Arbeitsplan

- Überarbeitete Version -

Projektleitung: Otto Bittner

24.01.17

GBAW-17

1 Projektvision

Die Universität Leipzig ist beteiligt an der Entwicklung einer Software, die einen zentralen Umschlagplatz für Präsentationen bietet. Die Lernsoftware soll im Rahmen dieses Projektes um ein Allgemeinwissens-Quiz erweitert werden. Es gilt eine autarke Webapplikation zu implementieren, welche nach einem auszuwählenden Zeitabschnitt den Arbeitsprozess pausiert und mithilfe von Quiz-Fragen auflockert. Diese regelmäßigen Pausen vom konzentrierten Arbeiten lehnen sich an die Pomodoro-Theorie an und sollen in spielerischen Stil den Workflow des Nutzers verbessern.
Somit ergibt sich folgende Projektvision:
Ein kleines Popover soll nach dem ausgewählten Zeitabschnitt im oberen rechten Bildschirm Abschnitt auftauchen. Nach einem kurzen Begrüßungsbildschirm, welcher automatisch verschwindet - falls ignoriert - zeigt die Webapplikation eine zu beantwortende Frage mit 4 Antwortmöglichkeiten und deren farblich abgehobenen Schaltflächen. Über kleinere Symbole lassen sich zusätzliche Einstellungen, Ranglisten und Profil anzeigen. Das Design soll einfach und übersichtlich gehalten werden, da das Projekt grundlegend jede Zielgruppe ansprechen soll. Die Sprachen für Menü und Fragen soll Englisch sein. Jeder Nutzer bekommt ein eigenes Quiz-Profil mit einem Nickname zugeteilt, worüber erspielte Punkte und Belohnungen gespeichert werden können. Über diese Punkte können sich die Nutzer untereinander vergleichen.

2 Voraussetzungen

Das Projekt SlideWiki ist noch in der Entwicklungsphase weshalb vorerst keine direkte Anbindung des Bots vorgesehen ist. Stattdessen soll eine autonome und plattformunabhängige Applikation entwickelt werden die zu einem späteren Zeitpunkt relativ einfach in verschiedenen Projekten eingebunden werden kann. Entsprechend ist eine ausreichende Dokumentation zur Nutzung und Einbindung der Applikation nötig.
Neben einer ansprechenden Darstellung, die mit Hilfe von SemanticUI erstellt werden soll, muss auch im Frontend Logik implementiert werden, welche das Stellen der Fragen, Annahme der Benutzereingaben oder Zählen der Zeit übernimmt. jQuery ist vorgesehen dabei das erste Grundgerüst liefern zu können.
Sicherzustellen ist, dass z. B. beim Neuladen der Seite der Timer sich nicht zurücksetzt. Eine mögliche Lösung wäre SessionIDs zu vergeben und auf dem Server benötigte Informationen festzuhalten. Auf dem Client kann dann die Einhaltung von Zeiten zB. anhand eines Cookies oder durch einen Push-Service gesichert werden. Darüber hinaus muss das Backend die Verwaltung der Fragen, Ranglisten usw. übernehmen. Geplant ist dafür das Python Framework Django zu verwenden. Damit das Spiel nicht langweilig wird muss ein Fragenkatalog mit großem Umfang vorhanden sein. Hierfür soll die online zur Verfügung stehende API “Open Trivia Database” genutzt werden. Diese Fragen werden auf dem Server in der Datenbank gespeichert. Die Möglichkeit später andere Frageformate nutzen zu können soll offen gehalten werden. PostgreSQL scheint hier ausreichende Freiheiten zu bieten und die Anforderung zu erfüllen.

