenhang mit dem Repräsentationsformat XML die beste Möglichkeit für dieses Projekt, da das Format in der Entwicklung bereits involviert ist und in dem Modul 'Web-basierte Anwendungen 2' praktisch umgesetzt wurde. Aus diesem Grund soll die Strukturierung der Daten anhand von XML-Schemata dargestellt und erläutert werden. Hierfür werden zunächst die XML-Dateien präsentiert werden. Im weiteren Verlauf werden zu den XML-Dateien zugehörigen XML-Schemata dargestellt und erläutert werden.

Die Sprache XML wurde entwickelt, um Daten zu beschreiben. Bevor die erstellten XML-Dateien dargestellt und erläutert werden, soll kurz auf die Wetterfunktion eingegangen werden. Die XML-Datei, die für die Wetterrepräsentation benötigt und entnommen wird kann unter der Referenz (2) aufgerufen werden. Bei der vorgegebenen URL handelt es sich dabei um die Wetterdaten für die Stadt Frechen. Da die XML-Datei bereits vorgegeben ist und abgerufen werden kann, wird nicht mehr weiter darauf eingegangen.

```
<?xml version="1.0" encoding="UTF-8"?>
<Routen>
    <Route>
        <Id>123</Id>
        <Verlauf>
            <Geodaten>
                <Startposition Längengrad="50.916672" Breitengrad="6.81667"/>
                <Zielposition Längengrad="50.916672" Breitengrad="6.81667"/>
            </Geodaten>
        </Verlauf>
        <Streckenlänge Einheit="Km" Länge="3.4"/>
        <Dauer Einheit="h" Stunden="2"/>
        <Bericht>
            <Startpunkt>Frechen</Startpunkt>
            <Endpunkt>Gummersbach</Endpunkt>
            <Beschreibung>Es ist eine schöne Strecke zum reiten..</Beschreibung>
            <Gefahren>Achtung ein Baum auf dem Waldweg eingestürzt..</Gefahren>
            <Datum>01.06.2015</Datum>
        </Bericht>
    </Route>
    <Route>
        <Id>124</Id>
        <Verlauf>
            <Geodaten>
                <Startposition Längengrad="50.954672" Breitengrad="6.81623"/>
                <Zielposition Längengrad="50.4326672" Breitengrad="6.84367"/>
            </Geodaten>
        </Verlauf>
        <Streckenlänge Einheit="Km" Länge="3.7"/>
        <Dauer Einheit="h" Stunden="2"/>
        <Bericht>
            <Startpunkt>Köln</Startpunkt>
            <Endpunkt>Gummersbach</Endpunkt>
            <Beschreibung>Super Strecke. Endlich mal etwas Neues.</Beschreibung>
            <Gefahren>Keine Gefahren unterwegs erkannt.</Gefahren>
            <Datum>03.06.2015</Datum>
        </Bericht>
    </Route>
</Routen>
```

Abbildung .1: XML-Datei 'Route'

Die XML-Datei der Route ist am umfangreichsten, weshalb diese beschrieben und erläutert werden soll. Die XML-Datei zur Nachricht und zu Profil wird nicht weiter erläutert, da es einen ähnlichen einfacheren Aufbau enthält.

Die Route enthält eine Menge von Daten. Aus der Abbildung 1 kann entnommen werden, dass das root-Element 'Routen' ist. Die Route hat ein Kindelemente und zwar 'Route'. Das Element 'Route' hat weitere Kindelemente wie 'ID', 'Geodaten', 'Streckenlänge', 'Dauer' und 'Bericht'. Die ID soll die Route eindeutig identifizieren. Die 'Geodaten' unterteilen sich in weitere Elemente: 'Startposition' und 'Zielposition'. Diese beiden enthalten die Attribute 'Längengrad' und 'Breitengrad' für die Positionierung auf der Karte. Da der aktuelle Standort beim Öffnen der Karte automatisch angezeigt wird, sind diese Angaben Pflicht. 'Bericht' unterteilt sich nochmal in weitere Kindelemente. Diese soll Auskunft über die Route geben. Das heißt, es werden Informationen zum 'Startpunkt' und 'Endpunkt' der Route und Gefahren erfasst. Der Bericht muss einen Datum enthalten, damit die Benutzer sehen, ob es sich um einen aktuellen Bericht handelt.

```
<?xml version="1.0" encoding="utf-8"?>
<Nachrichten>
  <Unterhaltung>
    <RoutenID>123</RoutenID>
    <NachrichtenID>234</NachrichtenID>
    <Teilnehmer1>Sabine S.</Teilnehmer1>
    <Teilnehmer2>Marius Müller</Teilnehmer2>
    <Thema>Köln-Gummersbach über ...</Thema>
    <Nachricht>
      <Absender>Sabine S.</Absender>
      <Empfänger>Marius Müller</Empfänger>
      <Zeitstempel>01.06.2015</Zeitstempel>
      <Inhalt>Hallo liebe Hanne...</Inhalt>
    </Nachricht>
  </Unterhaltung>
  <Unterhaltung>
    <RoutenID>121</RoutenID>
    <NachrichtenID>233</NachrichtenID>
    <Teilnehmer1>Hanne B.</Teilnehmer1>
    <Teilnehmer2>Guido M.</Teilnehmer2>
    <Thema>Deine Route...</Thema>
    <Nachricht>
      <Absender>Hanne B.</Absender>
      <Empfänger>Guido M.</Empfänger>
      <Zeitstempel>04.06.2015</Zeitstempel>
      <Inhalt>Hallo Hanne, wie geht es dir. Ich habe eine Frage bezüglich der neuen Route die du erstellt hast</Inhalt>
    </Nachricht>
  </Unterhaltung>
</Nachrichten>
```

Abbildung .2: XML-Datei 'Nachricht'

```
<?xml version="1.0" encoding="utf-8"?>
<Profil>
  <Benutzer>
    <Vorname>Josef</Vorname>
    <Nachname>Groß</Nachname>
    <EMail>JosefGroß@JosefGroß.de</EMail>
    <Status>Profi-Reiter</Status>
  </Benutzer>
  <Benutzer>
    <Vorname>Marie</Vorname>
    <Nachname>Brings</Nachname>
    <EMail>MarieBrings@MarieBrings.de</EMail>
    <Status>Amateur-Reiter</Status>
  </Benutzer>
</Profil>
```

Abbildung .3: XML-Datei 'Profil'

1.2 XML-Schemata

Anhand der XML-Schemata ist die Strukturierung der Daten nochmals deutlich zu erkennen. Hier kann man die einzelnen Elemente und die zugehörigen Attribute entnehmen. So sind die Attribute Längengrad und Breitengrad beispielsweise vom Typ float. Wichtig ist auch, dass z.B die Anzahl der Routen durch **maxOccurs** und **minOccurs** festgelegt sind. Das Element 'Route' kann garnicht oder mehrmals auftreten.

```
<xss:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xss="http://www.w3.org/2001/XMLSchema">
  <xss:element name="Routen">
    <xss:complexType>
      <xss:sequence>
        <xss:element name="Route" maxOccurs="unbounded" minOccurs="0">
          <xss:complexType>
            <xss:sequence>
              <xss:element type="xs:int" name="Id"/>
              <xss:element name="Name" default="">
                <xss:complexType>
                  <xss:sequence>
                    <xss:element name="Geodaten">
                      <xss:complexType>
                        <xss:sequence>
                          <xss:element name="StartPosition">
                            <xss:complexType>
                              <xss:simpleContent>
                                <xss:extension base="xs:string">
                                  <xss:attribute type="xs:float" name="Längengrad"/>
                                  <xss:attribute type="xs:float" name="Breitengrad"/>
                                </xss:extension>
                              </xss:simpleContent>
                            </xss:complexType>
                          </xss:element>
                          <xss:element name="Zielposition">
                            <xss:complexType>
                              <xss:simpleContent>
                                <xss:extension base="xs:string">
                                  <xss:attribute type="xs:float" name="Längengrad"/>
                                  <xss:attribute type="xs:float" name="Breitengrad"/>
                                </xss:extension>
                              </xss:simpleContent>
                            </xss:complexType>
                          </xss:element>
                        </xss:sequence>
                      </xss:complexType>
                    </xss:element>
                  </xss:sequence>
                </xss:complexType>
              </xss:element>
            </xss:sequence>
          </xss:complexType>
        </xss:element>
      </xss:sequence>
    </xss:complexType>
  </xss:element>
</xss:schema>
```

```

</xs:element>
<xs:element name="Streckenlänge">
  <xs:complexType>
    <xs:simpleType>
      <xs:extension base="xs:string">
        <xs:attribute type="xs:string" name="Einheit"/>
        <xs:attribute type="xs:float" name="Länge"/>
      </xs:extension>
    </xs:simpleType>
  </xs:complexType>
<xs:element name="Dauer">
  <xs:complexType>
    <xs:simpleType>
      <xs:extension base="xs:string">
        <xs:attribute type="xs:string" name="Einheit"/>
        <xs:attribute type="xs:int" name="Stunden"/>
      </xs:extension>
    </xs:simpleType>
  </xs:complexType>
<xs:element name="Bericht">
  <xs:complexType>
    <xs:sequence>
      <xs:element type="xs:string" name="Startpunkt"/>
      <xs:element type="xs:string" name="Endpunkt"/>
      <xs:element type="xs:string" name="Beschreibung"/>
      <xs:element type="xs:string" name="Gefahren"/>
      <xs:element type="xs:string" name="Datum"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:element>
</xs:schema>

