Doku GIBZ-Fiesta

Ramon Camenzind, Timo Schlumpf & André Möll

M326

GIBZ

**Inhaltsverzeichnis**

[THEORIE 2](#_Toc118398338)

[Vorwort 2](#_Toc118398339)

[Aufgabenstellung 2](#_Toc118398340)

[Aufbau 2](#_Toc118398341)

[Funktionen 2](#_Toc118398342)

[Mittel und Methoden 2](#_Toc118398343)

[Vorkenntnisse 2](#_Toc118398344)

[Zeitplan 2](#_Toc118398345)

[Arbeitsjournal 2](#_Toc118398346)

[Datum 1 2](#_Toc118398347)

[Datum 2 2](#_Toc118398348)

[…. 2](#_Toc118398349)

[PRAXIS 2](#_Toc118398350)

[Einleitung 2](#_Toc118398351)

[Informieren 2](#_Toc118398352)

[Planen 2](#_Toc118398353)

[Entscheiden 2](#_Toc118398354)

[Realisieren 2](#_Toc118398355)

[Kontrollieren 2](#_Toc118398356)

[Reflexion 2](#_Toc118398357)

[Anhang (?) 2](#_Toc118398358)

# THEORIE

**1 Vorwort**

Diese Dokumentation dient zur Erklärung und Festhaltung von relevanten Eckdaten der Projektarbeit des Jahres 2022 im Modul 326 am GIBZ.

Die Dokumentation ist dabei in zwei Teile unterteilt: Theorie und Praxis – Theorie dient dabei zur vorgängigen Klärung der Umstände und Angehensweise, während es in der Praxis dann um die tatsächliche Umsetzung des Projektes geht.

Für die Realisierung des Projektes wurde die Projektplanungsmethode IPERKA verwendet.

Dieses Projekt trägt den Namen **GIBZ FIESTA**.

**2 Aufgabenstellung**

Im Modul 326 soll gemäss Modulidentifikation eine objektorientierte Analyse (OOA) in ein objektorientiertes Design (OOD) überführt sowie implementiert, getestet und dokumentiert werden.

**2.1 Aufbau**

Dieses Projekt dient unter Anderem zur Vorbereitung auf die praktische Abschlussarbeit, welche wir als Abschluss der Informatiker-Lehre ablegen müssen. In besagter PA haben die Individuen jeweils 10 Tage Zeit, ein Projekt in die Realität umzusetzen. Dieses Projekt soll also helfen, um Erfahrungen mit einem längerfristigen Projekt und dessen Ablauf zu sammeln und erweitern.

Inhaltlich geht es in der Projektarbeit des Moduls 326 um die Erstellung einer Software, welche von den Lehrpersonen des GIBZ zur langfristigen Evaluation ihrer digitalen, unterrichtsbezogenen Anwendungskompetenzen genutzt werden kann: Aus einem Set definierter Kompetenzen sollen Lehrpersonen per Selbstdeklaration die individuelle Erreichung unterschiedlicher Kompetenzen erfassen und planen. Zur Unterstützung der Lehrpersonen können für die Kompetenzen unterschiedliche Ressourcen bereitgestellt werden. Wir haben uns hier für ein Webtool entschieden. Die Anforderungen an dieses Tool sind unten aufgelistet.

**2.1.1 Projektmitglieder**

Am Projekt sind folgende Personen beteiligt:

|  |  |
| --- | --- |
| Name | Rolle |
| Ramon Camenzind | Projektmitglied |
| Timo Schlumpf | Projektmitglied |
| André Möll | Projektmitglied |
| Peter Gisler | Experte, Fachvorgesetzte Person |

**2.2 Funktionen**

*Mindestanforderungen*

Die Mindestanforderung für die Projektarbeit umfasst die Konzeption, Realisierung, Testung und Dokumentation eines Tools für die Erfassung von unterrichtsbezogenen Informatik-Anwendungskompetenzen.