3 Designübersicht und Funktionalität

3.1 Design

Wie bei modernen Webanwendungen typisch soll auch bei diesem Projekt grundsätzlich eine Unterteilung in Front- und Backend stattfinden. Der Server übernimmt dabei z.B. die Verwaltung der Nutzerdaten, sowie die Auswahl von auf den Nutzer zugeschnittenen Fragen (Fragen sollen sich bspw. so wenig wie möglich wiederholen).
Auf Clientseite wird ein Quiz zur Verfügung gestellt, welches, ohne vorher langwierig Einstellungen vornehmen zu müssen, sofort beantwortet werden kann. Entsprechend soll eine zugängliche Oberfläche geschaffen werden, die dem Endbenutzer den Umgang mit der Software ohne weitere Erklärung erlaubt.

3.2 Funktionalität

3.2.1 Muss Ziele

3.2.1.1 Wiedererkennung von Nutzern

Akteur: Server
Ein registrierter Nutzer soll zu jeder Session vom Server bzw. der Applikation wiedererkannt werden um bereits beantwortete Fragen sowie Punktestände beizubehalten. Diese Funktion soll über eine einzigartige ID jedes registrierten Nutzers
realisiert werden.

3.2.1.2 Optionalität des Quiz

Akteur: Nutzer
Auf der jeweiligen Webseite arbeitende Nutzer haben die Möglichkeit Fragen zu beantworten, sie zu überspringen oder den Bot gänzlich abzulehnen. Auch eine ausbleibende Eingabe des Nutzers soll als Ablehnung erfasst werden.

3.2.1.3 Wertung der Fragen

Akteur: Server
Nutzer erhalten für richtig beantwortete Fragen Punkte. Anhand der erreichten Punkte können die Spieler in ihrer Performance verglichen werden. Diese Wertung kann durch Level und Ränge ergänzt werden, was in den Kann Zielen weiter beschrieben ist.

3.2.1.4 Zeitlimit für Fragen

Akteur: Client
Jede Frage besitzt das gleiche Zeitlimit um sie zu beantworten. Der Nutzer
soll in der Lage sein anhand einer Fortschrittsanzeige (Uhr o.ä.) zu erkennen wie viel Zeit er noch übrig hat um eine Frage zu beantworten.

3.2.1.5 Erkennung der Arbeitsintervalle

Akteur: Client
Das Pomodoro Prinzip fordert feste Zeiten für Arbeit und Pausen. Um diese zu gewährleisten muss ein Timer eingerichtet werden, welcher auch über das Neuladen der Seite hinaus, richtig erkennt wie viel Zeit seit dem letzten Quiz vergangen ist und entsprechend eine neue Runde veranlasst.

3.2.2 Kann Ziele

Diese Ziele stellen sinnvolle Erweiterungen der Software dar, nachdem die Muss Ziele vollständig umgesetzt wurden. Sie sind für die Einsetzbarkeit der Software aber nicht notwendig. Die Reihenfolge stellt eine absteigende Priorisierung dar.

3.2.2.1 Wertung einsehen

Akteur: Nutzer
Jeder Nutzer kann seine eigene Wertung einsehen und wie viele Fragen er richtig/falsch beantwortet hat. Er bekommt außerdem nach jeder Quizrunde eine Aufschlüsselung welche Frage der Runde wie beantwortet wurde.

3.2.2.2 Profil Einstellungen

Akteur: Nutzer
Nutzer haben die Möglichkeit ihr Profil zu bearbeiten um den Zeitraum nachdem ein Quiz auftaucht zu bearbeiten, ein Avatar-Bild anzupassen oder die Sichtbarkeit ihrer Wertungen einzustellen (öffentlich/privat).

3.2.2.3 Belohnungen

Akteur: Server
Durch das Erreichen von bestimmten Zielen, wie etwa die korrekte Beantwortung von 3 Fragen in Folge, erhalten Nutzer zusätzliche Belohnungen. Diese Belohnungen können sich durch Bonuspunkte oder auch durch Abzeichen/Ränge auszeichnen, die einen Nutzer weiter motivieren sollen.

