Einführung

Das Modul M295 wird in einem Projekt abgeschlossen. Dieses Projekt umfasst ein von dir zu entwickelndes REST-API sowie einer dazugehörigen Dokumentation. In der Gestaltung der Applikation bist du weitgehend frei, einzig die eingesetzten Techniken sind gemäss Unterricht einzusetzen. Dies ist deine Chance mal einen kompletten Rundumschlag mit einer App zu machen.

Ziel

Am Ende des Projektes sollen die folgenden Produkte vorliegen:

* eine komplett lauffähige REST-API Applikation (basierend auf M295)
* eine umfassende Dokumentation deines Projektes

Rahmenbedingungen

**Applikation (Was muss diese können?)**

Die Applikation soll einen von dir zu definierenden Prozess ermöglichen. Für die Daten einer relationalen Datenbank sollen CRUD-Operationen in einer REST-API zur Verfügung gestellt werden.

**Beispiele:**

1. Manga-Bibliothek wo jeder Manga zu einer Serie gehört und jeder Client festhalten kann welche Mangas er schon gelesen hat, welche auf seiner todo liste sind oder welche er noch kaufen muss. Selbiges ginge auch mit TV-Serien, Filmen, Büchern Trading Card Game Karten usw.
2. Web-Shop wo sich ein Client eine Bestellung zusammen klicken kann, da sind Artikelkategorien vorhanden sowie dazugehörige Artikel. Der Benutzer erstellt eine Bestellung und gibt hierfür eine Lieferadresse an. Beispiele digitec.ch, brack.ch, buch.ch usw.
3. Messaging-App bei der Clients sich gegenseitig in ein Notizbuch aufnehmen können, Message-Gruppen erstellen, und sich direkt oder über Gruppen Nachrichten zustellen können.

Das visuelle Design deiner Applikation ist dabei zweitrangig und gibt kaum Punkte. Viel wichtiger ist, dass deine Applikation sauber funktioniert. Stelle zuerst sicher, dass deine Applikation läuft. Danach kannst du deiner App immer noch den visuellen Feinschliff verpassen.

**Zeit**

Die Arbeit an der Web-Applikation beginnt noch während der Durchführung des Moduls. So kannst du teilweise während des Unterrichtes daran arbeiten sowie auch als Hausaufgabe. Im Unterricht stehen dir**5 x 90 Minuten**zur Verfügung um an deinem Projekt zu arbeiten.

Alle Lernenden müssen ein eigenes Projekt abgeben. Selbstverständlich ist es erlaubt sich auszutauschen und sich gegenseitig zu helfen. Hilfestellungen von Kameraden und benutze Internet-Quellen müssen in der Projektdokumentation festgehalten werden. Andernfalls kann dein Projekt als Plagiat abgestempelt werden. (Plagiat = Note 1)

Du kannst sofort nach Erhalt der Aufgabenstellung mit der Arbeit beginnen. Die Abgabe der Arbeit erfolgt wie mit der Prüfungsleitung vereinbart spätestens bei Unterrichtsende des letzten Blocks des Moduls. Eine frühere Abgabe ist natürlich erlaubt, dies aber auf eigenes Risiko. Verspätete Abgaben werden in jedem Fall als "nicht abgegeben" mit der Note 1 gewertet.

**Abgabeform**

Dein Projekt ist - soweit behandelt - als Github-Repository abzugeben oder in einem einzelnen Zip-File mit dem vorgegeben Aufbau:

LB-Projekt-M295\_Vorname-Nachname.zip   
| Projektdokumentation.pdf   
|---Backend\   
|---Ressourcen\ #optionale Dateien fürs Setup (z.B. zusätzliche DB-Skripte) 

Das Projekt sollen ohne grösseren Aufwand in eine IDE importierbar sein.

**Eingesetzte Technologien**

Die einzusetzenden Technologien orientieren sich an dem im Unterricht behandelten Themen. Dies bedeutet für dich folgendes:

Das Back-End soll mit den im Unterricht behandelten Technologien (Springboot) umgesetzt werden.

