**Abschlussprüfung Sommer 2016**

Fachinformatiker für Anwendungsentwicklung  
****Dokumentation zu betrieblichen Projektarbeit

**Webseite zur Diablo 3-Datenabfrage**

Entwicklung einer Webseite zur Abfrage   
von Diablo 3 Spiele-Daten

Abgabetermin: 09.09.2016

**Ausbildungsbetrieb:**  
Berufsbildungswerk Neckargemünd  
Im Spitzerfeld 25  
69151 Neckargemünd

**Prüfungsbewerber**  
Tobias Scholl  
Paul-Hindemith-Ring 7  
76669 Bad Schönborn

**Inhaltsverzeichnis**

[Abkürzungs- und Stichwortverzeichnis 3](#_Toc461108437)

[Begriffserklärungen 3](#_Toc461108438)

[1. Einleitung 4](#_Toc461108439)

[1.1. Projektbeschreibung 4](#_Toc461108440)

[1.2. Projektziel 4](#_Toc461108441)

[1.3. Projektumfeld 4](#_Toc461108442)

[1.4. Projektbegründung 4](#_Toc461108443)

[1.5. Projektschnittstellen 5](#_Toc461108444)

[1.6. Projektabgrenzung 5](#_Toc461108445)

[2. Projektplanung 5](#_Toc461108446)

[2.1. Projektphasen 5](#_Toc461108447)

[2.2. Projektkosten 6](#_Toc461108448)

[3. Projektablauf 6](#_Toc461108449)

[3.1. Vorbereitung 6](#_Toc461108450)

[3.2. Landingpage 6](#_Toc461108451)

[3.3. Header und Footer 6](#_Toc461108452)

[3.4. erste API-Abfrage 7](#_Toc461108453)

[3.5. Charakter-Übersicht 8](#_Toc461108454)

[3.6. Mouseover für Items und Fertigkeiten 9](#_Toc461108455)

[3.7. Designen der Item-Slots mit (S)CSS 10](#_Toc461108456)

[3.8. Charakter-Vergleich 11](#_Toc461108457)

[4. Auftragsergebnis 12](#_Toc461108458)

# Abkürzungs- und Stichwortverzeichnis

API Application Programming Interface

GitHub Desktop Desktop-Version des

SASS/SCSS Syntactically Awesome Stylesheet (CSS Erweiterung)

Prepros Preprocessor (um SASS in gewöhnliches CSS zu wandeln)

UX User Experience (Nutzererfahrung, nutzerfreundliche Gestaltung)

# Begriffserklärungen

|  |  |
| --- | --- |
| **Begriff** | **Erklärung** |
| Charakter | die erstellte Spielfigur |
| Hardcore Charakter | spezieller Spielmodus, bei dem der Charakter nach einem Tod nicht wiederbelebt werden kann. |
| Klasse | eine Klasse ist mit einem Beruf zu vergleichen und setzt fest, welche Fertigkeiten und Waffenklassen der Charakter tragen kann. |
| Item / Ausrüstung | Ein Gegenstand im Spiel. Kann zum Beispiel eine Waffe, ein Ausrüstungsteil oder ein Edelstein sein. |
| Qualitätsstufe | Definiert Seltenheit und Stärke eines Items und wird in fünf Stufen unterteilt: Gewöhnlich (weiß), Magisch (blau), Selten (Gelb), Legendär (Orange) und Set-Teil (Grün). Set-Teile sind wie Legendäre Items, allerdings geben Sie bestimmte Boni, wenn man mehrere Items des gleichen Sets trägt. |
| Paragonstufe | Ähnlich des Charakterlevels, Wird erhalten, sobald der Charakter die Maximalstufe (70) erreicht hat. |
| Fertigkeiten | Fertigkeiten sind Kräfte, die der Charakter bei bestimmten Levelstufen freischaltet. |
| Edelsteine / Sockelplätze | Edelsteine sind Items, das in eine Waffe oder Rüstungsgegenstand mit einem Sockelplatz eingesetzt werden kann. Ein Sockelplatz ist wie eine Edelsteinfassung. |

# Einleitung

Die folgende Projektdokumentation schildert den Ablauf des HK-Abschlussprojekts, welches der Autor im Rahmen seiner Ausbildung zum Fachinformatiker für Anwendungsentwicklung durchgeführt hat. Ausbildungsbetrieb ist die Berufsbildungswerk Neckargemünd GmbH (BBW). Die IT-Abteilung des Berufsbildungswerks umfasst über 100 Mitarbeiter und bildet unter anderem Informatikkaufleute, Fachinformatiker für Anwendungsentwicklung und Fachinformatiker für Systemintegration aus.