3.2.2.4 Änderbarkeit des Fragenkatalogs

Akteur: Administrator
Um den Fragenkatalog so gut wie möglich zu gestalten haben Administratoren die Möglichkeit anhand von Statistiken bestimmte Operationen auf Fragen durchzuführen.
Beispielsweise können neue Fragen hinzugefügt und alte gelöscht werden.

3.2.2.5 Statistiken zu Fragen

Akteur: Administrator
Administratoren können allgemeine Statistiken abrufen. Zum Beispiel wie viele Runden gespielt wurden, wie viel falsch beantwortet wurde, wie oft der Bot ignoriert wurde, usw.

3.2.2.6 Feedback-Funktion

Akteur: Nutzer, Administrator
Der Nutzer hat bei Fragen die ihm nicht gefallen die Möglichkeit Feedback in Form einer E-Mail zu hinterlassen. Ein extra Button, welcher während dem Beantworten der Frage angezeigt wird, öffnet hierzu einen Dialog, der den Nutzer nach dem Inhalt seines Feedbacks fragt.

3.2.2.7 Abwesehnheit des Nutzers erkennen

Akteur: Server
Der oben erwähnte Timer, welcher die Arbeitsintervalle misst, soll pausiert werden, sobald der Nutzer für einen festgelegten Zeitraum keine Eingabe getätigt hat, ähnlich zu einem Bildschirmschoner. Die Einhaltung der Zeiträume nach dem Pomodoro Prinzip wird so weiter unterstützt.

3.2.2.8 Multimedialer Fragenkatalog

Akteur: Nutzer / Client
Die Art der Fragen wird erweitert und beinhaltet nicht mehr nur Multiple-Choice-Fragen. Lückentexte, Bilderrätsel oder mathematische Rätsel sind Beispiele. Dabei hat der Nutzer die Option den Fragenkatalog einzuschränken und damit zu entscheiden ob er derartige Fragen beantworten will.

4 Arbeitspakete

4.1 Infrastruktur einrichten (5%, R1)

Im Rahmen dieses Arbeitspaket sollen die nötigen serverseitigen Infrastrukturen eingerichtet werden. Auf dem Server soll eine Django Instanz installiert sein, sowie eine lauffähige PostgreSQL Installation. Der Webserver soll so konfiguriert werden, dass auf einer Unterseite der Projektseite der aktuelle Stand der Software gezeigt werden kann.
Außerdem soll eine Docker Umgebung eingerichtet werden, welche die Entwicklung auf verschiedenen Systemen angleicht.

4.2 Modellierung (10%, R2)

Nun muss eine konkrete Modellierung für die Software erstellt werden. Dafür sind etwaige Architekturvorgaben durch die genutzten Frameworks umzusetzen, sowie alle nötigen funktionalen Komponenten zu identifizieren und zu modellieren. Die Modellierung soll sich in die Bereiche Frontend, Backend und Kommunikation unterteilen. Hierbei werden weitere Verfeinerungen und Unterteilungen nötig, die zum Zeitpunkt der Modellierung festgelegt werden. Als Teil der Modellierung für das Frontend soll ein Mokup erstellt werden.

4.3 Prototypisierung (25%, R2)

Diese Aufgabe besteht daraus einen funktionierenden Prototypen zu erstellen. Der funktionale Umfang dieses Abschnitts ist unter 5. Vorprojekt beschrieben. Der Fragenkatalog, welcher Teil der Software ist, wird zu diesem Zeitpunkt noch nicht sichtbar sein. Jedoch sollen die Fragen, welche durch die externe API “Open Trivia Database” zur Verfügung gestellt werden beispielhaft vorgestellt werden, um ein Feedback vom Kunden zu erhalten. Sollte der Kunde mit den Möglichkeiten der APIs nicht zufrieden sein, wird es Teil der folgenden Arbeitspakete sein, diesen Katalog händisch zu erstellen. Tritt dieser Fall ein, wird die Umsetzung der meisten Kann-Ziele als unrealistisch bewertet.