Benutzerinnen und Benutzer mit spezieller Berechtigung können Anwendungskompetenzen definieren. Dabei wird jede Kompetenz einem frei wählbaren Kompetenzbereich und einer Kompetenzstufe innerhalb dieses Bereichs zugewiesen. So könnte beispielsweise die Kompetenz *Ein Inhaltsverzeichnis erstellen* zum Kompetenzbereich Textverarbeitung gehören und auf der zweiten von insgesamt drei Kompetenzstufen eingeordnet sein.

Jeder Kompetenz können beliebig viele Ressourcen (URLs) zugeordnet werden. Die Inhalte der Ressourcen unterstützen die Lehrpersonen bei der Erreichung der jeweiligen Kompetenz. Die Ressourcen werden durch die gleichen Personen verwaltet, welche die Kompetenzen und Kompetenzbereiche verwalten.

Lehrpersonen können die zu entwickelnde Applikation nutzen, um ihre persönlichen Anwendungskompetenzen zu erfassen und weiterzuentwickeln. Dazu können Sie alle Kompetenzen (inkl. Ressourcen) einsehen, die individuell vorhandenen Kompetenzen in anonymisierter Form erfassen und persistieren. Zu einem späteren Zeitpunkt soll es möglich sein, den Kompetenzzuwachs abzubilden, indem die zusätzlich erworbenen Anwendungskompetenzen ergänzt werden.

*Erweiterte Anforderungen*

Erweiterte Anforderungen ergänzen die Mindestanforderungen. Die nachfolgend aufgeführten, erweiterten Anforderungen sind keine abschliessende Aufzählung.

* **Geplante Kompetenzentwicklung:** Lehrpersonen können mit der zu entwickelnden Applikation Ihre persönliche Kompetenzentwicklung planen. Dazu werden einzelne Kompetenzen durch eine spezielle Markierung vorgemerkt. Auf diese Weise kann die Lehrperson die individuelle Weiterbildung im Bereich der IT-Anwendungskompetenzen im Rahmen der regelmässig stattfinden Mitarbeiterfördergesprächen mit der zuständigen Person planen.
* **Berufsspezifische Kompetenzen:** Berufsspezifische IT-Anwendungskompetenzen können zur Ergänzung der allgemeinen IT-Anwendungskompetenzen erfasst werden. Diese berufsspezifischen Kompetenzen werden nur auf Lehrpersonen des jeweiligen Berufes angewendet.
* **Berufsspezifische Kompetenzstufen:** Die vordefinierte Zuweisung der Kompetenzen zu einer spezifischen Kompetenzstufe kann durch berufsspezifische Überschreibungen ergänzt werden. Dabei kann sowohl eine Erhöhung als auch eine Verringerung der Kompetenzstufe für einzelne oder mehrere Berufe des GIBZ erfasst werden.
* **Auswertungsmöglichkeiten:** Berechtigte Benutzerinnen und Benutzer können eine anonymisierte Übersicht der erreichten IT-Anwendungskompetenzen einsehen. Dabei wird wahlweise der absolute und/oder relative Anteil der Kompetenzerreichung für alle Kompetenzen ausgewiesen. Mit verschiedenen Filterkriterien – beispielsweise Kompetenzbereich, Kompetenzstufe oder Berufsgruppe – kann die Darstellung Auswertung modifiziert werden.