# Projektbeschreibung

Die Firma XY ist ein Unternehmen, das eine Spieleplattform-Webseite betreibt. Die Webseite veröffentlicht Inhalte wie News und Anleitungen zu verschiedenen Computerspielen. Um die Benutzerzahlen auf der Webseite zu erhöhen, möchte man ein weiteres Spiel hinzufügen.

# Projektziel

Der Spieleplattform-Webseite soll das Computerspiel „Diablo 3“ hinzugefügt werden, jedoch in größerem Umfang als die gewöhnlichen News und Anleitungen. Die Nutzer sollen mit ihrem Account-Namen (so genanntem BattleTag) bestimmte Daten ihrer Charaktere abfragen sowie statistische Werte wie Schaden, Zähigkeit, Erholung und Leben mit anderen Charakteren ihres Accounts vergleichen können.

# Projektumfeld

Die Hardware für das Projekt wird von der Firma zur Verfügung gestellt. Es wird ein Windows 7 64Bit PC mit der Serverumgebung XAMMP zur Verfügung gestellt. Zusätzlich genutzte Programme sind die kostenlosen Entwicklungstools Atom (Editor), Prepros (SASS Preprocessor) und GitHub.

# Projektbegründung

Das Projekt soll mehr Nutzer auf die Webseite locken, um dadurch höhere Nutzerzahlen und mehr Umsatz durch geschaltete Werbung zu erwirtschaften.

# Projektschnittstellen

Die Anwendung wird über die Blizzard-API mit den Servern von Blizzard Entertainment ® (der Entwicklungsfirma von Diablo 3) Daten abfragen.

# Projektabgrenzung

Im Projekt wird **keine** komplette Spieleplattform-Webseite, sondern lediglich eine Webseite mit API-Abfrage und –Ausgabe der Account- und Charakterdaten entwickelt und designt.

Ebenfalls wird die Webseite nicht ausgiebig auf Browserkompatibilität getestet und es wird kein Responsive Design erstellt.

# Projektplanung

# Projektphasen

Das Projekt wird in den zwei Arbeitswochen vom 29. August bis zum 09. September 2016 realisiert. Die Projektzeit beträgt 70 Stunden inklusive Dokumentation, Kundeneinweisung und Abnahme.

|  |  |  |
| --- | --- | --- |
| **Projektphase** | **Aufgabenbeschreibung** | **Soll-Stunden** |
| **Vorbereitung** | Informieren über den Aufbau und die Nutzung der Blizzard-API für Diablo 3 | 6 Stunden |
| **Umsetzung** | Grundgerüst der Oberfläche in HTM planen und erstellen | 4 Stunden |
| Planen und Designen der Oberfläche mit CSS | 8 Stunden |
| Implementieren der API-Abfragen mit PHP | 7 Stunden |
| Entwickeln der Charakter-Übersicht mit PHP | 10 Stunden |
| Erstellen des Charakter-Vergleichs mit PHP | 7 Stunden |
| **Abschluss** | Testen der Webseitenkompatibilität in verschiedenen Browsern | 4 Stunden |
| Erstellung der Dokumentation | 24 Stunden |
| **Gesamtzeit** | | **70 Stunden** |

Tabelle 2: Phasen der Projektarbeit

# Projektkosten

Die Projektkosten werden mit Pauschalsätzen berechnet, die für Personal- und Ressourcenkosten anfallen. Der Stundensatz eines Auszubildenden im 3. Lehrjahr beträgt 10 €, der Stundensatz der Ressourcen (z.B. Strom, Hard- und Softwarenutzung) wird auf 15€ festgesetzt. Als Entwicklungsressourcen wurden ein PC mit Windows 7 64Bit und die Entwicklungstools Git Desktop, XAMMP, Prepros und Atom Text Editor verwendet.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Vorgang** | **Zeit** | **Personalkosten** | **Ressourcenkosten** | **Gesamtkosten** |
| Entwicklung | 70 h | 700,00 € | 1050,00 € | 1750,00 € |