```

Abbildung .4: XML-Schema 'Route'

```

<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="Nachrichten">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Unterhaltung" maxOccurs="unbounded" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element type="xs:int" name="RoutenID"/>
            <xs:element type="xs:int" name="NachrichtenID"/>
            <xs:element type="xs:string" name="Teilnehmer1"/>
            <xs:element type="xs:string" name="Teilnehmer2"/>
            <xs:element type="xs:string" name="Thema"/>
            <xs:element name="Nachricht">
              <xs:complexType>
                <xs:sequence>
                  <xs:element type="xs:string" name="Absender"/>
                  <xs:element type="xs:string" name="Empfänger"/>
                  <xs:element type="xs:string" name="Zeitstempel"/>
                  <xs:element type="xs:string" name="Inhalt"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>

```

Abbildung .5: XML-Schema 'Nachricht'

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Profil">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Benutzer" maxOccurs="unbounded" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element type="xs:string" name="Vorname"/>
              <xs:element type="xs:string" name="Nachname"/>
              <xs:element type="xs:string" name="EMail"/>
              <xs:element type="xs:string" name="Status"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Abbildung .6: XML-Schema 'Profil'

1.3 Entity-Relationship-Modellierung

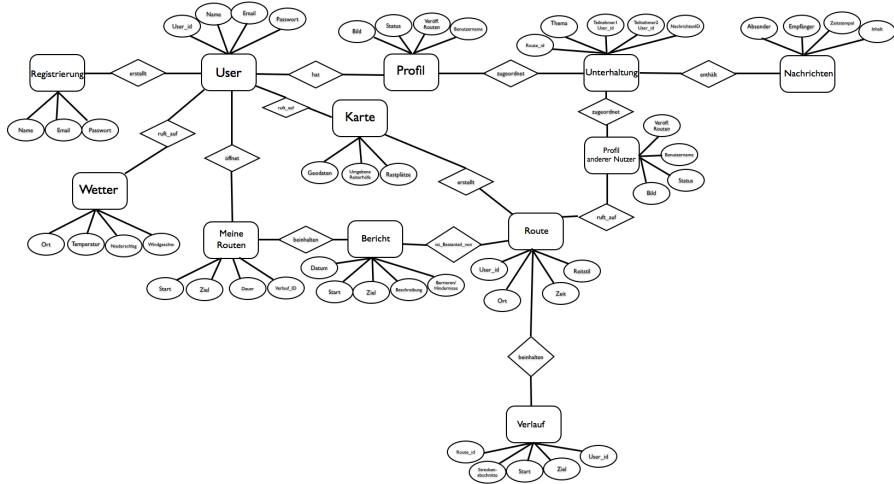


Abbildung .7: ER-Modellierung zur Datenstruktur

Die gesamte Datenmodellierung der Applikation soll durch ein Entity-Relationship-Diagramm verdeutlicht werden. Die Entität 'User' hat die Attribute 'User id', 'Name', 'E-Mail', 'Passwort' und kann die Entität 'Karte' aufrufen, die die Attribute 'Geodaten', 'Umgebene Reiterhöfe' und 'Rastplätze' enthält.

Über die Karte können Routen erstellt werden. Die erstellten Routen enthalten wiederum Berichte und einen Verlauf zur jeder Route.

Der Benutzer kann bereits erstellte Routen in 'Meine Routen' aufrufen, die ebenfalls die Berichte enthalten. 'Meine Routen'.

Jeder registrierte User enthält ein eigenes Profil. Um den Nachrichtenaustausch darstellen zu können, mussten zwei Entitäten zu den User-Profilen erstellt werden: 'Profil' und 'Profil anderer Nutzer'.

Der Nachrichtenaustausch zweier Benutzer folgt über die Entität 'Unterhaltung', welche den Profilen zugeordnet ist. Jede Unterhaltung besitzt eine 'Route id'. Über diese kann der User eine Unterhaltung eindeutig einer Route zuordnen. Jede Unterhaltung wird über ein Thema beschrieben. Des weiteren beinhaltet die Entität 'Unterhaltung', die User id von beiden Teilnehmern und eine 'Nachrichten id'. Über diese werden die ausgetauschten Nachrichten zugeordnet. Außerdem wird die Thematik einer Unterhaltung unter dem Attribut 'Thema' kurz beschrieben.

2 WBA-Modellierung

2.1 Ressourcen

Die im folgenden tabellarisch festgelegten Ressourcen wurden aus dem in Meilenstein 2 angefertigten Kommunikationsmodell abgeleitet. Die Ressourcen sind Repräsentationen aus der realen Umwelt der Benutzer. Die Struktur zur Darstellung der Ressourcen wurde vom Buch 'Rest und HTTP' von Stefan Tilkov übernommen (1).

2.1.1 User

2.1.2 Profil

Die Profil Ressource enthält alle Profilangaben eines Benutzers. Dabei kann der Webservice ein Profil erstellen, ändern, erweitern und löschen. Das Profil beinhaltet Texte, ein Profilbild und veröffentlichte Routen. Über das Profil kann jeder Benutzer Nachrichten an andere Benutzer des Systems erfassen und senden. Jedes mal wenn der Benutzer eine Route erstellt, erhält er anliegende Routen von anderen Benutzern. Somit kann er über die Verlinkung auf der Karte die Profile aufrufen und in Kontakt treten, um weitere Informationen zu erhalten oder Erfahrungen auszutauschen.

Methoden: GET,PUT,POST,DELETE

Attribute:

- Benutzername
- Status
- Profilbild
- Veröffentlichte Routen

Einsatz: Erstellung eines Profils, Ändern von Angaben, Löschen des Accounts

2.1.3 Karte

Die Ressource Karte erlaubt das Anzeigen und das Aufrufen von Standorten auf einer Karte. Der Webservice kann über diese Anzeige, Routen erstellen. Auf der Karte erkennt der Benutzer die Umgebung und kann sich anhand dieser orientieren. Die auf der Karte angezeigten Icons beinhalten Informationen und können über das Display angetippt werden. Sie erhalten Texte oder die Benutzer können sich an dem Punkt navigieren lassen.

Methoden: GET,

Attribute:

- Geodaten
- Umgebungsinformationen
- Rastplätze

Einsatz: Anzeige der aktuellen Position auf einer Karte, Navigation zu einem bestimmten Punkt, Abrufen von Informationen zur Umgebung, Umgebung auf einer Karte anzeigen lassen

2.1.4 Nachricht

Die Ressource Nachricht repräsentiert die einzelnen Mitteilungen die der Webservice an die Benutzer schickt, diese sind in Textformat abgefasst. Die Nachrichten sind Bestandteile einer Unterhaltung. Möchte ein Benutzer einem anderen Benutzer etwas mitteilen, kann er dies über das Profil. Eine Nachricht kann abgerufen, erstellt und gelöscht werden.

Methoden: GET,POST,DELETE

Attribute:

- Sender
- Absender
- Inhalt
- Uhrzeit

Einsatz: Nachricht Lesen, Schreiben, Löschen

2.1.5 Wetter

Über die Ressource Wetter stellt der Webservice Informationen zum Wetter zur Verfügung. Dabei werden diese über den Query-Parameter gefiltert. Über den Webservice kann zu jedem Ort, die Wetterdaten angezeigt werden. Des weiteren kann die Vorhersage auch im 3-Stunden-Takt angezeigt werden.

Methoden: GET

Einsatz: Abrufen von Wetterdaten, Filter nach Temperatur, Niederschlag und

Windgeschwindigkeit, Land, Bewölkung

Query-Parameter: Temperatur=[Celsius], Niederschlag=[Prozent], Windgeschwindigkeit=[Beschreibung], Land=[Land],Bewölkung=[Beschreibung]

2.1.6 Route

Die Ressource Route repräsentiert die Planung der Strecke, die ein Reiter mit seinem Pferd vornehmen möchte. Die Ressource beinhaltet die Startposition, die Zielposition, die Streckenlänge und die Dauer des Ritts. Über die Route werden die User angezeigt, die diese Route geteilt haben. Wird über den Webservice eine neue Route erstellt, kann diese mit anderen Benutzern geteilt oder abgespeichert werden.