*Eigene Anforderungen*

Zusätzlich zu den vorgegebenen Anforderungen (Mindestanforderungen & Erweiterte Anforderungen) wurden vom Projektteam eigene, ergänzende Anforderungen definiert:

**Muss**

* **Dark-Mode Switch**: Um die Applikation auch visuell ansprechend zu gestalten und dabei die Augen trotzdem etwas zu schonen, soll eine Funktion eingebaut werden, die zwischen einem hellen und einem dunklen Design der Website wechseln lässt.
* **Responsive UI Design / Layout**: Damit die App auch auf mobilen Geräten seinen Charme behält und um die Benutzerfreundlichkeit zu steigern, soll das Design / Layout responsive umgesetzt werden, d.h. es passt sich jeweils entsprechend an die Grösse des Endgeräts an.
* **Authentifizierung für API-Requests:** Auch die Sicherheit der App soll nicht missachtet werden – So ist vorgesehen, dass API-Anfragen nur mit entsprechender Verifizierung gemacht werden können. Dies dient dazu, um unerwünschte Eindringlinge von der API fernzuhalten.
* **Passwort-Hash:** Datenschutz ist ebenfalls ein wichtiges Thema. So steht es ausser Frage, dass die Passwörter der Benutzer jeweils «gehasht» werden sollen, so dass diese nicht nur für allfällige Eindringlinge, sondern auch für die Systemadministratoren uneinsehbar sind.

**Kann**

* **Login-Sperre:** Bei einer dreimalig falschen Eingabe des Passworts soll der jeweilige Besucher für eine gewisse Dauer (3’) von weiteren Loginversuchen blockiert werden. Dies hilft unter Anderem gegen Missbrauch in Form von Brute Force Angriffen.
* **Erweiterte Logfunktionen:** Die Administration darf ebenfalls nicht zu kurz kommen – so soll die Dokumentation der Aktivitäten aktiviert werden, so dass für den Admin ersichtlich ist, wann sich ein Benutzer einloggt / ausloggt. Zudem sollen dabei jeweils die IP’s des benutzen Geräts ebenfalls festgehalten werden.

*Weiteres*

* Objektorientierte Programmiersprache
* Testkonzept inkl. Beschreibung aller Testfälle
* Ausweisung & Begründung von fremdem Code
* Goddamn das huere Fachgspröch sött auno festghalte werde

**Mittel und Methoden**

Da wir vorhaben, eine Web-Applikation zu entwickeln, brauchen wir passende Programmiersprachen und Tools, um dieses Vorhaben umzusetzen.

Um die Seite zu entwickeln werden wir verschiedene Tools verwenden. Zum einen wäre da **VSCode**, ein gratis Open-Source Texteditor.

Zusätzlich werden wir **Datagrip** und **Webstorm** der Jetbrains Toolbox verwenden, um allfällige Probleme mit VSCode zu kompensieren.

Für das Frontend haben wir uns für **Svelte** entschieden. Es ist einfach zu verstehen und vielseitig einsetzbar.

Beim Backend fiel unsere Entscheidung auf **Rust**, da es eine moderne, effiziente (bezüglich der Ressourcen) und beliebte Sprache ist, die für Backend-Services viel verwendet wird.

Genauer begründen (oben)

Natürlich müssen die Daten auch irgendwo gespeichert werden. Unsere Wahl fällt auf **MongoDB**, da wir mit diesem Tool schon vertraut sind und es Erfahrungsgemäss einfach zu verwenden ist. Obwohl die Applikation sich eigentlich für eine relationale Datenbank eignen würde, haben wir uns trotzdem dagegen entschieden, weil:

- Schutz vor SQL - Injections (Mongo - Injections gibts auch, aber viel seltener)

- Responses von Queries sind direkt in JSON - Format (keine Umwandlung nötig)

- Integrierte Library für div. Sprachen zur einfachen Verwendung von Queries

- Skalierbarer als MySQL (Resourcenfreundlicher)

- Bessere Übersicht für Admins

- Schnellere Responses bei simplen Queries

**Vorkenntnisse**

Wir sind alle angehende Informatiker (Fachrichtung Applikationsentwicklung) im 3. Lehrjahr. In der Schule verwenden wir primär die Programmiersprache C#. Für dieses spezifische Projekt wird allerdings Svelte verwendet.