# Projektablauf

Tabelle 1: geschätzte Projektkosten

# Vorbereitung

Um den Aufbau der API von Blizzard zu verstehen, habe ich zuerst mithilfe einer Demo-Webseite von Blizzard grundlegende Abfragen zu Account-, Charakter- und Item-Daten getätigt. Somit habe ich mir einen Überblick über die nötigen Methoden wie „file\_get\_contents“ (Inhalt einer Webseite kopieren) und „json\_decode“ (Text in ein JSON-Array speichern) in PHP verschafft. Für diese Vorbereitung wurde weniger Zeit als geplant benötigt, da die Demo-Webseite inklusive Dokumentation seitens Blizzard sehr verständlich ist (mit Ausnahme einer Sache, dazu aber später mehr).

# Landingpage

Als zweiten Schritt habe ich eine Startseite (sog. Landing Page) mit HTML und CSS entwickelt. Diese Seite soll veranschaulichen, welche Funktionen die Webseite beinhaltet. Diese Idee hatte ich erst nachträglich und wurde mit der gewonnenen Zeit aus der Vorbereitung (ca. eine Stunde) realisiert.

# Header und Footer

Der dritte Schritt war nun, ein Grundgerüst für Header und Footer in HTML und CSS zu schreiben. Dieses bleibt auf jeder weiteren Seite identisch, sodass die Nutzererfahrung (UX) ebenfalls gleich bleibt.

Das CSS für den Header und Footer wurden mithilfe SCSS in eigene Dateien geschrieben, sodass diese einfacher gewartet und geändert werden können. Diese Dateien werden dann in einer SCSS-Hauptdatei importiert und mithilfe von Prepros in standardisiertes CSS „übersetzt“.   
Da ich in einem kurz zuvor absolvierten Praktikum sehr viel über HTML5, CSS3 und deren konforme Umsetzung gelernt habe, wurde auch an diesem Projektpunkt weniger Zeit (ungefähr 2 Stunden) als geplant benötigt.

# erste API-Abfrage

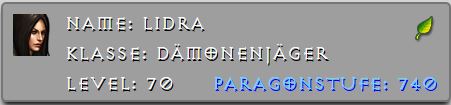
Nachdem das Grundgerüst stand habe ich begonnen die ersten API-Abfragen für den Account zu erstellen. Diese Request-URLs setzen sich aus mehreren Teilen zusammen: der **statische battle.net URL**, der **BattleTag** (ausgelesen aus dem Input-Feld), die **Lokalisierung** und der **API-Key**. Beispiel:  
**https://eu.api.battle.net/d3/profile/Hurric4ne-2268/?locale=de\_DE&apikey= 27hd5khrafctn6dv8g9egtbt25qbzcqz**Teile der von der API gesendeten Daten werden dann dynamisch mit HTML und CSS in Charakter-Containern untergebracht (Bild 1).

Bild 1: Charakter-Container

Der Charakter-Container ist anklickbar, um weiter auf die Charakter-Übersicht zu gelangen. Hier bin ich auf das erste Problem gestoßen, da die Links des Containers dynamisch sein müssen.

**Problem:** Um auf der Charakter-Übersicht die richtigen Daten zu erhalten und anzuzeigen, muss die Request-URL des richtigen Charakters auf die nächste Seite übertragen werden.

**Erste Lösungsidee:** Die Request-URL wird komplett an die vorhandene PHP-URL angefügt. Allerdings wird so die URL sehr lange, da die URL dann aus zwei Links besteht. Ein größeres Problem ist allerdings, dass durch diese Lösung der API-Key im Klartext in der URL steht. Dieser Key ist zwar nicht wie ein Passwort, fremde Personen könnten allerdings damit ebenfalls API-Requests durchführen.

**Zweite Lösungsidee:** Nur die Charakter-ID wird an die PHP-URL angefügt. Die Request-URLs aller Charaktere werden in ein Session-Array gespeichert, bei dem der Array-Index der Charakter-ID entspricht. So lässt sich mit $\_SERVER[“REQUEST\_URI“] die Charakter-ID auf der Folgeseite wieder auslesen und kann als Index für das Session-Array genutzt werden.

