



## Diplomarbeit

# GeoQuest

Imst, 15. Mai 2017

Eingereicht von

Julia Tiefenbrunner Verantwortlich für IT: HTML, CSS, BWL: Kaufvertrag  
Laura Huber Verantwortlich für IT: HTML, CSS, BWL: Kaufvertrag

Eingereicht bei

Dominik Neuner

# **Eidesstattliche Erklärung**

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbst verfasst und keine anderen als die angeführten Behelfe verwendet habe. Alle Stellen, die wörtlich oder inhaltlich den angegebenen Quellen entnommen wurden, sind als solche kenntlich gemacht. Ich bin damit einverstanden, dass meine Arbeit öffentlich zugänglich gemacht wird.

---

Ort, Datum

---

Julia Tiefenbrunner

---

Laura Huber

# **Abnahmeerklärung**

Hiermit bestätigt der Auftraggeber, dass das übergebene Produkt dieser Diplomarbeit den dokumentierten Vorgaben entspricht. Des Weiteren verzichtet der Auftraggeber auf unentgeltliche Wartung und Weiterentwicklung des Produktes durch die Projektmitglieder bzw. die Schule.

---

Ort, Datum

---

Auftraggeber

# **Vorwort**

z. B. Hinweise, wie das bearbeitete Thema gefunden wurde oder Dank für die Betreuung (Kooperationspartner/in, Betreuer/innen, Sponsoren) etc.

# **Abstract (Deutsch)**

(ca. ½ bis max. 2 Seiten) Kurzbeschreibung von Aufgabenstellung und Problemlösung.

# **Abstract (Englisch)**

(ca. ½ bis max. 2 Seiten)

# Inhaltsverzeichnis

<b>1. Projektmanagement</b>	<b>12</b>
1.1. Metainformationen . . . . .	12
1.1.1. Projekt . . . . .	12
1.1.2. Team . . . . .	12
1.1.3. Betreuer . . . . .	12
1.1.4. Partner . . . . .	13
1.1.5. Ansprechpartner . . . . .	13
1.2. Vorerhebungen . . . . .	13
1.2.1. Projektzieleplan . . . . .	13
1.2.2. Projektumfeld . . . . .	15
1.2.3. Risikoanalyse . . . . .	17
1.3. Pflichtenheft . . . . .	18
1.3.1. Zielbestimmung . . . . .	18
1.3.2. Produkteinsatz und Umgebung . . . . .	19
1.3.3. Funktionalitäten . . . . .	19
1.3.4. Testszenarien und Testfälle . . . . .	20
1.3.5. Liefervereinbarung . . . . .	21
1.4. Planung . . . . .	21
1.4.1. Projektstrukturplan . . . . .	21
1.4.2. Meilensteine . . . . .	22
1.4.3. Gant-Chart . . . . .	22
1.4.4. Abnahmekriterien . . . . .	22
1.4.5. Pläne zur Evaluierung . . . . .	22
1.4.6. Ergänzungen und zu klärende Punkte . . . . .	22

<b>2. Vorstellung des Produktes</b>	<b>23</b>
<b>3. Eingesetzte Technologien</b>	<b>24</b>
<b>4. Problemanalyse</b>	<b>26</b>
4.1. USE-Case-Analyse . . . . .	26
4.2. Domain-Class-Modelling . . . . .	27
4.3. User-Interface-Design . . . . .	28
<b>5. Systementwurf</b>	<b>29</b>
5.1. Architektur . . . . .	29
5.1.1. Design der Komponenten . . . . .	29
5.1.2. Benutzerschnittstellen . . . . .	30
5.1.3. Datenhaltungskonzept . . . . .	30
5.1.4. Konzept für Ausnahmebehandlung . . . . .	30
5.1.5. Sicherheitskonzept . . . . .	30
5.1.6. Design der Testumgebung . . . . .	31
5.1.7. Design der Ausführungsumgebung . . . . .	31
5.2. Detailentwurf . . . . .	31
<b>6. Implementierung</b>	<b>33</b>
<b>7. Deployment</b>	<b>34</b>
<b>8. Tests</b>	<b>35</b>
8.1. Systemtests . . . . .	35
8.2. Akzeptanztests . . . . .	35
<b>9. Projektevaluation</b>	<b>36</b>
<b>10. Benutzerhandbuch</b>	<b>37</b>
<b>11. Betriebswirtschaftlicher Kontext</b>	<b>38</b>
<b>12. Zusammenfassung</b>	<b>39</b>

<b>13. Beispielkapitel</b>	<b>40</b>
13.1. Beispiele zitieren . . . . .	40
13.1.1. Beispiele Abbildungen . . . . .	41
13.2. Beispiele Listen . . . . .	42
13.3. Beispiel Codesequenz . . . . .	44
13.3.1. Quicksort in JAVA . . . . .	44
13.4. Beispieltext . . . . .	46
<b>Abbildungsverzeichnis</b>	<b>49</b>
<b>Tabellenverzeichnis</b>	<b>50</b>
<b>Quelltexte</b>	<b>51</b>
<b>Literaturverzeichnis</b>	<b>52</b>
<b>A. Anhang-Kapitel</b>	<b>54</b>
A.1. Anhang-Section . . . . .	54

# **Einleitende Bemerkungen**

# **Notationen**

Beschreibung wie Code, Hinweise, Zitate etc. formatiert werden

# **1. Projektmanagement**

## **1.1. Metainformationen**

### **1.1.1. Projekt**

GeoQuest

Es soll eine Applikation entwickelt werden, bei der Fragen (zu einem Thema) und zugehörigen möglichen Antworten (mit einer richtigen) erstellt werden können. Jeder Frage muss ein Standort auf der Karte zugewiesen werden. Im Front-End können Fragen im Umkreis aufgelistet und gelöst werden. Für jede gelöste Frage erhöht sich der Punktestand des Benutzers.