Ramon hat schon kleine Svelte-Kenntnisse, Rust hingegen ist für alle neu. MongoDB allerdings haben wir alle schon genutzt und verstehen wir zum grossen Teil auch recht gut. ++Genauer beschreiben

**Zeitplan**



**Arbeitsjournal**

**Datum 1: 16.09.22**

Nicht Projektstart, aber Arbeitsjournal startet hier

|  |  |
| --- | --- |
| Tätigkeiten | -Start Backend  -Start Frontend  -Erstellung Zeitplan  -Beginn Arbeitsjournal |
| Erreichte Ziele | Projektauftrag akzeptiert  Dokumentation angefangen |
| Probleme | - |
| Hilfestellung | - |
| Ausserplanmässige Arbeiten | - |
| Reflexion | Erstellung des Zeitplans war zeitaufwändiger als erwartet  Svelte sieht bisher vielversprechend aus, Rust braucht dagegen momentan noch einige Nerven |

**Datum 2: XY**

|  |  |
| --- | --- |
| Tätigkeiten |  |
| Erreichte Ziele |  |
| Probleme |  |
| Hilfestellung |  |
| Ausserplanmässige Arbeiten |  |
| Reflexion |  |

**PRAXIS**

**Einleitung**

Wir haben uns entschieden, diesen Projektauftrag in Form einer Webapplikation umzusetzen. Es folgt nun die Dokumentation der Umsetzung dieser Webapplikation.

**Ausgangslage**

Es wird eine Möglichkeit zur Kompetenzerfassung der Lehrpersonen des GIBZ gesucht. Momentan gibt es noch keine allgemeine Lösung, dies soll nun geändert werden.

+++ Umsetzung (Zusammenfassung IPERKA), Ergebnis, ausgangssituation etwas ausschmücken

**Informieren**

*Beim Informieren geht es um die Beschaffung, Ordnung und Bewertung von Informationen*

Um was geht es? Aus einem Set definierter Kompetenzen sollen Lehrpersonen per Selbstdeklaration die individuelle Erreichung unterschiedlicher Kompetenzen erfassen und planen. Zur Unterstützung der Lehrpersonen können für die Kompetenzen unterschiedliche Ressourcen bereitgestellt werden.

Was wird benötigt? Um eine Webanwendung realisieren zu können, sind Kenntnisse und ein grundlegendes Verständnis in den verwendeten Programmiersprachen vonnöten oder zumindest sehr vorteilhaft. Deshalb ist es wichtig, hier eine Lösung zu finden, die allen passt. Auch für die DB braucht es eine Entscheidung, da es dort verschiedenste Ausführungen gibt.

Wozu dient das Programm? Im Grunde soll es für Lehrpersonen des GIBZ eine (Web-)lösung geben, bei welcher Sie ihre eigenen Kompetenzen ähnlich einer To-Do-Liste nach Können bzw. Nicht-Können bewerten können.

Wie lange haben wir Zeit? Start des Projektes war am 2.9.2022 und voraussichtlicher Abschluss beziehungsweise Abgabe der Arbeit wird am 16.12.2022 stattfinden.

|  |  |
| --- | --- |
| Datum | Termin |
| 26.08.22 | Modulstart |
| 02.09.22 | Abgabe des Projektauftrags / Projektstart |
| 09.09.22 | Beginn Umsetzungsarbeiten |
| 04.11.22 | Zwischengespräche |
| 11.11.22 |
| 16.12.22 | Abgabetermin Projektarbeit / Projektbericht |
| 13.01.23 | Fachgespräche |
| 20.01.23 |
| 22.01.23 | Notenabgabe |

Maybe fix this table idk man

And add some Anforderungen

**Planen**

Bei der Wahl der Programmiersprache wurde uns die Entscheidung bedingt abgenommen, da wir zur Gruppenfindung bereits unsere präferierten Sprachen nennen mussten. Bei der Diskussion während der Teamfindung kamen einige davon auf:

Für das Frontend war die Debatte vor allem zwischen Svelte, einem JS-Framework und AngularJS. Beim Backend wollten wir grundsätzlich alle etwas Neues ausprobieren, da wir zu diesem Zeitpunkt mit C# (und teilweise auch Java) bereits einige Projekte realisiert hatten.