# Charakter-Übersicht

Nachdem die Account-Auswahl mit den Charakter-Container und die Verlinkung auf die nächste Seite funktioniert, habe ich begonnen ein Charakter-Layout basierend auf der Übersicht im Spiel zu erstellen (Bild 2).

Bild 2: Charakter-Übersicht

Um die Bilder und Anker-Links für Items und Skills richtig zu erstellen, habe ich weitere API-Abfragen erstellt. Dabei ist ein weiteres Problem aufgetaucht, da bei leeren Fertigkeiten oder Charakter-Items API-Abfragen an den Server eine Fehlermeldung ausgeben (da nichts hinterlegt ist).

**Problem:** Wenn ein Charakter einen Slot in der Fertigkeiten-Leiste noch nicht freigeschaltet, keine Fertigkeit eingesetzt oder kein Charakter-Item angelegt ist, entstehen leere Bild- und Anker-Tags.

**Lösung:** Mit einem Isset-If wird zuerst abgefragt, ob eine Fertigkeit oder Item gesetzt ist. Wenn Ja, wird der API-Request ausgeführt und der Anker- und IMG-Tag erstellt und mit den Request-Daten (Link und Bild-URL) gefüllt, ansonsten wird der Vorgang übersprungen und es wird zur nächsten Fertigkeit / zum nächsten Item gesprungen.

# Mouseover für Items und Fertigkeiten

Mithilfe der von Blizzard Entertainment ® zur Verfügung gestellten „Tooltips.js“ kann ein Mouseover bei einem Anker-Tag automatisch generiert werden, der die Werte des Items anzeigt (Bild 3).

Bild 3: Mouseover einer Fertigkeit

Um diese Mouseover zu erstellen, muss das Bild mit einem bestimmten Anker-Link umgeben sein. Standartwerte für Items und die Fertigkeiten werden mit der **statischen URL** von Blizzard, der Auswahl ob **Item oder Fertigkeit** und dem englischen **Item- oder Fertigkeitsnamen** erstellt. Beispiel:  
**http://eu.battle.net/d3/de**/**item**/**yangs-recurve**

Um die echten Werte des Items zu erhalten ist es notwendig, den Link mit einem für das Item erstellten Hashwert anzupassen. Dieser Hashwert wird dann von der „tooltips.js“ übersetzt und die richtigen Werte werden im Mouseover angezeigt.

**Problem:** Aufgrund ungenauer Beschreibung der Blizzard API-Dokumentation war unklar, wie genau ein Link für ein Item auszusehen hat, der von der Tooltips.js richtig interpretiert wird.

**Lösung:** Damit die Tooltips.js die Werte des Items als Mouseover richtig erstellt, muss anstatt des Item-Namens der Item-Hashwert an die URL angefügt werden.

Beispiel: **http://eu.battle.net/d3/de/item/Item-Hashwert**

Durch die ungenaue Beschreibung der Dokumentation habe ich sehr lange an dieser Problematik gesessen, bis ich die Lösung gefunden hatte. Die geplanten 10 Stunden sind auf ca. 14 Stunden angewachsen, wodurch meine Planung um 2 Stunden im Verzug war (durch die verkürzte Design-Zeit hatte ich 2 Stunden Puffer gewonnen).

# Designen der Item-Slots mit (S)CSS

Nachdem die richtigen Items, Fertigkeiten und deren Icons richtig angezeigt wurden, ging es nun daran, die Icons zu positionieren und die Hintergründe gemäß der Qualitätsstufe des Items zu färben. Beispiele:

seltener  
Gegenstand

legendärer  
Gegenstand

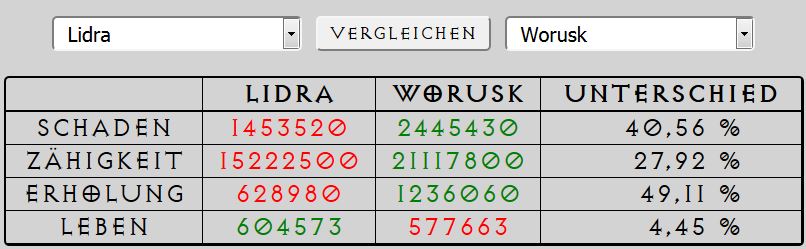
Set-Gegenstand

**Problem:** Verschiedene Item-Icons von Brustteilen sind unterschiedlich groß, sodass eine generelle Positionierung des IMG-Tags zuerst nicht möglich war.   
**Lösung:** Änderungen an der Positionierung des IMG-Tags in Kombination mit der CSS-Eigenschaft „overflow:hidden“ hat dieses Problem gelöst.