4.4 Einarbeitung der Muss Ziele (40%, R3-4)

Zu diesem Zeitpunkt werden Features, welche fester Bestandteil der Software sein sollen aber nicht Teil des Prototypen sind, implementiert. Der Fragenkatalog ist in jedem Fall Teil hiervon. Entweder ist dies die Anbindung einer bestehenden API oder ein selbst erstellter Fragenkatalog. Außerdem soll die Einbettung der Software in eine größere Website erprobt werden. Hierdurch soll die Machbarkeit dieser Aufgabe gesichert werden und eventuelle Anpassungen in der Architektur angestoßen werden. Auch soll die Oberfläche optisch weiterentwickelt werden.
Mit Abschluss dieses Arbeitspaket sollen alle Muss-Ziele enthalten sein.
Sollte sich, wie im letzten Arbeitspaket angedeutet, herausstellen, dass ein Fragenkatalog händisch erstellt werden muss, erhöht sich hier die Arbeitslast um etwa 10%.

4.5 Auswahl und Planung von Kann-Zielen (5%, R5)

Bei erfolgreichen und termingerechten Abschluss von Arbeitspaket 4 werden nun, unter Rücksprache mit dem Kunden, Kann Ziele gewählt welche noch realistischerweise abgeschlossen werden können bis zur Abnahme. Diese müssen außerdem modelliert werden.

4.6 Einarbeitung der Kann Ziele (25%, R5-6)

Die ausgewählten Kann Ziele werden nun umgesetzt. Geplant ist die Umsetzung der Kann Ziele 3.2.2.1 bis 3.2.2.4.

5 Vorprojekt

Im Vorprojekt soll ein lauffähiger Prototyp erstellt werden, welcher sich aber äußerlich und inhaltlich noch in einer Rohform befinden.
Der Bot soll sich hierfür bereits als Pop-over zeigen. Die Anzeige von Fragen und deren Beantwortung soll funktionieren. Der Mokup für die äußerliche Gestaltung ist außerdem präsentierbar.

6 Glossar

6.1 Datenbanksystem

Datenbanksysteme werden dazu genutzt elektronische Daten dauerhaft, effizient und widerspruchsfrei zu speichern. Besonderes Merkmal dieser Systeme ist die Möglichkeit die Daten auf verschiedene Arten darzustellen und entsprechend für verschiedene Anwender passend aufzubereiten.

6.2 Pomodoro

Die Pomodoro-Technik ist ein Zeitmanagement-Konzept welches von Francesco Cirillo in den 80er Jahren entwickelt wurde. Dabei werden die Arbeiten in zumeist 25 Minuten Abschnitte unterteilt die von kleinen Pausen (meist 5 Minuten) unterbrochen werden, aller 4 Abschnitte ist dabei eine größere Pause von 20-30 min vorgesehen. Während der 25 min soll sich dabei rein auf die Aufgabe Konzentriert werden. Störungen und Unterbrechungen sind zu notieren um sich später darum kümmern zu können. Am Ende der 25 Minuten setzt man einen Schlussstrich unter das Aktuelle Pomodoro , was den unmittelbaren Effekt hat zu sehen wie viel man erledigt hat. Außerdem stellt man so Daten zur späteren Selbstreflexion bzw. für das Projektmanagement bereit.

6.3 WebApp

Eine Webanwendung (auch Online-Anwendung, Webapplikation oder kurz Web-App) ist ein Anwendungsprogramm nach dem Client-Server-Modell. Anders als klassische Desktopanwendungen werden Webanwendungen also nicht lokal auf dem Rechner des Benutzers installiert und dort ausgeführt. Die Datenverarbeitung und -auswertung findet stattdessen hauptsächlich auf einem entfernten Webserver statt. Nur das Ergebnis der Datenverarbeitung wird zur Anzeige oder Ausgabe an den lokalen Client Rechner des Benutzers übertragen (Thin Client). Genutzt wird eine Webanwendung dabei in der Regel über einen Webbrowser.