+++ Mockups? Klassendiagramm?

**Entscheiden**

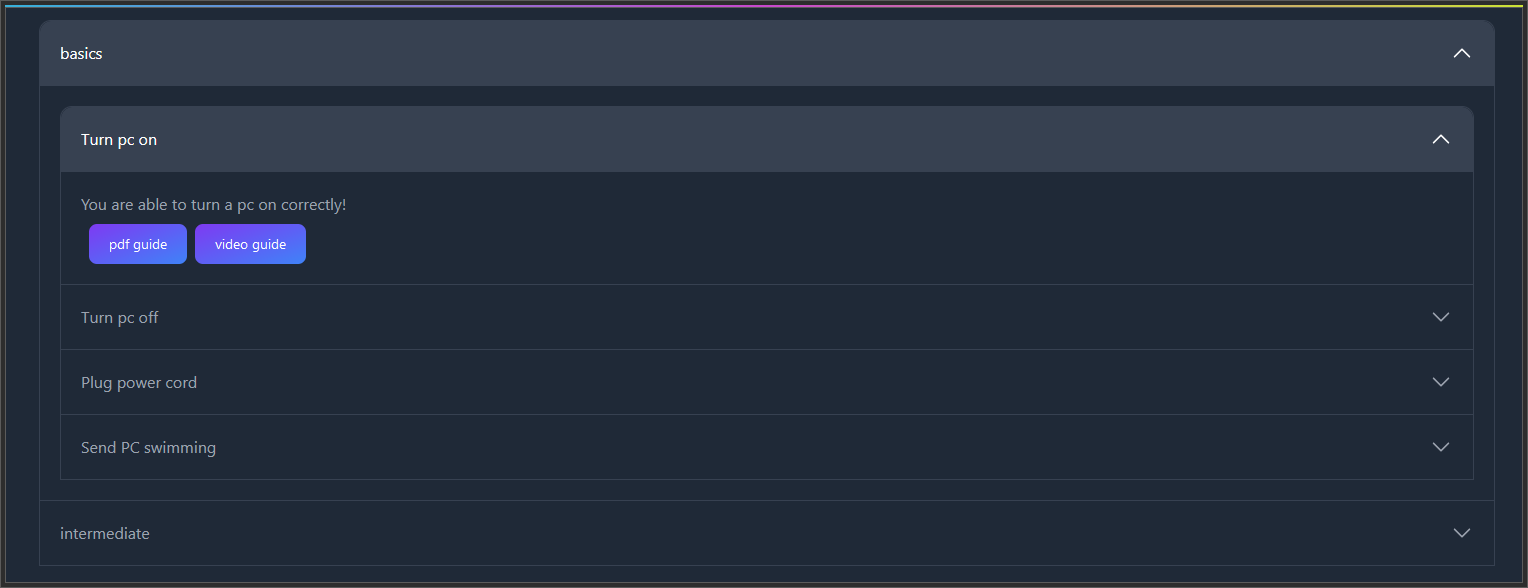
*Programmiersprachen*

Im Backend fiel der einstimmige Konsensus fiel schlussendlich auf Rust, da dies für uns alle relativ Neuland war und wir trotzdem schon mal davon gehört hatten. ++Genauere Begründung

Für das UI und generell das Frontend haben wir uns relativ schnell auf Svelte geeinigt, da es einerseits einfach zu verstehen und vielseitig einsetzbar, während andererseits Ramon bereits schon erste Erfahrungen mit Svelte gesammelt hat.

**Realisieren**

Yeah buddy, this will be rather interesting



**Kontrollieren**

**Reflexion**

**Anhang (?)**

Ein Bild, das Text, Schiefertafel enthält.

Automatisch generierte Beschreibung