Die Ausgabe der statistischen Werte wurde ebenfalls mit den gleichen API-Abfragen erstellt und verlief problemlos.

Zusätzlich gibt es auch Items, die einen so genannten Sockelplatz besitzen. Ob ein Item einen oder mehrere Sockelplätze enthält, wird ebenfalls aus dem JSON Code ausgelesen. Falls ein Sockelplatz im Item enthalten ist, wird der IMG-Tag mit einem Isset-If ausgegeben.  
**Problem:**Set-Waffen und Nebenhand-Items haben keine Sockel im JSON-Code. Dieses Problem kann nicht in der Projektzeit gelöst werden, da die API keine Möglichkeit enthält. Für eine Lösung müsste das Projekt mehr Zeit erhalten.

# Charakter-Vergleich

Die letzte Funktion der Webseite soll es ermöglichen, zwei Charaktere eines Accounts zu vergleichen. Die Auswahl der Charaktere wurde über zwei Dropdown-Menüs realisiert. Diese Dropdown-Menüs werden dynamisch mithilfe der Charakternamen erstellt. Da die benötigten API-Abfragen allerdings mit der Charakter-ID erstellt werden müssen, wurde das value-Feld des option-Tags mit den Charakter-IDs generiert. Dadurch kann man mit $\_POST die ID des ausgewählten Charakters dynamisch in die Request-URL einbauen.

Das Erstellen des Charakter-Vergleichs verlief wesentlich einfacher als geplant, von den gelplanten 7 Stunden wurden nur ca. 3 Stunden benötigt. Dadurch habe ich auch die überschrittene Zeit der „Mouseover Problematik“ (2 Stunden) wieder aufgeholt und eine Stunde Pufferzeit gewonnen. Diese gewonnene Zeit wurde in ein weiteres Feature investiert, das mir während der Entwicklung des Charakter-Vergleichs eingefallen ist:

Bild 4: Charakter-Vergleich

**Anzeigen des prozentualen Unterschieds zwischen beiden Charakterwerten**

Das Ergebnis der Berechnung wird auf die letzten zwei Nachkommastellen gerundet, damit die Tabellenspalte nicht zu lang wird.

# Auftragsergebnis

Hier liste ich nochmals alle Features des Projekts auf, inklusive der beiden zusätzlich, nicht im Projektantrag enthaltenen Features:

* Landingpage zur Übersicht der Features
* Account-Auswahl und Anzeige der Charakter mit Name, Klasse, Level und Paragonlevel
* Charakter-Übersicht mit Name, Klasse, angelegte Items und Fertigkeiten inklusive Mouseover für die Items und Fertigkeiten
* Charakter-Vergleich zum Vergleichen statistischer Werte zweier Charaktere desselben Accounts inklusive Anzeige der prozentualen Unterschiede.

**Quellen:**

**Deckblatt:**

http://www.diablo-3.net/wp-content/uploads/2014/07/diablo3-reaper-of-souls-logo\_seite.png

**Toolitps.js:**

https://github.com/dawg6/dhcalc/blob/master/war/tooltips.js

**Landingpage Background:**

http://media.blizzard.com/d3/media/artwork/artwork-0107-full.jpg

**Favicon:**

http://vignette3.wikia.nocookie.net/diablo/images/9/96/Diablo\_III\_ROS\_icon.png/revision/latest?cb=20140316104854

**Skill-Overlay:**

http://eu.battle.net/d3/static/images/profile/hero/skill-overlays.png

**Dialo-Font:**

http://eu.battle.net/d3/static/fonts/exocet/exocet-blizzard-light.woff