### **1.1.2. Team**

Julia Tiefenbrunner und Laura Huber

### **1.1.3. Betreuer**

Neuner Dominik, MSc

#### **1.1.4. Partner**

Tourismusverband Imst BHAK Imst

#### **1.1.5. Ansprechpartner**

### **1.2. Vorerhebungen**

#### **1.2.1. Projektzieleplan**

Projektziele-Hierarchie:

Oberziel: App/Spiel GeoQuest im Appstore für jeden zum Download bereit

SMART-Prinzipien:

Spezifisch

Unser Ziel ist es, eine App für den Tourismusverband Imst einzurichten. Diese können Gäste nutzen um dort ganz leicht nachzusehen welche Sehenswürdigkeiten, Attraktionen und Freizeitaktivitäten die Ferienregion Imst bietet. Unser großes Ziel wird in folgende kleine Ziele eingeteilt:

- App programmieren, strukturieren und designen
- Alle Sehenswürdigkeiten, Attraktionen und Freizeitmöglichkeiten sammeln und Informationen darüber herausfinden
- informative Texte zu jeder Sehenswürdigkeit verfassen
- Bilder und Videos der Sehenswürdigkeiten produzieren

## *GeoQuest*

### Messbar

Die App soll nicht mehr als 20000€ kosten, inklusive den Gehältern der Programmierer. Die Bilder müssen Skalierbar sein, verschwommene Bilder und Videos werden nicht genutzt. Bilder und Videos müssen schnell geladen werden. Fotoapparate, Kameras, Schnittprogramme und Photoshop dürfen 12000€ nicht überschreiten. Für Gehälter und Löhne der anderen Mitarbeiter ist ein Budget von 20000€ vorgesehen. Das geplante Budget für das Projekt sind 70000€.

### Akzeptiert/Attraktiv

Jedes Ziel wird im Team abgesprochen, sollte jemand ein Ziel nicht akzeptieren wird diskutiert und um Verbesserungsvorschläge gebeten bis alle zufrieden sind. Das Team besteht aus einem Fotograf (zuständig für Bilder), einem Videoproduzenten (zuständig für Videos), einer Sekretärin (Texte), zwei Programmierer (App Entwicklung) und einer Webdesignerin (zuständig für CSS). Die App wird für Nutzer (Gäste) sehr angenehm werden, da sie einfach nachsehen können welche Sehenswürdigkeiten geöffnet (z. B. Museen) haben und was sie alles wo ansehen können.

### Realistisch

Das gesamte Projekt geht 6 Monate. Die einzelnen Ziele gehen jedoch nicht so lange. Zuerst werden alle Sehenswürdigkeiten, Attraktionen und Freizeitmöglichkeiten angesammelt und Informationen darüber gesucht. Danach wird die App programmiert, eingerichtet, strukturiert und designt, zeitgleich werden Texte verfasst und natürlich auf die App übertragen. Zu guter Letzt werden Fotos geschossen, bearbeitet und Kurzfilme produziert.

### Terminierbar

- Alle Sehenswürdigkeiten, Attraktionen und Freizeitmöglichkeiten sammeln und Informationen darüber herausfinden – 2 Wochen
- App programmieren, strukturieren und designen und informative Texte zu jeder Sehenswürdigkeit verfassen und auf die App übertragen – 12

Wochen

- Bilder und Videos der Sehenswürdigkeiten produzieren – 6 Wochen

### 1.2.2. Projektumfeld

- Projektumfeldanalyse:

Identifikation der Stakeholder:

Auftraggeber: Tourismusverband Imst

Kunden

Projektbetreuer: Dominik Neuner

Mitarbeiter: Laura Huber, Julia Tiefenbrunner

Partner: BHAK Imst

Charakterisierung der Stakeholder:

Stakeholder	Einfluss	Nähe	Einstellungen	Beschreibung
Tourismusverband	groß	mittel	positiv	Auftraggeber
Kunden	gering	nahe	neutral	„Konsumenten“ des Produktes
Projektbetreuer	groß	nahe	positiv	Hilft bei Problemen
Mitarbeiter	groß	nahe	positiv	Erstellen App
HAK	gering	fern	neutral	Stellt Räumlichkeiten zur Verfügung

Maßnahmen:

Tourismusverband:

- Aufgabenstellung vorgeben
- Wünsche, Erwartungen
- Kontrolle des Projekts

Kunden:

- Feedback geben
- Verbesserungsvorschläge

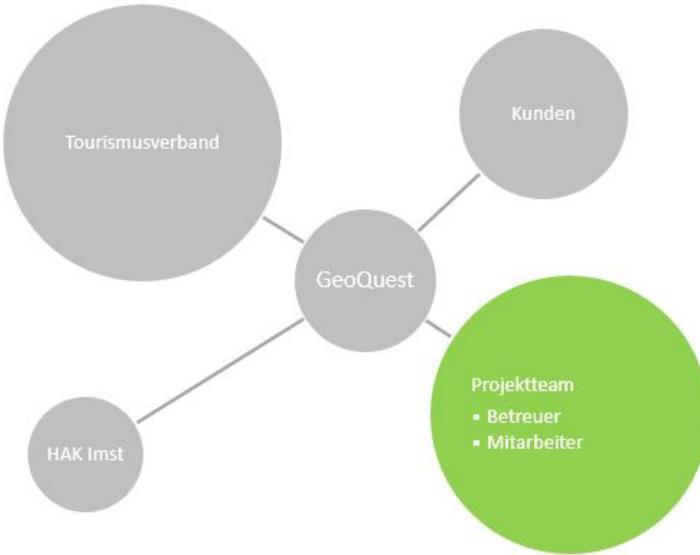
Mitarbeiter:

- Erfüllung der Aufgaben
- Teamgeist bilden
- Respektvoller Umgang
- Zuverlässigkeit

HAK:

- Raumbelegung prüfen
- Räumlichkeiten bereitstellen

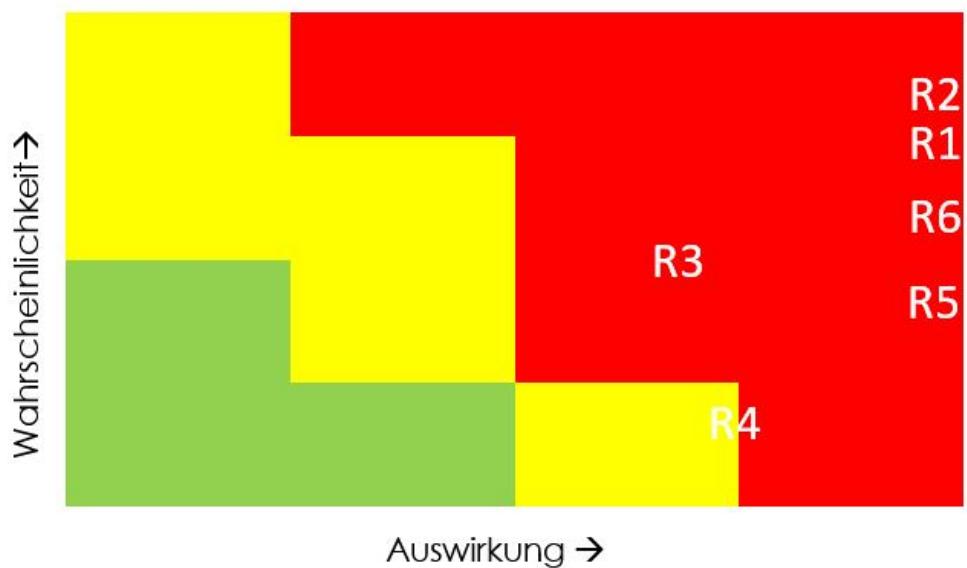
Grafische Darstellung des Umfeldes



### 1.2.3. Risikoanalyse

- Risikomatrix

Risiko	Wahrs.	Auswirkung	Priorisierung	Maßnahmen
R1: Kunden realisieren nicht, dass es die App gibt	8	10	80	Werbung für die App machen (z.B. Plakate, ...)
R2: Die App kommt bei Kunden nicht gut an	9	10	90	Auf Feedback und Verbesserungsvorschläge der Gäste eingehen und die App so gestalten
R3: Mitarbeiter wie Programmierer kündigen	5	7	35	Mitarbeiter motivieren umso solch einem Szenario aus dem Weg zu gehen, ansonsten andere Programmierer mehr Stunden einstellen und einen neuen MA suchen
R4: Datenschutzprobleme	2	8	16	Das betroffene Objekt austauschen, Strafe zahlen
R5: Unzufriedenheit des Auftraggebers	4	10	40	Mit dem Auftraggeber reden, genau seine Wünsche erfüllen, ...
R6: Ungeplante Kosten → Pleite	6	10	60	Auftraggeber um mehr Geld fragen



## 1.3. Pflichtenheft

### 1.3.1. Zielbestimmung

- Projektbeschreibung
- IST-Zustand

Die Stadt Imst (Tourismusverband Imst) hat eine Internetseite, (<http://urlaub.imst.at/>) mit Freizeitangeboten usw. Des Weiteren sind auf <http://www.imst.at/de> diverse Sehenswürdigkeiten unter der Registerkarte „Kultur und Brauchtum“ zu sehen. Eine spezielle Auflistung der Sehenswürdigkeiten, für Einheimische als auch Touristen gibt es bislang nicht, genauso wie eine „Sightseeing“-Funktion, bei der ein Programm eine Sightseeing-Tour durch Imst individuell nach Wünschen planen kann. Da die Stadt Imst einiges an Sehenswürdigkeiten zu bieten hat, angefangen von sämtlichen Brunnen bis zum Fasnachtshaus, bietet sich die Möglichkeit eine App für die oben genannten Funktionen zu erstellen.

- SOLL-Zustand

Die App ermöglicht vor allem Touristen, eine individuelle Tour zu planen, und durch die übersichtliche Auflistung keine Sehenswürdigkeit zu missen. Neben den Sehenswürdigkeiten ist auch eine Liste für Freizeitaktivitäten und mit deren Preise und Standort vorhanden, um den Einwohnern Tirols als auch Urlaubern die Recherche über Ausflüge und Aktivitäten in Imst zu ersparen.