6.4 Frontend/ Backend

Die Begriffe Front-End und Back-End werden in der Informationstechnik an verschiedenen Stellen in Verbindung mit einer Schichteinteilung verwendet. Dabei ist typischerweise das Front-End näher am Benutzer, das Back-End näher am System. Es gilt aber prinzipiell, dass das Front-End näher an der Eingabe und das Back-End näher an der Verarbeitung ist. Bei Client-Server Anwendungen wird das auf dem Client laufende Programm als Front-End, das auf dem Server laufende als Back-End bezeichnet.

6.5 API

Eine Programmierschnittstelle, ist ein Programmteil, der von einem Softwaresystem anderen Programmen zur Anbindung an das System zur Verfügung gestellt wird. Neben dem Zugriff auf Datenbanken oder Hardware wie Festplatte oder Grafikkarte kann eine Programmierschnittstelle auch das Erstellen von Komponenten der grafischen Benutzeroberfläche ermöglichen oder vereinfachen.

6.6 Gamification

Gamification beschreibt das Einbinden von Spiel typischen Elementen in eine Spiel fremde Umgebung. Dazu zählen z.B. Fortschrittsbalken, Erfahrungspunkte, Highscores, Ranglisten und virtuelle Güter oder Auszeichnungen. Dies soll in erster Linie zur Motivationssteigerung beitragen um wenig herausfordernde monotone oder zu komplexe Aufgaben besser bzw. motivierter zu erledigen. Der Erfolg von Gamification ist vor allem von der Haltung des Anwenders und die persönliche Affinität zu Spielen abhängig.

6.7 Framework

Ein Framework (Rahmenstruktur) ist ein Programmiergerüst, das in der Softwaretechnik, insbesondere im Rahmen der objektorientierten Softwareentwicklung sowie bei komponentenbasierten Entwicklungsansätzen, verwendet wird. Im allgemeineren Sinne bezeichnet man mit Framework auch einen Ordnungsrahmen. Ein Framework ist selbst noch kein fertiges Programm, sondern stellt den Rahmen zur Verfügung, innerhalb dessen der Programmierer eine Anwendung erstellt, wobei u. a. durch die in dem Framework verwendeten Entwurfsmuster auch die Struktur der individuellen Anwendung beeinflusst wird.

6.8 Client-Server-Architektur

Die Client-Server-Architektur beschreibt eine Möglichkeit, Aufgaben, die einem gemeinsamen Ziel zuarbeiten, auf verschiedene Rechner innerhalb eines Netzwerks zu verteilen. Hierbei findet eine Unterscheidung in die Rollen des Servers und des Clients statt. Beide Rollen werden letztlich durch einen Prozess realisiert, sie können also auch auf einem Rechner ausgeführt werden. Ziel dieser Architektur ist es, Dienste an einer zentralen Stelle verfügbar zu machen, also die Möglichkeit zu schaffen, dass mehrere Clients mit der gleichen Software oder den gleichen Daten arbeiten können, ohne diese selbst ausführen/vorhalten zu müssen.

6.9 Ajax (Asynchronous JavaScript and XML)

Im Gegensatz zu einem klassischen Modell einer Webanwendung, bei der nach einer Serveranfrage die komplette Seite neu erstellt wird, kann im Ajax Modell eine asynchrone Datenübertragung zwischen dem Browser und Server stattfinden. So wird nur ein bestimmter Teil der Website nach einer Abfrage geändert während der restliche Teil unverändert angezeigt wird.

6.10 DOM

Document Object Model ist eine Spezifikation für HTML und XML die eine Struktur im Sinne der objektorientierten Programmierung darstellt. Ziel ist dabei auf die strukturierten Daten einfach und einheitlich zugreifen zu können.