Über den Webservice kann eine Route erstellt, geändert und gelöscht werden.

Methoden: GET,POST,DELETE

Attribute:

- Zielposition
- Reitstil
- Zeit
- User_id

Einsatz: Route erstellen, ändern und löschen

2.1.7 Berichte

Die Ressource Bericht repräsentiert die Feedback-Funktion, die der Benutzer zu der erstellten Route gibt. Die Benutzer können unter Anderem einen Bericht über die Vegetation oder über Hindernisse erfassen. Die Ressource beinhaltet einen Start- und Zielpunkt, die der Benutzer manuell eingibt. Zusätzlich enthält die Ressource eine allgemeine Beschreibung und eine Beschreibung zu Barrieren beziehungsweise Gefahren und einen Datum zur Identifizierung der Aktualität des Berichts.

Methoden: GET,POST

Attribute:

- Startpunkt
- Zielpunkt

- Beschreibung
- Barrieren/Gefahren
- Datum

Einsatz: Bericht erstellen, Bericht aufrufen

2.2 Ressourcentabelle

In der folgenden Tabelle werden die ermittelten Ressourcen mit der angewendeten Methode und der festgelegten URI dargestellt.

Ressource	Methode	Semantik
User/	GET	Wird nicht genutzt, da keine Verwendung besteht
User/	PUT	Wird nicht genutzt, da keine Verwendung besteht
User/	POST	Der Webservice legt ein neuen User an
User/	DELETE	Wird nicht genutzt, da keine Verwendung besteht
Karte/	GET	Der Webservice ruft über die Wetter Api die Open-Street-Karte auf
Karte/	PUT	Wird nicht genutzt, da keine Verwendung besteht
Karte/	POST	Wird nicht genutzt, da keine Verwendung besteht
Karte/	DELETE	Wird nicht genutzt, da keine Verwendung besteht
Karte/Route/{:ID}	GET	Der Webservice ruft eine bestimmte Route auf
Karte/Route/{:ID}	PUT	Wird nicht genutzt, da keine Verwendung besteht
Karte/Route/{:ID}	POST	Der Webservice erstellt eine Route
Karte/Route/{:ID}	DELETE	Der Webservice löscht eine bestimmte Route
Route{:ID}/Profil/{:ID}	GET	Der Webservice ruft ein bestehendes Profil auf
Route{:ID}/Profil/{:ID}	PUT	Wird nicht genutzt, da keine Verwendung besteht
Route{:ID}/Profil/{:ID}	POST	Wird nicht genutzt, da keine Verwendung besteht
Route{:ID}/Profil/{:ID}	DELETE	Wird nicht genutzt, da keine Verwendung besteht
User{:ID}/Profil/{:ID}	GET	Wird nicht genutzt, da keine Verwendung besteht
User{:ID}/Profil/{:ID}	PUT	Der Webservice aktualisiert ein Profil

User{ID}/Profil/{ID}	POST	Wird nicht genutzt, da keine Verwendung besteht
User{ID}/Profil/{ID}	DELETE	Der Webservice löscht ein Profil
Wetter/Ort{ID}/heute/	GET	Der Webservice ruft für heute die Wetterdaten zur gewünschten Ortschaft auf
Wetter/Ort{ID}/heute/	PUT	Wird nicht genutzt, da keine Verwendung besteht
Wetter/Ort{ID}/heute/	POST	Wird nicht genutzt, da keine Verwendung besteht
Wetter/Ort{ID}/heute/	DELETE	Wird nicht genutzt, da keine Verwendung besteht
Wetter/Ort{ID}/heute/3-Stunden-Takt	GET	Der Webservice ruft Wetterdaten für den aktuellen Tag zur gewünschten Ortschaft im drei Stunden Takt auf
Wetter/Ort{ID}/heute/3-Stunden-Takt	PUT	Wird nicht genutzt, da keine Verwendung besteht
Wetter/Ort{ID}/heute/3-Stunden-Takt	POST	Wird nicht genutzt, da keine Verwendung besteht
Wetter/Ort{ID}/heute/3-Stunden-Takt	DELETE	Wird nicht genutzt, da keine Verwendung besteht
Profil/{ID}/Unterhaltung/	GET	Alle Unterhaltungen werden aufgelistet
Profil/{ID}/Unterhaltung/	PUT	Wird nicht genutzt, da keine Verwendung besteht
Profil/{ID}/Unterhaltung/	POST	Wird nicht genutzt, da keine Verwendung besteht
Profil/{ID}/Unterhaltung/	DELETE	Wird nicht genutzt, da keine Verwendung besteht
Profil/{ID}/Unterhaltung/Nachricht/{ID}	GET	Der Webservice ruft eine bestimmte Nachricht auf
Profil/{ID}/Unterhaltung/Nachricht/{ID}	PUT	Wird nicht genutzt, da keine Verwendung besteht
Profil/{ID}/Unterhaltung/Nachricht/	POST	Der Webservice erstellt eine Nachricht
Profil/{ID}/Unterhaltung/Nachricht/{ID}	DELETE	Der Webservice löscht eine Nachricht
User/{ID}/Nachrichtenliste	GET	Der Webservice ruft alle vorhandenen Nachrichten auf

Es werden noch weitere Ressourcen wie zum Beispiel Bodenbeschaffenheit, Straßen, Wege, Rastplätze, Trinkwasserstellen, Wälder, Weiden und Flüsse darge-

User/{:ID}/Nachrichtenliste	PUT	Wird nicht genutzt, da keine Verwendung besteht
User/{:ID}/Nachrichtenliste	POST	Wird nicht genutzt, da keine Verwendung besteht
User/{:ID}/Nachrichtenliste	DELETE	Wird nicht genutzt, da keine Verwendung besteht
User/{:ID}/Routenliste	GET	Der Webservice ruft alle seine gespeicherten Routen in einer Liste auf
User/{:ID}/Routenliste	PUT	Wird nicht genutzt, da keine Verwendung besteht
User/{:ID}/Routenliste	POST	Wird nicht genutzt, da keine Verwendung besteht
User/{:ID}/Routenliste	DELETE	Wird nicht genutzt, da keine Verwendung besteht
Karte/Route/{:ID}/Bericht	GET	Der Webservice ruft einen Bericht zur Route auf
Karte/Route/{:ID}/Bericht	PUT	Der Webservice ändert oder aktualisiert den Bericht zur Route
Karte/Route/{:ID}/Bericht	POST	Der Webservice erstellt einen Bericht zur Route
Karte/Route/{:ID}/Bericht	DELETE	Wird nicht genutzt, da keine Verwendung besteht

Tabelle .2: Ressourcen Übersicht

stellt. Diese werden über die OpenStreetMap(4) Schnittstelle genutzt. Um diese Ressourcen darzustellen werden diese explizit von dem Datenbestand der OpenStreetMap gefiltert.

2.3 Pseudocode

```

Funktion RouteErmitteln(Graph, Startknoten):
    \Initialisierung
        für jedes v in Graph:
            distanz[v]:= unendlich
            vorgänger[v]:= nicht definiert
            distanz[startpunkt]:= 0
            Q:= Menge aller Knoten
            Zeit:= 0
            Geschwindigkeit:= 0
            Entfernung:= Zeit*Geschwindigkeit

            gefährlicheBodenbeschaffenheit[]:= {Asphalt, Kies}
            gefährlicheBarrieren[]:= {Autobahn, Fußgängerwege, verbotene Straßen, Lebensraum gefährlicher Tiere}
            gefährlicheVegetation[]:= {Adonisröschen, Eisenhut, Goldregen, Gundaman, Raps}
            gefährlichesWetter[]:= {Hagel, Eis}

            mittelBodenbeschaffenheit[]:= {Sand, Stein, Schlamm}
            mittelBarrieren[]:= {umgefallener Baum}
            mittelVegetation[]:= {Korkenzieherweide}
            mittelWetter[]:= {Schnee, Regen}

            besteBodenbeschaffenheit[]:= {Wiese, Erde}
            besteBarrieren[]:= {Wald, Wiese, Brücke, Bach}
            besteVegetation[]:= {Weide, Birke}
            besteWetter[]:= {Sonnig, Bewölkt}

    \main-Schleife
        solange Q nicht leer ist:
            u:= Knoten in Q mit kleinstner distanz[]
            entferne u von Q
            für jeden nachbar v von u:
                \Die Distanzen aller Nachbarknoten vom Startknoten werden berechnet, durch Addition von dessen Distanz mit den Kantengewichten
                alternativ:= distanz[u]+distanz_zwischen[u,v]

                \Wenn die berechnete Distanz für einen Knoten kleiner ist als die aktuelle, die Kante die unangemessenen Bodenbeschaffenheiten besitzt, keine
                gefährlichen Barrieren aufweist, die gefährlichsten Vegetation ausschließt und die gefährlichsten Wetterverhältnisse enthält, wird nochmal abgefragt, ob
                eine Kante existiert, die die besten Bodenbeschaffenheiten, die ungefährlichsten oder keine Barrieren aufweist, die ungefährlichsten Vegetationen oder
                keine enthält und die besten Wetterverhältnisse enthält, wird wird die Distanz aktualisiert und setzt den aktuellen Knoten als Vorgänger. Ansonsten
                werden die Kanten gewählt, die die mittelBodenbeschaffenheit oder mittelBarrieren oder mittelVegetation oder mittelWetter aufweisen.

                wenn alternativ < distanz[v] && !gefährlicheBodenbeschaffenheit && !gefährlicheBarrieren && !gefährlicheVegetation
                    wenn besteBodenbeschaffenheit && (Barrieren || keine Barrieren) && (Vegetation || keineVegetation) && Wetter
                        distanz[v]:= alternativ
                        vorgänger[v]:= u
                    sonst mittelBodenbeschaffenheit || mittelBarrieren || mittelVegetation || mittelWetter
                        distanz[v]:= alternativ
                        vorgänger[v]:= u

                \Wenn die aktuelle Distanz die erwünschte Entfernung überschreitet wird die Route an der Position abgebrochen und die Route wird erstellt
                wenn vorgänger[] > Entfernung
                    brich die Schleife ab
                return vorgänger[]