### **1.3.2. Produkteinsatz und Umgebung**

- Anwendungsgebiet  
Stadt Imst
- Zielgruppen  
Touristen und Gäste sowohl auch Einheimische
- Hard-/Softwareumgebung  
Smartphone mit entsprechender App

### **1.3.3. Funktionalitäten**

- MUSS-Anforderungen
  - Funktional
  - Nicht-funktional
- KANN-Anforderungen
  - Funktional
  - Nicht-funktional

MUSS-Anforderungen	KANN-Anforderungen	MUSS-Anforderungen	KANN-Anforderungen
<b>Funktionalität der App: Sie muss flüssig laufen &amp; nicht abstürzen etc.</b>	Der Verbrauch der mobilen Daten und des Akkus sollte gering sein.	Corporate Identity: Die Gestaltung der App im Layout und den Farben des Tourismusverbandes.	Farben können je nach Gefallen des Nutzers umgeändert werden
<b>Übersicht über die Sehenswürdigkeiten in Imst: Mit Informationen zu diesen und Bildern.</b>	Filter für die Liste, um sich beispielsweise nur Brunnen oder Kirchen anzeigen zu lassen, und eine Suchfunktion um eine bestimmte Sehenswürdigkeit schnell zu finden.	Die Übersicht ist schön strukturiert und benutzerfreundlich gestaltet	Die Übersicht kann von jedem User an seinen Gefallen angepasst werden – z. B. in Tabellenform oder Aufzählungen
<b>Eine Liste für Freizeitaktivitäten mit Beschreibung, deren Preise und Standort.</b>	Stetige Aktualität der Öffnungszeiten und Preise der Freizeitangebote.	Nach dem Besuch einer Aktivität kann eine Rezension über diese abgegeben werden, um es anderen Benutzern weiterzuempfehlen.	Abzeichen-System: Beim Besuch eines Ortes erhält man ein Abzeichen.
<b>Unterstützung von Android und iOS, um die App für möglichst viele zugänglich zu machen.</b>	Optional auch Unterstützung von anderen Betriebssystemen.	Durch eine Benachrichtigung die Benutzer auf die Bewertungsfunktion im App-Store hinweisen.	Optional zur App-Store-Bewertung eine eigene Bewertungsfunktion erstellen, um Verbesserungsvorschläge der Kunden einzuholen und zu berücksichtigen.

#### 1.3.4. Testszenarien und Testfälle

- Beschreibung der Testmethodik
- Testfall 1
- Testfall 2
- ...

### 1.3.5. Liefervereinbarung

- Lieferumfang
- Modus
- Verteilung(Deployment)

## 1.4. Planung

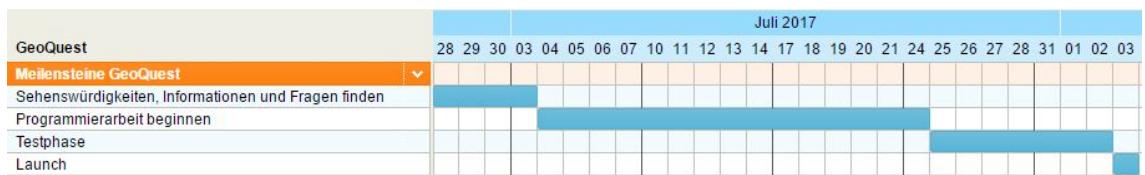
### 1.4.1. Projektstrukturplan



### 1.4.2. Meilensteine

- Sehenswürdigkeiten, Informationen und Fragen finden
- Programmierarbeit beginnen
- Testphase
- Launch

### 1.4.3. Gant-Chart



### 1.4.4. Abnahmekriterien

### 1.4.5. Pläne zur Evaluierung

### 1.4.6. Ergänzungen und zu klärende Punkte

## **2. Vorstellung des Produktes**

Vorstellung des fertigen Produktes anhand von Screenshots, Bildern, Erklärungen.

### **3. Eingesetzte Technologien**

- Kurzbeschreibung aller Technologien, die verwendet wurden.
- Technologien die aus dem Unterricht bekannt sind, nur nennen und deren Einsatzzweck im Projekt beschreiben, nicht die Technologien selbst.
- Technologien die aus dem Unterricht nicht bekannt sind, im Detail beschreiben incl. deren Einsatz im Projekt
- Fokus aus eingesetzten Frameworks

Technologien:

- HTML – für Strukturierung und Zuweisung der Texte auf den verschiedenen Seiten der App.
- CSS – für das Layout der App.
- JavaScript – Programmierung der verschiedenen Funktionen wie Buttons, Analysierung der Sehenswürdigkeiten (mit zum Beispiel QR-Code-Scanner) und laden der Texte und Bilder dazu, usw.

Programme:

- PHP Storm

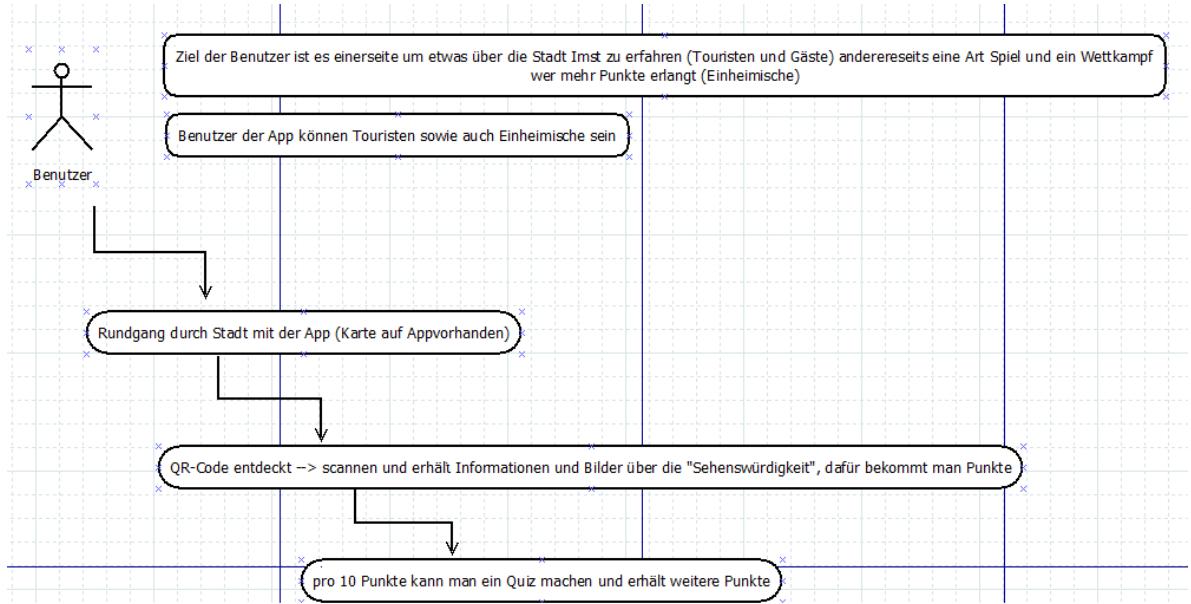
Unsere mobile Web-App basiert auf dem Programm PHP-Storm, dort setzen wir die oben genannten und erklärten Technologien HTML, CSS und JavaScript ein. Als Framework benützen wir das kostenlose LungoJS 1.2, das im folgenden Absatz genauer beschrieben wird: Funktionsumfang: Entwicklung mobiler Web-Apps mit Unterstützung für Touch-Gesten, WebSQL, Browserverlauf, Geolocation, und vielem mehr. Es handelt sich um ein mobiles Framework zur Entwicklung HTML5- basierter Apps für iOS, Android, Blackberry und Windows Phone 7. LungoJS erzeugt semantisches Mark-up und versteht sich auf die Auswertung von Touch- Ereignissen wie Wischbewegungen, die dynamische Handhabung des Ausrichtungswechsels des Displays, WebSQL, und Geolocation. Außerdem können Sie damit den Browserverlauf in Ihren Apps nutzen. Alle UI-Elemente sind vektorbasiert, so dass die Darstellungsqualität unabhängig von der Auflösungsdichte des Displays gewährleistet ist. Das Framework lässt sich mit Hilfe der so genannten Sugars erweitern. Die resultierenden Apps können Sie sowohl über Ihre eigene Website als auch über die offiziellen Distributionskanäle vertreiben.

# **4. Problemanalyse**

## **4.1. USE-Case-Analyse**

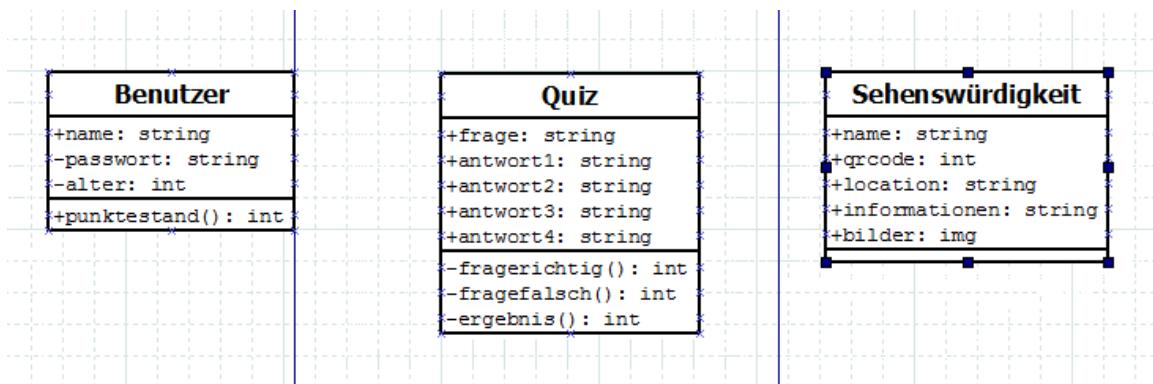
- UseCases auf Basis von Benutzerzielen identifizieren:
  - Benutzer eines Systems identifizieren
  - Benutzerziele identifizieren (Interviews)
  - Use-Case-Liste pro Benutzer definieren
- UseCases auf Basis von Ereignissen identifizieren:
  - Externes Event triggert einen Prozess
  - zeitliches Event triggert einen Prozess (Zeitpunkt wird erreicht)
  - State-Event (Zustandsänderung im System triggert einen Prozess)
- Werkzeuge:
  - USE-Case-Beschreibungen (textuell, tabellarisch)
  - USE-Case-Diagramm
  - Aktivitätsdiagramm für den Use-Case (Interaktion zwischen Akteur und System abbilden)
  - System-Sequenzdiagramm (Spezialfall eines Sequenzdiagramms: Nur 1 Akteur und 1 Objekt, das Objekt ist das komplette System, es geht um die Input/Output Requirements, die abzubilden sind)

## GeoQuest



## 4.2. Domain-Class-Modelling

- "Dinge" (Rollen, Einheiten, Geräte, Events etc.) identifizieren, um die es im Projekt geht
- ER-Modellierung oder Klassendiagramme
- Zustandsdiagramme (zur Darstellung des Lebenszyklus von Domain-Klassen darstellen)



### **4.3. User-Interface-Design**

- Mockups
- Wireframes

# **5. Systementwurf**

## **5.1. Architektur**

### **5.1.1. Design der Komponenten**

Darstellung und Beschreibung der Systemarchitektur;