**Ablauf**

Junior Engineers neigen dazu direkt mit dem Code anzufangen, dabei kann man sich aber rasch mal verlieren. Also ist ein strukturiertes Vorgehen in Anlehnung an eine Projektmanagement-Methode (z.B. IPERKA) empfehlenswert:

**Vorbereitung**

1. Überlege dir zuerst was für eine Applikation du realisieren möchtest. Nimm dir genügend Zeit DEINE Projektidee zu entwickeln. Versuche dabei deinen Fähigkeiten Rechnung zu tragen und bleibe realistisch mit deiner Zeitplanung.
2. Halte deine Projektidee in der Projektdokumentation fest
3. Stelle deine Projektidee der Lehrperson vor. Diese wird dir sagen können ob deine Idee den Kriterien entspricht und dir ein OK für die weitere Entwicklung geben.
4. Beschreibe die einzelnen Arbeitsabläufe (z.B. neues Buch erfassen, Buch ausleihen) deiner Applikation in der Projektdokumentation mit User Stories. (max. 2h)
5. Schätze ab, wie lange du für jeden User Stories etwa benötigst und summiere die Zeiten auf. Bleibe realistisch mit deiner Planung - es muss noch Zeit für Testing und Dokumentation bleiben.
6. Entwickle das Datenmodell basierend auf den User Stories.
7. Plane die benötigten REST-API Endpunkte basierend auf dem Datenmodell und den User Stories (siehe M295-Sidequest 2B)

**Entwicklung**

Nachdem du die Vorbereitungen abgeschlossen hast, kannst du loslegen mit deinem Projekt. Mit den zuvor gemachten Überlegungen sollte dir der Einstieg leichter fallen.

Bevor du direkt mit der Implementierung beginnst, sind noch nachfolgende Vorarbeiten empfohlen. Sofern nicht anders beschrieben, sind diese aber optional und müssen NICHT dokumentiert werden.

**Back-End**

* Erstellen eines Klassendiagramms, worin die Modell-Klassen festgehalten sind (siehe M295-Sidequest 3A) (Bestandteil der Projektdokumentation)
* Erstellen einer Planung deiner Restschnittstellen, welche dein Back-End anbieten soll (siehe M295-Sidequest 2B) (Bestandteil der Projektdokumentation)
* Richte auf dem Datenbanksystem einen neuen Nutzer sowie eine Datenbank für das Projekt ein.

Jetzt ist der Zeitpunkt gekommen, deine Projektstruktur aufzusetzen. Am einfachsten orientierst du dich an der Struktur des am Schluss abzugebenden Zip-Files. Erstelle das Backend-Projekt wie im SQ1C und lege los.

Auf alle Fälle solltest du deinen Code und die Dokumentation in einem Online-CVS (Code Version-System) wie GitHub ablegen. Damit bist du gegen Datenverluste abgesichert und kannst die Entwicklung deines Codes verfolgen.

Wem kurz vor der Abgabe der Laptop verstirbt oder verlustig geht und kein aktuelles GIT-Repository vorweist muss nicht auf Gnade hoffen.

**Abschluss**

Erstelle einen Testplan für manuelle Tests, welche beim Testen der laufenden Applikation, zur Beurteilung derer korrekten Funktion, durchgeführt werden sollten.

Verfasse eine Installationsanleitung worin alle für den Betrieb notwendigen Schritte festgehalten sind. Damit soll eine Lehrperson deine Applikation ohne Probleme zum Laufen bringen können.

Du kannst die Dokumentation mit Word, Markdown o.ä. erstellen erstellen. Gib deine Dokumentation jedoch als PDF-Datei ab

Erstelle ein Zip-File gemäss Abschnitt: Abgabeform und gib dieses ab

Detailanforderungen des Projektes

**Projektdokumentation**

Die folgenden Bestandteile müssen nebst einem Titelblatt und dem Inhaltsverzeichnis in der Projektdokumentation vorhanden sein:

|  |  |
| --- | --- |
| **Anforderung** | **Beschreibung** |
| Projektidee | Beschreibe hier auf maximal einer halben A4-Seite deine Projektidee. Stell es dir als elevator-pitch vor wo du jemandem in kurzer Zeit dein Projekt “verkaufen” möchtest. So, dass nicht zuerst die gesamte Dokumentation durchgelesen werden muss. |
| Anforderungskatalog | Identifiziere die Kernaufgaben in deinem System. |
| Klassendiagramm | Halte hier fest welche Modell-Klassen in deiner Applikation vorkommen und in welcher Beziehung diese zueinander stehen. Dokumentiere die Klassen mit ihren Attributen. (draw.io empfohlen) |
| REST-Schnittstellen | Hier dokumentierst du die benutzten REST-Schnittstellen. Dokumentiere was diese Schnittstellen machen und und insbesondere den Aufbau deiner Datentypen. |
| Testplan | Die lauffähige Applikation muss manuell getestet werden. Wie dies passiert definierst du im Testplan. Beschränke dich dabei auf die wichtigste Funktionalität. Die Test-Ergebnisse dokumentierst du in einem weiteren Kapitel. |
| Installationsanleitung | Für die Inbetriebnahme durch die Lehrperson soll eine kurze Installationsanleitung verfasst sein. |
| Hilfestellungen | Hier gehören alle Hilfestellungen von Mitlernenden, Dozenten und Internetrecherchen hin. |