```

Abbildung .8: Systemkomponente Server / Pseudocode zur Ermittlung von Routen

Bei dem in Abbildung 8 dargestellten Pseudocode, handelt es sich um den Pseudocode des Dijkstra Algorithmus(5). Der Dijkstra Algorithmus sucht nach dem kürzesten Weg. Die angezeigten Routen müssen die Kriterien der Benutzereingaben erfüllen, weshalb der Algorithmus um einige Zeilen ergänzt wurde. Der Algorithmus kann in zwei Abschnitte unterteilt werden. Der erste Teil kümmert sich um die Initialisierung und der zweite Teil um die Schleife, die die Route ermittelt.

Zunächst werden alle Knoten ermittelt.

Durch die vom Benutzer eingegeben Kriterien, Reitstil und Zeit wird die Länge der Strecke ermittelt, die der Reiter erreichen kann unter diesen Kriterien. Die Geschwindigkeit wird anhand dem vom Benutzer ausgewähltem Reitstil erfasst. Dabei werden Geschwindigkeitsangaben für den jeweiligen Reitstil übermittelt. Der Reiter gibt an, wie lange er reiten möchten. Diese Angabe wird bei der Berechnung der Entfernung genutzt. Die Entfernung ist das Produkt von Zeit und Geschwindigkeit. Dabei sagt die Entfernung aus, wie weit der Reiter eine Route ausführen kann. Die Route wird dann anhand dieser Entfernung ermittelt. Die Main-Schleife ist zuständig für die Ermittlung der Route. Die Schleife prüft alle Knoten, bis Q leer ist oder die Abbruchbedingung am Ende der Schleife erfüllt ist. Die Distanz des Startknotens wird als permanent markiert und die restlichen Distanzen als temporär. Anschließend werden die Distanzen von den Nachbarknoten des Starknotens durch die Addition von dessen Distanzen mit den Kantengewichten ermittelt. Die Distanz wird aktualisiert (der Vorgänger wird ersetzt), wenn die Distanz eines anderen Knotens kleiner ist, als die des aktuellen Knotens. Zu Letzt wird geprüft, ob die ermittelte Distanz die Entfernung überschreitet. Wenn ja, wird die Schleife abgebrochen und die Route wird erstellt oder die Schleife wird ein weiteres Mal durchgelaufen.

Wichtig ist, dass in dem Pseudocode nicht dargestellt werden konnte, dass dem Benutzer die besten 3 Routen vorgeschlagen werden. Dies sollte aber festgehalten werden.

Um die Routenplanung an die Reiter anzupassen, werden bei der Berechnung der Strecken, vorab die Merkmale für die Kanten festgelegt. Die über die OpenStreetMap erhaltenen Straßendaten, werden genutzt, um die Kanten anhand der Namen zu identifizieren. Außerdem werden weitere Vergleichskriterien einer Straße über die OpenStreetMap übermittelt. Diese sind die Bodenbeschaffenheit, die Vegetation und die Barrieren. Zur Das Wetter wird nicht als Kriterium zur Routenplanung mit einbezogen. Da davon ausgegangen wird, dass die Reiter bei schlechtem Wetter keine Routen planen. Die Reiter erhalten über die Wetter-Funktion die Möglichkeit vorausschauend darüber zu entscheiden, ob Sie die Routen heute starten wollen oder nicht. Die Merkmale zur Bestimmung der Kanten sehen wie folgt aus:

Bodenbeschaffenheit

Bei der Gewichtung der Bodenbeschaffenheit wird darauf geachtet, dass der Boden eine gesunde Grundlage für den Pferd bietet. Es sollten für das Pferd mögliche gesundheitlichen Schäden ausgeschlossen werden. Dabei werden alle Böden an ihrer Angemessenheit eingestuft. Die Gewichtung beruht auf Grundlage von dem Wissen der realen Benutzer aus Meilenstein 3.

1 - sehr angemessen bis 5 - unangemessen

Boden	Angemessenheit
Asphalt	4
Kies	4
Sand	3
Stein	3
Schlamm	2
Wiese	1
Erde	1

Tabelle .3: Priorisierung der Bodenbeschaffenheit

Barrieren

Die Barrieren werden daran gewichtetet, wie gefährlich sie für Tier und Mensch sein können. Sie müssen frühzeitig erkannt werden um die Reiter in keine Gefahr für andere und für sich und sein Pferd zu bringen.

1 - nicht gefährlich bis 5 - sehr gefährlich

Barriere	Gefahr
verbotene Straßen	5
Autobahn	5
Fußgängerweg	5
Lebensraum gefährlicher Tiere	4
umgefallener Baum	2
Wald	1
Wiese	1
Brücke	1
Bach	1

Tabelle .4: Priorisierung der Barrieren

Vegetation

Bei der Vegetation werden die Pflanzen berücksichtigt, die für das Pferd nicht giftig bis zu stark giftig Pflanzen sein können. Die giftigen Pflanzen können zu starken gesundheitlichen Problemen oder sogar zu Lebensgefahr führen. Die Liste der aufgelisteten Pflanzen ist nicht vollständig. Es werden lediglich einige Pflanzen erwähnt, um den Pseudocode zu vervollständigen. Die Gefahrenstufung kann von folgender Webseite entnommen werden: (?)

1 nicht gefährlich - bis 5 - sehr gefährlich

Pflanze	Gefahr
Adonisröschen	5
Eisenhut	5
Goldregen	5
Gundaman	5
Raps	5
Aronstab	4
Bohne	4
Korkenzieherweide	2
Weide	1
Birke	1

Tabelle .5: Priorisierung der Vegetation

Wetter

Das Wetter spielt eine wichtige Rolle beim Reiten auswärts. Die Priorisierung soll die Gefahrenstufung darstellen.

1 nicht gefährlich - bis **5** - sehr gefährlich

Wetter	Gefahr
Eis	5
Hagel	5
Schnee	3
Regen	3
Sonnig	1
Bewölkt	1

Tabelle .6: Priorisierung des Wetters

Die Kanten, die mit einer 4 und 5 eingestuften Gefahr oder Bodenbeschaffenheit sollen in dem Suchalgorithmus sofort ausscheiden. Diejenigen, die mit einer 1 bewertet wurden, sollen die erste Wahl darstellen. Die Kanten, die mit einer 2 und 3 bewertet wurden werden als zweite Wahl genommen, falls keine Kante vorhanden ist, die eine 1 aufweist.

Funktion aktuelle PositionErmittler

\Initialisierung

\ Über die GPS-Schnittstelle werden die Koordinaten anhand von Längen- und Breitengrad bestimmt

```
position_koordinate_x= lat;
position_koordinate_y= lon;
new_position_koordinate_x= lat;
new_position_koordinate_y= lon;
```

\ Sobald der Reiter seine Position wechselt werden die neuen bestimmten Koordinaten mit den davor bekannten Koordinaten verglichen

```
if position_koordinate_x != new_position_koordinate_x
  && position_koordinate_y != new_position_koordinate_y
```

\ Stimmen die Koordinaten nicht überein werden die zuvor bekannten Koordinaten durch die neuen Koordinaten ersetzt. Dieser Vorgang wiederholt sich bei jeder Änderung und der Reiter kann auf der Karte erkennen wo er sich im Moment befindet.

dann ersetze,

```
position_koordinate_x = new_position_koordinate_x
position_koordinate_y = new_position_koordinate_y
```

Anzeige_auf_der_Karte();

Abbildung .9: Systemkomponente Client/ Pseudocode zur Ermittlung der aktuellen Position

3 Prototyp UI

3.1 Gestaltungslösungen

In Meilenstein 2 wurde festgelegt, dass das Paperbased-Prototyping-Verfahren zum Ermitteln der Gestaltungslösung verwendet werden soll. Die Gründe beruhen auf der Vielfältigkeit des Verfahrens. Es erfordert keinerlei Kosten und ist für allgemeine Änderungen und für Änderungen des Interfaces sehr geeignet, da es sich nur um Zeichnungen auf Papier handelt. Dadurch sind mögliche Iterationen schneller und einfacher umsetzbar.