- statische Zerlegung des Systems in seine physischen Bestandteile (Komponenten, Komponentendiagramm)
- (textuelle) Beschreibung des dynamischen Zusammenwirkens aller Komponenten
- (textuelle) Beschreibung der Strategie für die Architektur, d. h. wie die Architektur in Statik und Dynamik funktionieren soll.
- Verwendung von Referenzarchitekturen bzw. Architekturmustern (als Schablonen, z.B. MVC, Plugin, Pipes and Filters)
  - MVC
  - Schichten
  - Pipes
  - Request Broker
  - Service-Oriented

### **5.1.2. Benutzerschnittstellen**

- Design des UIs
- Dialoge, Dialogsteuerung, Ergonomie, Gestaltung, Eingabeüberprüfungen

### **5.1.3. Datenhaltungskonzept**

- Design der Datenbank (ER-Modell)
- Design des Zugriffs auf diese Daten (Datenhaltungskonzept)
- Caching, Transaktionen

### **5.1.4. Konzept für Ausnahmebehandlung**

- Systemweite Festlegung, wie mit Exceptions umgegangen wird
- Exceptions sind primär aus den Bereichen UI, Persistenz, Workflow-Management

### **5.1.5. Sicherheitskonzept**

Beschreibung aller sicherheitsrelevanten Designentscheidungen

- Design der Security-Elemente
- Design von Safety-Elementen (Fehlertoleranz, Verfügbarkeit etc.)

### 5.1.6. Design der Testumgebung

- wie wird getestet (Unit-Testing, Integrationstesting, Systemtests, Akzeptanztests)
- Testumgebung, Testprozess, Teststrategie, Testmethoden, Testfälle

### 5.1.7. Desing der Ausführungsumgebung

- Deployment (DevOps)
- Betrieb (besonders Hoch- und Hertunerfahren der Anwendung)

## 5.2. Detailentwurf

Design jedes einzelnen USE-Cases

- Design-Klassendiagramme vom Domain-Klassendiagramm ableiten (incl. detaillierter Darstellung und Verwendung von Vererbungshierarchien, abstrakten Klassen, Interfaces)
- Sequenzdiagramme vom System-Sequenz-Diagramm ableiten
- Aktivitätsdiagramme
- Detaillierte Zustandsdiagramme für wichtige Klassen

Verwendung von CRC-Cards (Class, Responsibilities, Collaboration) für die Klassen

- um Verantwortlichkeiten und Zusammenarbeit zwischen Klassen zu definieren und
- um auf den Entwurf der Geschäftslogik zu fokussieren

Design-Klassen für jeden einzelnen USE-Case können z.B. sein:

- UI-Klassen
- Data-Access-Klassen
- Entity-Klassen (Domain-Klassen)
- Controller-Klassen
- Business-Logik-Klassen
- View-Klassen

Optimierung des Entwurfs (Modularisierung, Erweiterbarkeit, Lesbarkeit):

- Kopplung optimieren
- Kohäsion optimieren
- SOLID
- Entwurfsmuster einsetzen

# **6. Implementierung**

Detaillierte Beschreibung der Implementierung aller Teilkomponenten der Software entlang der zentralsten Use-Cases:

- GUI-Implementierung
- Controllerlogik
- Geschäftslogik
- Datenbankzugriffe

Detaillierte Beschreibung der Teststrategie (Testdriven Development):

- UNIT-Tests (Funktional)
- Integrationstests

Zu Codesequenzen:

- kurze Codesequenzen direkt im Text (mit Zeilnummern auf die man in der Beschreibung verweisen kann)
- lange Codesequenzen in den Anhang (mit Zeilennummer) und darauf verweisen (wie z.B. hier ??)

# **7. Deployment**

- Umsetzung der Ausführungsumgebung
- Deployment
- DevOps-Thema

# **8. Tests**

## **8.1. Systemtests**

Systemtests aller implementierten Funktionalitäten lt. Pflichtenheft

- Beschreibung der Teststrategie
- Testfall 1
- Testfall 2
- Tesfall 3
- ...

## **8.2. Akzeptanztests**

## **9. Projektevaluation**

siehe Projektmanagement-Unterricht

# **10. Benutzerhandbuch**

falls im Projekt gefordert

# **11. Betriebswirtschaftlicher Kontext**

BW-Teil

## **12. Zusammenfassung**

- Etwas längere Form des Abstracts
- Detaillierte Beschreibung des Outputs der Arbeit

# **13. Beispielkapitel**

## **13.1. Beispiele zitieren**

Das ist ein Zitat mit Klammern, (Resnick, 1996), das ein Zitat ohne Klammern: Harel und Papert (1991). Hier das selbe Zitat mit einer Seitenangabe und Klammern (Resnick, 1996, S. 23).

Wird ein Absatz aus einer Quelle sinngemäß übernommen (nicht wörtlich), dann kann nach dem Absatz das entsprechende Zitat in Klammern angeführt werden. (Anastopoulou u. a., 2012, S. 33)

Wenn ein Zitat im Text angegeben wird, wie z.B. so Beer, Rudolf und Benischek, Isabella (2011), können die Klammern weggelassen werden.