Bedenke, dass die Projektdokumentation "lebt". Am Anfang dient diese dir als Starthilfe für die Entwicklung, danach dient sie als Dokumentation des gemachten. Wenn du also etwas am Ablauf, der Schnittstellen usw. veränderst, trage dies auch in der Dokumentation nach. Die Dokumentation sollte immer den aktuellen Zustand widerspiegeln.

**Back-End Applikation**

Folgende Anforderungen sollten in deinem Back-End Projekt berücksichtigt werden:

|  |  |
| --- | --- |
| **Anforderung** | **Beschreibung** |
| Maven | Das Back-End Projekt sollte mit Maven entwickelt werden. Konfiguriere also entsprechend das POM-File so, dass bei einem Import in eine beliebige IDE dein Projekt nach wie vor erfolgreich gebaut werden kann. |
| Validierung | Stelle sicher, dass deine Daten im Back-End Projekt validiert werden, bevor sie an die Datenbank geschickt werden. Bei Validierungsproblemen werden wohlgeformte Fehlermeldungen an den Client zurückgegeben. |
| Code-Dokumentation | Dokumentiere alle deine Java-Klassen mittels entsprechender JavaDoc-Kommentare. |
| Unit-Tests | Automatisierte Unit-Tests sollen in deinem Projekt das Funktionieren der wichtigsten Komponenten sicherstellen. |

Bewertungskriterien

Das Projekt wird als ein ganzes Bewertet.

Die Berechnung der Note erfolgt dabei, wie gewohnt, linear.

(ErreichtePunkte \* 5 / maximal Punkte) + 1

Bei jeder Position können 2 Punkte erreicht werden. Aufgeschlüsselt werden die Punkte:

* 0 Punkte (nicht erfüllt / fehlt)
* 1 Punkt (teilweise erfüllt)
* 2 Punkte (vollständig erfüllt)

**A: Allgemein - 8 Punkte**

* Die Komplexität des individuellen Projektes ist angemessen (mindestens so komplex wie WISS-Quiz)
* Arbeiten sind weitestgehend nach Vorgaben vollständig
* Abgabe enthält alle verlangten Dateien und ermöglicht den Betrieb
* Positiver Gesamteindruck des Projektes und der Dokumentation

**B: Projektdokumentation - 18 Punkte**

* Die Dokumentation wurde als PDF-Datei oder Github-Markdown abgegeben
* Grundidee des Projekts ist enthalten und gibt einen Grobüberblick welcher Prozess damit abgebildet wird
* Anforderungsanalyse wurde durchgeführt und dokumentiert (mindestens drei User Stories mit Akzeptanzkriterien)
* Diagramm der Modell-Komponenten ist enthalten und korrekt
* Storyboard der Applikation ist enthalten und gibt den Ablauf und Design der Applikation wieder
* Testplan für Testen ist vorhanden und beinhaltet 5 sinnvolle Testfälle
* Durchführung des Testplans wurde dokumentiert
* Installationsanleitung ist vorhanden und korrekt
* Hilfestellungen wurden massvoll eingesetzt und dokumentiert

**C: Implementierung REST-API - 24 Punkte**

* Das Projekt lässt sich ohne Anpassungen in eine IDE importieren.
* Das Projekt lässt sich ohne Kompilierfehler bauen.
* Die Java-Klassen beinhalten sinnvolle und korrekte JavaDoc Kommentare (ohne getter/setter).
* Die Applikation stürzt nicht ab, Exceptions werden sauber gehandelt.
* Das REST-API bietet geeignete Endpoints für das Laden von Daten an.
* Das REST-API bietet geeignete Endpoints für das Erfassen von Daten an.
* Das REST-API bietet geeignete Endpoints für das Editieren von Daten an.
* Das REST-API bietet geeignete Endpoints für das Löschen von Daten an.
* Daten werden sinnvoll validiert.
* Daten werden in einer Datenbank persistent abgelegt.
* Unit-Tests sind vorhanden und beinhalten 5 sinnvolle Testfälle.
* Unit-Tests sind ausführbar

**D: Fachgespräch - 20 Punkte**