3.1.1 Design Principles

Bevor der Prototyp erstellt wird, sollen die aus der Norm ISO 9241 Teil 110 erfassten Grundsätze der Gestaltung interaktiver Systeme berücksichtigt werden.

Aufgabenangemessenheit

Die Aufgabenangemessenheit ist von enormer Bedeutung, denn die Funktionalitäten des Systems müssen den Benutzer bei seiner Aufgabe mit dem System unterstützen, sodass er seine Aufgabe effektiv und effizient erledigen kann. Hierfür soll das User Interface so einfach wie möglich gestaltet werden, um den Benutzer nicht mit unnötigen Informationen zu belasten. So soll beispielsweise die Erstellung einer Route in mehrere Schritte unterteilt werden, damit nicht zu viele Informationen als Ganzes erscheinen.

Selbstbeschreibungsfähigkeit

Der angefertigte Prototyp muss selbstbeschreibungsfähig sein. Dem Benutzer muss eine eindeutige Führung durch das System gewährleistet werden, um an das gewünschten Ziel zu gelangen. Damit der Benutzer intuitiv zum nächsten Schritt gelangt, sollten die Buttons aussagekräftig gestaltet werden. Sofern ein Benutzer eine Funktionalität wie "Wetter abrufen" aufruft, muss dem Benutzer erkennbar sein wie er zur nächsten Seite gelangt. Um die eingesetzten Formulare selbstbeschreibungsfähig zu gestalten, sollen innerhalb der Eingabefelder Beispiel eingaben stehen. So würde beispielsweise bei der Wetterfunktion in dem Eingabefeld für den Ort 'Ort eingeben' stehen.

Konformität mit Benutzererwartungen

Um den Erwartungen der Benutzer gerecht zu werden und die Konformität, die erwünscht ist, bieten zu können, muss das System entsprechend der Benutzererfahrungen gestaltet werden. Die Routenplanung sollte so gestaltet werden, dass die Benutzer ein bekanntes Muster wiedererkennen können. Dadurch soll die Konzentration auf den Inhalt der Karte gelenkt werden.

Lernförderlichkeit

Um die Lernförderlichkeit des Systems bestmöglich umsetzen zu können, sollte ein logischer Aufbau der einzelnen User Interfaces umgesetzt werden. Es sollte nicht zu viel vom Benutzer erwartet werden, was die Interaktion mit dem System betrifft. Das Hauptmenü sollte die einzelnen Elemente des Systems deutlich unterscheiden. Die Bezeichnungen der Funktionen wie z.B der Buttons, ist deswegen sehr wichtig.

Steuerbarkeit

Um bei der Gestaltung die Steuerbarkeit des Systems zu gewährleisten, soll dem Benutzer Abbruchbedienungen angeboten werden. Der Benutzer kann beispielsweise die gestartete Route jederzeit abbrechen, indem er 'Route beenden' tätigt. Außerdem soll der Benutzer jederzeit die Möglichkeit haben, über die "Zurück-Taste" zum vorherigen Schritt zu gelangen. Ziel ist es hierbei, dem Benutzer das Gefühl zu geben, Kontrolle über das System zu haben und es zu steuern.

Fehlertoleranz

Die Fehlertoleranz ist in dem Bereich der Eingabe der Orte bei der Routenplanung und bei der Wetterfunktion ein wichtiger Aspekt, welches umgesetzt werden muss. Es besteht die Gefahr, dass der Benutzer die Orte nicht richtig erfasst. Trotz der falschen Schreibweise sollte das System in der Lage sein, den richtigen Ort oder die Wetterdaten zum richtigen Ort anzuzeigen.

Individualisierbarkeit

Die Individualisierbarkeit soll innerhalb der Funktion "Route erstellen" realisiert werden. Dem Benutzer soll die Möglichkeit geboten werden, die Routen ihren eigenen Fähigkeiten und Bedürfnissen anzupassen. Die Karten können individuell vergrößert beziehungsweise verkleinert werden. Dies soll eine Zoom-Funktion ermöglichen. Außerdem ist das Teilen der Routen für jeden Benutzer selbst gelassen. Das heißtt, es ist nicht zwingend notwendig eine Route zu teilen. Der Benutzer kann die Route auch lediglich speichern.

3.1.2 Paperbased-Prototyping

Der Prototyp baut zum größten Teil auf den Resultaten der Anforderungsanalyse. Es wurde versucht die funktionalen, organisatorischen und qualitativen Anforderungen bestmöglich umzusetzen. Im Folgenden werden die erstellten paperbasierten Prototypen in Reihenfolge dargestellt und erläutert. Zudem sollte durch eine kurze und knappe Beschreibung klar werden, welche Abbildung welche Anforderungen erfüllt. Um die Begründung nachvollziehen zu können, sollten die Anforderungen aus Meilenstein 3 vorliegen.

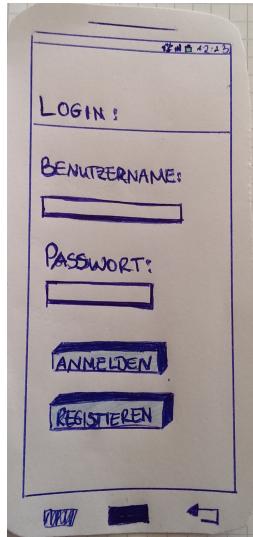


Abbildung .10: Paperbased Prototyp - Login

Ist der Benutzer bereits registriert, kann er durch die Eingabe von Benutzernamen und Passwort in das System gelangen und sich somit einloggen. Ist der Benutzer nicht registriert, so muss er zunächst eine Registrierung vollziehen. Das Ziel hierbei ist es eine schnelle Registrierung bzw. Login zu gewährleisten. In kurzen Schritten kann der Benutzer erfolgreich mit der Nutzung des Systems beginnen.



Abbildung .11: Paperbased Prototyp - Registrierung

Die Registrierung erfolgt durch die Eingabe von Benutzername, Passwort und E-Mail. Über den Benutzernamen kann der User von anderen User angesprochen werden.

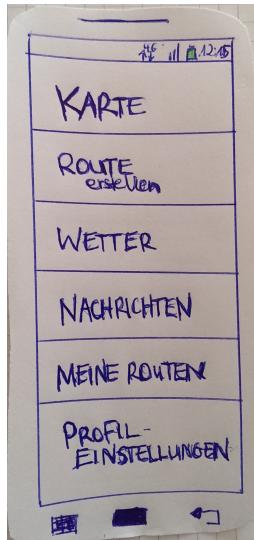


Abbildung .12: Paperbased Prototyp - Hauptmenü

Das Hauptmenü besteht aus 6 Buttons: Karte, Route erstellen, Wetter, Nachrichten, Meine Routen, Profil-Einstellungen. Diese Button stehen für die einzelne Funktionen. Über diese soll der Benutzer zu seinem Nutzungsziel kom-

men. Q30 erfordert eine einfache und ersichtliche Benutzeroberfläche. Aus dem Grund wurde das User-Interface recht simpel gehalten. Q50 wiederum erfordert die einfache Darstellung der Informationen. Die Buttons sollten deswegen recht einfache und aussagekräftige Titel haben. Eine schnelle und effektive Bedienung (Q70) soll durch die einfache Struktur und der Größe der Buttons gewährleistet werden.

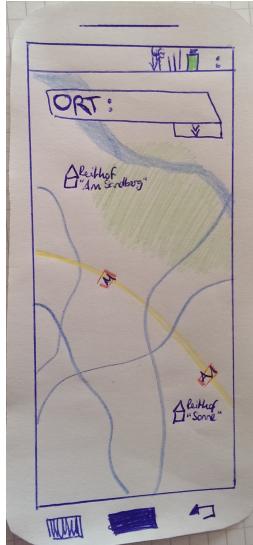


Abbildung .13: Paperbased Prototyp - Karte

Sobald der Benutzer den Button 'Karte' tätigt, öffnet sich die Karte mit den Umgebungsinformationen und dem aktuellen Standort. Zusätzlich kann er andere Orte erforschen, indem er den erwünschten Ort in das Eingabefeld eingibt. Die funktionale Anforderung F10 erfordert die Darstellung der Umgebungsinformationen auf der Karte. Dies wird in diesem Prototyp erfolgreich dargestellt. Das Erfassen des Standorts wird ermöglicht, welches die Anforderung O20 erfordert.