Der folgende Absatz zeigt ein Blockzitat (wörtlich übernommene Textpassage aus einer Quelle):

Dr. Heinrich Faust ist ein angesehener Wissenschaftler und Akademiker, der trotz seiner wissenschaftlichen Studien und einer guten Bildung seinen Wissensdurst nicht stillen kann. Eines Nachts sitzt er in seinem Studierzimmer und grübelt über den Sinn des Lebens nach, findet jedoch keine Antworten. Daraufhin wendet er sich der Geisterwelt zu. Er beschwört einen Erdgeist, versucht sich den Geistern gleich zu stellen, was ihm jedoch nicht gelingt. Von Ohnmacht

getrieben will er sich das Leben nehmen. Sein Selbstmordversuch wird jedoch von Glockenläuten zum Ostertag und seinen Kindheitserinnerungen gestört. (Ackermann, 2001, S. 21)

Hier wird ein wörtliches Zitat inline angegeben: „Das ist ein kleines direktes Zitat.“ Göhlich und Zirfas (2007), und danach geht es gleich wieder direkt weiter. Ob ein wörtliches Zitat inline oder als eigener Block angezeigt wird, entscheidet Latex auf Basis der Länge.

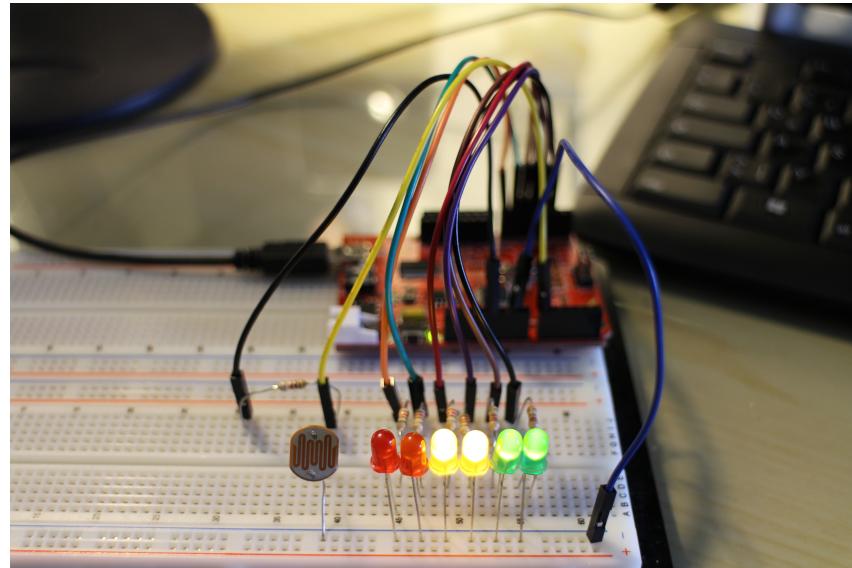
### **13.1.1. Beispiele Abbildungen**

Auf diese Weise kann man zum Beispiel in Latex auf die Abbildung 13.1 verweisen. Die Kennung für den Verweis vergibt man selbst mit dem „label“ Kommando bei der Abbildung.

Jede Abbildung muss nicht nur mindestens einen Verweis im Text haben. Es wird außerdem eine Bildunterschrift verlangt. Für diese ist festgesetzt, dass die Abbildungsunterschrift alleine ausreichend sein muss, um zu verstehen, was am Bild zu erkennen ist.

Der nächste wichtige Punkt sind die Quellenangaben bei Abbildungen. Der Author muss zu jeder Abbildung die notwendigen Rechte haben und idealer Weise gibt man diese bei der Abbildung mit an. In Abbildung 13.1 auf Seite 42 sieht man das.

Es ist wichtig zu verstehen, dass Latex die Positionierung von Abbildungen übernimmt. Man definiert die Abbildung über begin-figure dort, wo man die Abbildung in etwa haben möchte, den Rest übernimmt Latex



© Stefan Stolz (CC BY-SA 3.0)

Abbildung 13.1.: Hintergrund: Arduino Board; Vordergrund: eine Lichterreihe und ein Lichtsensor (Fotowiderstand); In diesem Beispiel wird die Lichterreihe je nach Helligkeit des Umgebungslichtes gesteuert. Durch leichte Modifikationen kann man damit eine Lichtschranke oder auch eine Helligkeitssteuerung für das Smartphone simulieren.

### **Beispiele Tabellen**

Tabelle 13.1 ist ein Beispiel für eine aufwändiger Tabelle mit einer Abbildung und Überschrift.

Tabellen sind in Latex sehr kompliziert zu erzeugen. Alternativ kann man die Tabellen auch in einem anderen Programm gestalten und als Bild wieder einfügen. Dieses Bild kann dann innerhalb von begin-Table verwendet werden.

## **13.2. Beispiele Listen**

Im Folgenden wird eine Liste gezeigt:

DW OR N PACKAGE (TOP VIEW)	
NC	20
V <sub>CC</sub>	19 GND
SER IN	18 SER OUT
DRAIN0	17 DRAIN7
DRAIN1	16 DRAIN6
DRAIN2	15 DRAIN5
DRAIN3	14 DRAIN4
SRCLR	13 SRCK
$\overline{G}$	12 RCK
GND	11 GND
NC – No internal connection	
$V_{cc}$	Positive supply voltage
GND	Ground
SER IN	Daten Pin
SRCK	Clock Pin
RCK	Latch Pin
$\overline{SRCLR}$	Wenn <b>shift-register clear</b> LOW ist, werden die input Register gelöscht Wenn <b>output enable</b> HIGH ist, werden die Daten im Output Buffer LOW gehalten
$\overline{G}$	

Tabelle 13.1.: Aufwändige Tabelle mit Abbildung und Caption

- Ich weiß, dass viele Geräte des täglichen Lebens durch Computer gesteuert werden und kann für mich relevante nennen und nutzen.
  1. Und jetzt eine Numerierung
  2. Und jetzt eine Numerierung
- Ich kann wichtige Bestandteile eines Computersystems (Eingabe-, Ausgabegeräte und Zentraleinheit) benennen, kann ihre Funktionen beschreiben und diese bedienen.