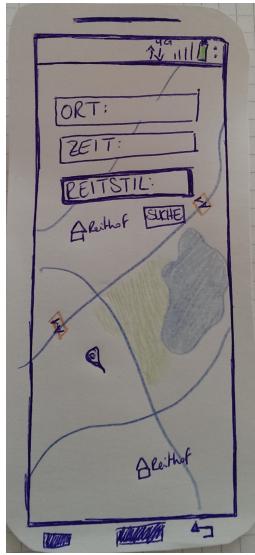


Abbildung .14: Paperbased Prototyp - Route erstellen 1

Tätigt der Benutzer den Button 'Route erstellen', gelangt er ebenfalls auf die Karte. Dort kann er die erwünschte Reitzeit angeben und den Reitstil. Muss der Benutzer eine Route erstellen werden ihm dabei mehrere Kriterien angeboten um die Route dem Bedürfnissen anzupassen. Durch diesen Prototyp wird die funktionale Anforderung F10 (Route erstellen) umgesetzt. Außerdem können werden Umgebungsinformationen dargestellt (F20).

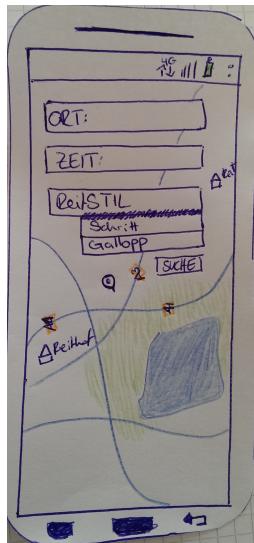


Abbildung .15: Paperbased Prototyp - Route erstellen 2

Durch Tätigkeiten des Reitstils öffnen sich Optionen, die sich in 'Sprint' und 'Gallopp' unterscheiden. Jeder Reitstil enthält konstante Mittel-Geschwindigkeiten (3), die als Kriterien zur Suche der Routen genutzt werden. Durch den Button 'Suche' wird nach der Besten Alternativ-Route gesucht. Der Benutzer kann eine Strecke öfter geritten sein dabei sollte ihm eine Alternative zur Verfügung stehen um neue Strecken zu entdecken.

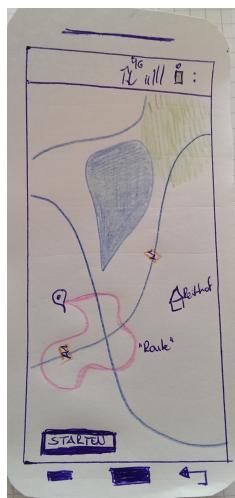


Abbildung .16: Paperbased Prototyp - Route erstellen 3

Die Route wird gefunden und angezeigt. Durch den Button 'Starten' kann die Route gestartet werden. Hier sollten dem Benutzer auch Alternativ-Routen angeboten werden, welches ein Erfordernis der funktionalen Anforderung F80 ist.

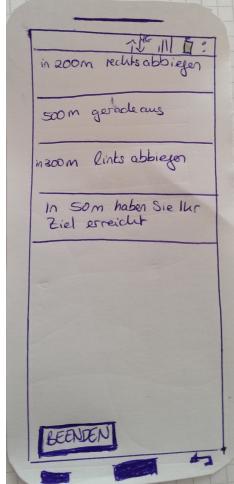


Abbildung .17: Paperbased Prototyp - Route erstellen 4

Die Route wird schriftlich navigiert und gleichzeitig auditiv wiedergegeben. Durch den Button 'Beenden' wird die Route beendet. Während des Reitens hat der Reiter nicht die Möglichkeit sein Smartphone in der Hand zu halten und ständig aufs Display zu schauen. Aus diesem Grund müssen auditive Navigationen zu Verfügung stehen, um dem Reiter ständig zu begleiten und zu warnen. Dadurch wäre die funktionale Anforderung F60 ermöglicht. Durch die Navigation wird dem Benutzer keinerlei Vorkenntnisse zur Ortschaft vorausgesetzt, welches die Anforderung O70 erfordert. Q40 beinhaltet eine einfache und deutliche Sprachführung. Sowohl das textuelle, als auch das auditive Navigieren sollte dementsprechend angepasst werden.

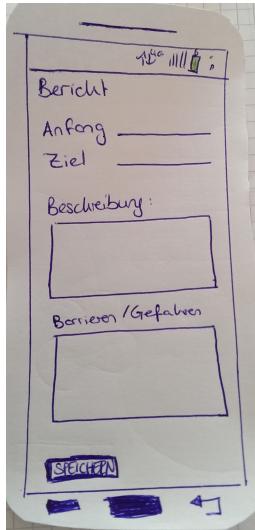


Abbildung .18: Paperbased Prototyp - Route erstellen 5

Nach Beenden der Route erscheint ein Fenster zum Erfassen eines Berichts. Dies ist für die Benutzer optional. Das bedeutet, dass der Benutzer nicht verpflichtet ist eine Eingabe durchzuführen. Handelt es sich um keine neue Route, sondern um eine, die ein anderer Benutzer erstellt und geteilt hat, wird der Benutzer, der die Route erstellt hat über neue Berichte benachrichtigt. Dies umfasst die Realisierung und Umsetzung der Anforderung F100.



Abbildung .19: Paperbased Prototyp - Wetter 1

Tägt der Benutzer im Hauptmenü den Button 'Wetter', so erscheint dieses Fenster. Durch Eingabe des Ortes und anschließend durch das Tätigen des 'Wetter'-Buttons werden die Werte für den erwünschten Ort angezeigt. Die Wetter-Anzeige (F40) muss für den Benutzer unter geeigneten Kriterien angezeigt werden um vorausschauende Planung zu erstellen. Dadurch gelingt es dem Benutzer die Routenplanung unabhängig vom Wetter zu ermöglichen, welches die Anforderung O50 erfordert.

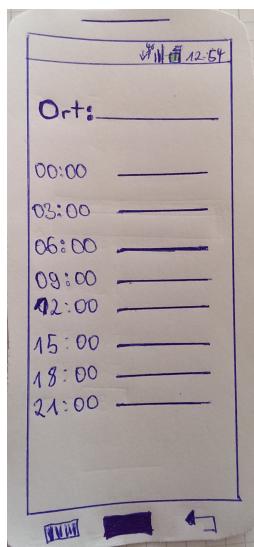


Abbildung .20: Paperbased Prototyp - Wetter 2

Hier wird die Temperatur und der Niederschlag im 3-Stunden Takt angezeigt.

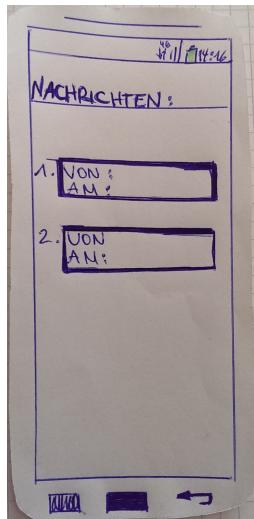


Abbildung .21: Paperbased Prototyp - Nachrichten 1

Tätigt der Benutzer den 'Nachrichten'-Button, so erscheint das Nachrichten-Fenster. Hier kann der Benutzer auf seine Nachrichten zugreifen und die neuesten Nachrichten lesen. Die Benutzer können sich hier untereinander über die Domäne und Routen spezifische Themen austauschen (F30).

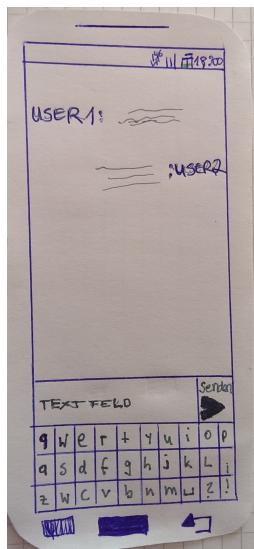


Abbildung .22: Paperbased Prototyp - Nachrichten 2

Abbildung 16 stellt die Kommunikation durch Nachrichtenaustausch zwischen zwei unterschiedlichen Benutzern dar. Dadurch wird die Anforderung F30 die Kommunikation zu anderen Nutzern erfüllt.



Abbildung .23: Paperbased Prototyp - Meine Routen

Durch den 'Meine Routen'-Button im Hauptmenü, werden dem Benutzer die bereits erstellten Routen angezeigt. Der Benutzer kann seine Routen abspeichern und sie für eigene Performance-Zwecke oder gar zur Erinnerung festhalten. Hier kann der Benutzer seine eigenen Routen auch abändern, indem er die Route öffnet, so wie es die Anforderung F50 erfordert.

Folgende Prototypen sind während der Iteration entstanden, die nach diesem Abschnitt durchgeführt wurde.

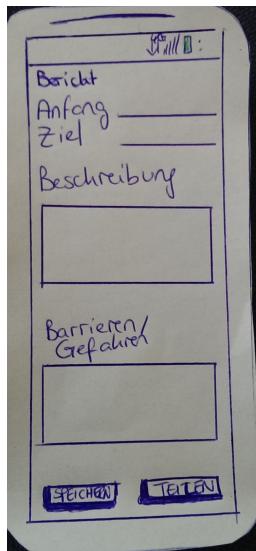


Abbildung .24: Paperbased Prototyp - Route erstellen 5

Durch die Berichterstattung seitens der Benutzer, soll ein Feedback über die Route entstehen, wovon andere Benutzer profitieren sollen. Hierbei ist es von großer Bedeutung, dass die Routen veröffentlicht werden können.



Abbildung .25: Paperbased Prototyp - Meine Routen

Hier bietet sich die Möglichkeit für die Benutzer, einen unkomplizierten Zugriff auf die eigenen Routen zu haben. Optional kann die Route, die noch nicht geteilt wurde, veröffentlicht werden, so dass andere Benutzer die Route sehen.

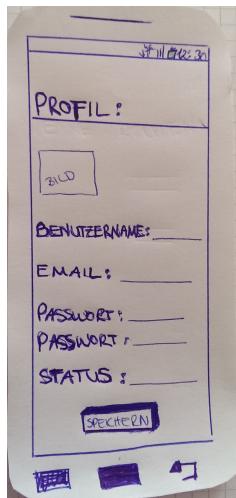


Abbildung .26: Paperbased Prototyp - Profileinstellungen Benutzer Ansicht

Durch das Tätigen des 'Profil-Einstellungen'- Buttons gelangt der Benutzer in die Einstellungen und kann dort sein Profil-Foto, Benutzername, Passwort und sein Status ändern und speichern. Die angegebenen Daten werden dabei vertraulich und vor Missbrauch geschützt (O10). Der Benutzer selbst hat das alleinige Zugriffsrecht auf sein Profil.

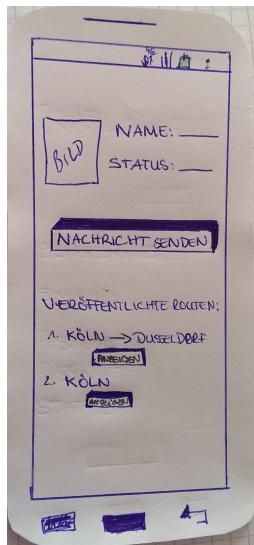


Abbildung .27: Paperbased Prototyp - Profilansicht von anderen Benutzern

Dies ist die Sicht eines Benutzers, der das Profil eines anderen Benutzers aufruft. Durch den Button 'Nachricht senden' kann der Benutzer eine Nachricht senden (siehe Abbildung 16). Zusätzlich sieht er erstellten Routen des Benutzers, die er auf der Karte abrufen kann.

3.2 Fazit zu der Gestaltungslösung

Die Gestaltungslösung erfüllt die Anforderungen, die in der Anforderungsanalyse festgelegt wurden. Jedoch sind einige funktionalen User-Interfaces entstanden, die zwingend notwendig sind, aber keine Anforderungen aufweisen. Um die Anforderungsanalyse zu vervollständigen und die fehlenden Anforderungen zu identifizieren wurde der gesamte Meilenstein 3 iteriert. Zur Ermittlung der Anforderungen wurde ein neues präskriptives Szenario erfasst, woraus die fehlenden Funktionalitäten ermittelt wurden.

4 Projektplan

Datum / KW	Aktivität	1. Unteraktivität	2. Unteraktivität	Workload geplant / p.P.	Workload gesamt	Workload tatsächlich	
					Derya Ergue	Sinem Kaya	
14	Exposé	Ideenfindung	Brainstorming	3	3	3	
	Dokumentaufbau	Layout / Struktur	Latex	3	0	4	
	Projektplan			2	6	1,5	
13.04.2015	Milestein 1	Nutzungsproblem		1	1	1	
		Zielsetzung		1	1	1	
		Verteilte Anwendungslogik		1	1	1	
		Wirtschaftliche und gesellschaftliche Relevanz		1	1	1	
27.04.2015	Milestein 2	Zielhierarchie	Strategische Ziele	1	2	2	
			Taktische Ziele	1	2	2	
		related-works	Operative Ziele	1	1	1,5	
			ReiterApp	1	0	1	
			Cavallo-Retcoach	1	1	0	
			sonstige	1	0	1	
		Alleneinstellungsmerkmale		1	1	1	
		Methodischer Rahmen (MCI)		15	11	9	
		Kommunikationsmodell		5	8	3	
		Risiken		3	2	2	
11.05.2015	Milestein 3	Spezifikation der POCs		1	2	3	
		Architekturdigramm/ Architekurbegründung	Architekturdigramm	3	2	3	
			Architekurbegründung	3	0	5	
					MS2 gesamt IST:	32	33,5
01.06.2015	Milestein 4	Datenstrukturen	XML-Schemata	4	0	7	
			ER-Diagramm	2	4	0	
		WBA-Modellierung		10	7	0	
		Prototypen UI		10	11	10	
					MS4 gesamt IST:	22	17
15.06.2015	Milestein 5	funktionale Prototypen	Programmierung gesamt	150	40	66	
		Evaluationsergebnisse UI		10			
		narratives Konzept für filmische Präsentation		10			
29.06.2015	Milestein 6	Prozessassessment		10			
		Fazit		4			
		Installationsdokumentation		5			
					MS6 gesamt IST:	0	0
			Insgesamt Soll	200	Insgesamt Ist:	132	130,5

Abbildung .28: Projektplan zu Meilenstein 3 und 4

Abbildungsverzeichnis

.1	XML-Datei 'Route'	3
.2	XML-Datei 'Nachricht'	4
.3	XML-Datei 'Profil'	5
.4	XML-Schema 'Route'	6
.5	XML-Schema 'Nachricht'	6
.6	XML-Schema 'Profil'	7
.7	ER-Modellierung zur Datenstruktur	8
.8	Systemkomponente Server/ Pseudocode zur Ermittlung von Routen	16
.9	Systemkomponente Client/ Pseudocode zur Ermittlung der aktuellen Position	20
.10	Paperbased Prototyp - Login	23
.11	Paperbased Prototyp - Registrierung	24
.12	Paperbased Prototyp - Hauptmenü	24
.13	Paperbased Prototyp - Karte	25
.14	Paperbased Prototyp - Route erstellen 1	26
.15	Paperbased Prototyp - Route erstellen 2	27
.16	Paperbased Prototyp - Route erstellen 3	27
.17	Paperbased Prototyp - Route erstellen 4	28
.18	Paperbased Prototyp - Route erstellen 5	29
.19	Paperbased Prototyp - Wetter 1	29
.20	Paperbased Prototyp - Wetter 2	30
.21	Paperbased Prototyp - Nachrichten 1	31
.22	Paperbased Prototyp - Nachrichten 2	31
.23	Paperbased Prototyp - Meine Routen	32
.24	Paperbased Prototyp - Route erstellen 5	33
.25	Paperbased Prototyp - Meine Routen	34
.26	Paperbased Prototyp - Profileinstellungen Benutzer Ansicht . . .	34
.27	Paperbased Prototyp - Profilansicht von anderen Benutzern . . .	35
.28	Projektplan zu Meilenstein 3 und 4	37
.29	Paperbased Prototyp - Route erstellen 5	43
.30	Paperbased Prototyp - Meine Routen	44
.31	Paperbased Prototyp - Profileinstellungen Benutzer Ansicht . . .	44
.32	Paperbased Prototyp - Profilansicht von anderen Benutzern . . .	45

Tabellenverzeichnis

.1	Daten innerhalb der Anwendung	2
.2	Ressourcen Übersicht	15
.3	Priorisierung der Bodenbeschaffenheit	18
.4	Priorisierung der Barrieren	18
.5	Priorisierung der Vegetation	19
.6	Priorisierung des Wetters	19
.7	Glossar für die Anforderungen	41

5 Literaturverzeichnis

- [1] Autor: Stefan Tilkov Titel: REST und HTTP Verlag:dpunkt.verlag 2009 ISBN: 978-3-89864-732-8 2., aktualisierte und erweiterte Auflage - Sichtungsdatum: 18.05.2015
- [2] <http://api.openweathermap.org/data/2.5/forecast/?q=frechen&mode=xml&APPID=fbc548f0ad7993a7b9a589366a0a84fb> - Sichtungsdatum: 25.05.2015
- [3] http://equivetinfo.de/html/eckdaten_pferd.html - Sichtungsdatum: 25.05.2015
- [4] http://wiki.openstreetmap.org/wiki/DE:OSMC_Reitkarte - Sichtungsdatum: 08.06.2015
- [5] http://www.gitta.info/Accessibiliti/de/html/Dijkstra_learningObject1.html - Sichtungsdatum: 21.06.2015
- [6] <http://www.botanikus.de/Botanik3/Tiere/Pferde/pferde.html> - Sichtungsdatum: 25.06.2015