Und jetzt eine Numerierung:

1. Aufzählungspunkt
  - a) Unteraufzählung
  - b) Unteraufzählung
    - Und jetzt noch eine Ebene ohne Aufzählung
    - Und jetzt noch eine Ebene ohne Aufzählung
2. Aufzählungspunkt

3. Aufzählungspunkt
4. Aufzählungspunkt
5. Aufzählungspunkt

## 13.3. Beispiel Codesequenz

In Listing 13.1 sieht man ein Quick-Sort-Listing in der Programmiersprache JAVA. Das Listings-Paket übernimmt die Formatierung von Codebausteinen und kann in der Präambel nach Belieben auf eine andere Sprache konfiguriert werden.

### 13.3.1. Quicksort in JAVA

Quelltext 13.1: QuickSort in Java

```
1 public class QuickSort
2 {
3     public static void main(String[] args)
4     {
5         int [] x =
6         {
7             9, 2, 4, 7, 3, 7, 10
8         }
9         ;
10        System.out.println(Arrays.toString(x));
11
12        int low = 0;
13        int high = x.length - 1;
14
15        quickSort(x, low, high);
```

```
16     System.out.println(Arrays.toString(x));
17 }
18
19 public static void quickSort(int[] arr, int low,
20 int high)
21 {
22     if (arr == null || arr.length == 0)
23         return;
24
25     if (low >= high)
26         return;
27
28     // pick the pivot
29     int middle = low + (high - low) / 2;
30     int pivot = arr[middle];
31
32     // make left < pivot and right > pivot
33     int i = low, j = high;
34     while (i <= j)
35     {
36         while (arr[i] < pivot)
37         {
38             i++;
39         }
40
41         while (arr[j] > pivot)
42         {
43             j--;
44         }
45
46         if (i <= j)
47         {
48             int temp = arr[i];
49             arr[i] = arr[j];
50             arr[j] = temp;
51             i++;
52             j--;
53         }
54     }
55
56     quickSort(arr, low, middle - 1);
57     quickSort(arr, middle + 1, high);
58 }
```

```

48         arr[i] = arr[j];
49         arr[j] = temp;
50         i++;
51         j--;
52     }
53 }
54
55 // recursively sort two sub parts
56 if (low < j)
57     quickSort(arr, low, j);
58
59 if (high > i)
60     quickSort(arr, i, high);
61 }
62 }
```

### 13.4. Beispieltext

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Das hier ist der zweite Absatz. Dies hier ist ein Blindtext zum Testen von Text-

ausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Und nun folgt – ob man es glaubt oder nicht – der dritte Absatz. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Nach diesem vierten Absatz beginnen wir eine neue Zählung. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen

Zweck, da sie eine falsche Anmutung vermitteln.

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

# **Abbildungsverzeichnis**

13.1. Arduino mit Lichtsensor und Lichterkette . . . . . 42

# **Tabellenverzeichnis**

13.1. Aufwändige Tabelle mit Abbildung und Caption . . . . . 43

# Quelltexte

13.1. QuickSort in Java . . . . . 44

# Literaturverzeichnis

[Ackermann 2001] ACKERMANN, Edith: Piaget's constructivism, Papert's constructionism: What's the difference. In: *Future of learning group publication* 5 (2001), Nr. 3, S. 438. – URL [http://lovettresourcenetwork.wiki.lovett.org/file/view/EA.Piaget+\\_+Papert.pdf](http://lovettresourcenetwork.wiki.lovett.org/file/view/EA.Piaget+_+Papert.pdf). – Zugriffsdatum: 2014-07-09

[Anastopoulou u. a. 2012] ANASTOPOULOU, Stamatina ; BERLAND, Matthew ; FRANT, Janete B. ; BOYTCHEV, Pavel ; BRENNAN, Karen ; CHRONAKI, Anna ; CLAYSON, James ; CORREIA, Secundino ; DAGIENE, Valentina ; DEKOLI, Margarita: Constructionism 2012 Theory Practice and Impact. (2012), August. – URL [http://users.uoa.gr/~zsmyrnaiou/conferences\\_after2008/constructionism%201\\_2012.pdf](http://users.uoa.gr/~zsmyrnaiou/conferences_after2008/constructionism%201_2012.pdf). – Zugriffsdatum: 2014-03-26

[Beer, Rudolf und Benischek, Isabella 2011] BEER, RUDOLF ; BENISCHEK, ISABELLA: Aspekte kompetenzorientierten Lernens und Lehrens. In: BIFIE (Hrsg.): *Kompetenzorientierter Unterricht in Theorie und Praxis*. Graz : Leykam, 2011

[Göhlich und Zirfas 2007] GÖHLICH, Michael ; ZIRFAS, Jörg: *Lernen: Ein pädagogischer Grundbegriff*. Stuttgart : Kohlhammer, April 2007. – ISBN 9783170188693

[Harel und Papert 1991] HAREL, Idit ; PAPERT, Seymour: *Situating Con-*

*structionism.* Norwood, N.J : Ablex Publishing Corporation, U.S., 1991. – ISBN 9780893917869

[Resnick 1996] RESNICK, Mitchel: Distributed constructionism. In: *Proceedings of the 1996 international conference on Learning sciences*, International Society of the Learning Sciences, 1996, S. 280–284. – URL <http://dl.acm.org/citation.cfm?id=1161173>. – Zugriffsdatum: 2015-04-20

# **A. Anhang-Kapitel**

## **A.1. Anhang-Section**

Testtext