6 Anhang

6.1 Iteration

6.1.1 Szenarien

Szenario 6: Anke möchte neue Leute kennenlernen

Anke hatte letzte Woche ihre ersten Reitstunden und möchte heute neue Leute kennenlernen, die sie eventuell begleiten möchten. Anke ist begeistert vom Reitsport und hatte einen tollen Einstieg in die Reitwelt. Das einzige was ihr fehlt sind neue Bekanntschaften mit Reit-Anfängern wie sie selbst. Jedoch ist es für sie wichtig, dass es sich hierbei ausschließlich um Mädchen/Frauen handelt, da sie einen Freund hat, der sehr eifersüchtig ist. Um diese Situation zu ändern ladet sich Tina die Anwendung auf ihrem Smartphone runter, die sie von einer Freundin letzte Woche erfahren hat. Sie öffnet die Anwendung und es erscheint prompt ein Eingabefeld zum Einloggen und falls noch nicht registriert wurde, ein Button der die Benutzer zur Registrierung weiterleitet. Anke registriert sich und beginnt direkt mit dem Suchen nach neuen Freunden. Sie sucht nach Benutzern, die das selbe Interesse teilen. Es ist sehr wichtig für sie, dass die Benutzer, die sie kennenlernen möchte, auch Reit-Anfänger sind. Sie erforscht einige Benutzerprofile und wird sehr schnell fündig, da der Status den Benutzer kategorisiert. Sie landet aus Zufall auf Zeyneps Profil und bemerkt, dass sie weiblich ist, da sie ein Profilfoto hochgeladen hat. Anke freut sich sehr, da sie bemerkt, dass viele Reit-Anfänger vorhanden sind. Einigen dieser Benutzer, die in der Nähe wohnen, schreibt Anke eine Nachricht, um einen Kontakt zu knüpfen. Als sie die Nachricht versendet und auf das Profil eines Benutzers genauer hinschaut, entdeckt sie, dass ein Reit-Anfänger bereits eine Route geteilt hat. Sie öffnet die Route und liest sich den Bericht zu der Route durch. Anke ist begeistert und motiviert sich umso mehr, sehr bald auch eine eigene Route zu erstellen und diese mit weiteren Benutzern zu teilen, da die Benutzer sich damit gegenseitig informieren können.

Analyse des Szenarios:

Der Reiter möchte neue Leute kennenlernen und dabei unter gewissen Voraussetzungen in Kontakt treten. Der Reiter möchte eigene Routen erstellen und die gesammelten Informationen in Form von Feedback mit anderen Benutzern teilen. Der Reiter möchte den Erfahrungsgrad eines anderen Reiters erkennen.

Claim Analyse:

- +Erhalt von Informationen zu Personen
- +Kategorisierung der Personen über den Erfahrungsgrad
- +Orientierung an den Erfahrungsberichten von anderen Benutzer
- +geschlossenes System durch Registrierung

Begriff	Erklärung
Status	Der Status repräsentiert die reale Reiterfahrung. Dabei wird der Status in 3 Untertitel gegliedert: Professionell, Amateur und Anfänger

Tabelle .7: Glossar für die Anforderungen

Erweiterung der Anforderung anhand des Szenarios 6:

6.1.2 Funktionale Anforderungen

- **F110 Identifikation der Nutzer**

Das System muss jedem Nutzer eine möglich reale Repräsentation einer Person bereitstellen.

- **F120 Entdecken neuer Routen**

Das System muss dem Nutzer die Möglichkeit bieten neue Routen zu entdecken.

- **F130 Speicherung der Routen**

Das System muss dem Nutzer die Speicherung und das abrufen von eigenen Routen gewährleisten.

- **F140 Feedback zu Routen**

Das System muss dem Nutzer eine Möglichkeit anbieten sich zu Routen zu äußern und diese innerhalb der Anwendung sichtbar für andere Nutzer ist die die selbe Route reiten.

- **F150 Kategorisierung der Reiter**

Das System muss dem Nutzer einen "Status" bereitstellen anhand der Nutzer sich in die Domäne kategorisieren lassen kann.

- **F160 Wettervorhersage**

Das System muss dem Nutzer eine Wettervorhersage anbieten.

6.1.3 Organisatorische Anforderungen

- **O80 Registrierung**

Das System darf nur über eine Registrierung den Zugriff in das System gewährleisten.

- **O90 Teilen von Informationen**

Das System darf Informationen von Nutzer innerhalb des System anderen Nutzern zu Verfügung stellen.

- **O90 Teilen von Benutzerinformationen**

Das System soll dem Benutzer die Möglichkeit bieten, eigene Benutzerinformationen mit anderen Benutzern zu teilen.

6.1.4 Qualitative Anforderungen

- **Q80 persistente Speicherung**

Das System darf keine Daten mehrfach abspeichern.

- **Q90 Speicherung**

Das System soll die Daten der Benutzer speichern.

- **Q70 Zugriffsrechte**

Das System darf keine Ansichten oder Veränderungen an den Informationen unbefugten dritten Personen gestatten.

6.1.5 Prototyp

Nun soll der Prototyp vorgestellt werden, der die Anforderungen, die während der Iteration entstanden sind, erfüllt und mit der die Evaluierung in Meilenstein 5 fortgesetzt werden soll. Die Prototypen, die sich nicht geändert haben, werden nicht nochmal dargestellt. Es werden lediglich die Prototypen gezeigt, an denen eine Veränderung erkennbar ist und die aufgrund der neuen Anforderungen neu entstanden sind.

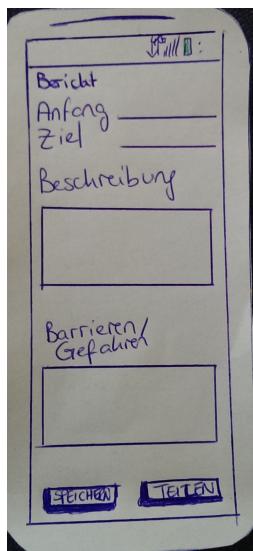


Abbildung .29: Paperbased Prototyp - Route erstellen 5

Durch die Berichterstattung seitens der Benutzer, soll ein Feedback über die Route entstehen, wovon andere Benutzer profitieren sollen. Hierbei ist es von großer Bedeutung, dass die Routen veröffentlicht werden können.



Abbildung .30: Paperbased Prototyp - Meine Routen

Hier bietet sich die Möglichkeit für die Benutzer, einen unkomplizierten Zugriff auf die eigenen Routen zu haben. Optional kann die Route, die noch nicht geteilt wurde, veröffentlicht werden, so dass andere Benutzer die Route sehen.

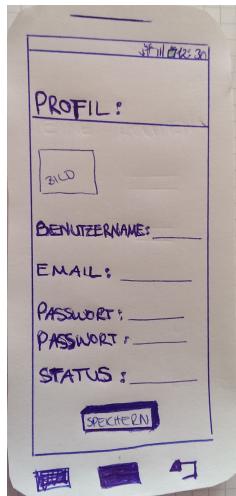


Abbildung .31: Paperbased Prototyp - Profileinstellungen Benutzer Ansicht

Durch das Tätigen des 'Profil-Einstellungen'- Buttons gelangt der Benutzer in die Einstellungen und kann dort sein Profil-Foto, Benutzername, Passwort und sein Status ändern und speichern. Die angegebenen Daten werden dabei vertraulich und vor Missbrauch geschützt (O10). Der Benutzer selbst hat das alleinige Zugriffsrecht auf sein Profil.

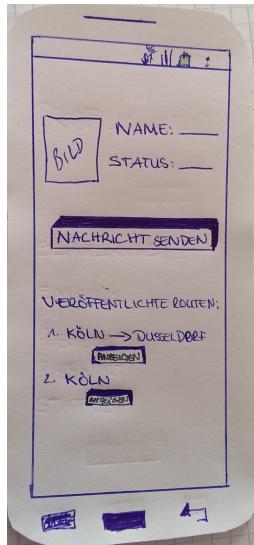


Abbildung .32: Paperbased Prototyp - Profilansicht von anderen Benutzern

Dies ist die Sicht eines Benutzers, der das Profil eines anderen Benutzers aufruft. Durch den Button 'Nachricht senden' kann der Benutzer eine Nachricht senden (siehe Abbildung 16). Zusätzlich sieht er erstellten Routen des Benutzers, die er auf der Karte abrufen kann.