Modulprüfung

Modul 223

Multi-User-Applikationen objektorientiert realisieren

Praktische Umsetzungsarbeit (LBV Modul 223-2)

Prüfungsnummer **MP223-NYP-2018**
Leistungsbeurteilungsvorgabe: **Modulidentifikation zum Modul 223 gemäss MBK**

Durch den/die Kandidaten/in auszufüllen

Vorname und Name	
Lehrbetrieb	
Klasse / Kursnummer	
Prüfungsdatum	
Prüfungsort	Noser Young Professionals AG, Zürich

Durch die Prüfungsleitung auszufüllen

Vorname und Name	
Korrekturdatum	
Erreichte Punktzahl	
Note	
Unterschrift	

Prüfungsform	Praktisch, handlungsorientiert, Gruppenarbeit (3 Personen) im Rahmen eines Projekts
Prüfungsdauer	Ab dem ersten Kurstag bis Kursende (mit begleitendem Unterricht), d.h. 6 Tage, während 2 Wochen
Maximale Punktezahl	100
Prüfungsunterlagen	Vorliegende Prüfung (insgesamt 11 Seiten)
Erlaubte Hilfsmittel	Informationen aus dem Internet, den ausgeteilten Kursunterlagen und Fachbüchern. Eingesetzte Codesequenzen von externen Quellen (z.B. Internet, Fachbüchern, Kollegen) müssen im Sourcecode durch entsprechende Kommentare kenntlich gemacht und dokumentiert werden, ansonsten gelten sie als Plagiate.
Arbeitsplatz	PC-Arbeitsplatz mit Java Entwicklungsumgebung (Spring Tool Suite) und MySQL-Server. Office-Tools.
Prüfungsresultate	Aufgabenstellung Allfällige Skizzen auf Papier Programm und Datenbank Sourcecode, Dokumentation und DB-Export (SQL-Datei) im GIT-Repository eingecheckt.
Notenberechnung	Zu den einzelnen Aufgaben sind die erreichbaren Punkte angegeben, es werden nur ganze Punkte vergeben. Die Note wird mit folgender Formel berechnet und auf halbe Notenwerte gerundet. $$Note = \frac{erreichte\ Punkte}{maximale\ Punkte} * 5 + 1$$

Werden die Prüfungsbestimmungen verletzt, z.B. unerlaubte Hilfe, kopieren von Code aus dem Internet oder von Kollegen ohne Quellenangabe (Plagiate), führt dies für den Kandidaten zum Abbruch der Prüfung und zu einer Prüfungsnote von 1.0

Bewertungskriterien

		Punkte pro Aufgabe						
Nr.	**Bewertungskriterium**	**1**	**2**	**3**	**4**	**5**	**6**	**Total**
1	Analyse der Anforderungen hinsichtlich Mehrbenutzeranforderungen	5	10					15
2	Benutzerinterface für mehrere Benutzer implementieren			5			15	20
3	Mehrbenutzeranforderungen und deren Fehlerbehandlung implementieren				5		20	25
4	Transaktionen und deren Fehlerbehandlung implementieren				5		15	20
5	Testspezifikation schreiben und Tests durchführen					20		20
	Summen	**5**	**10**	**5**	**10**	**20**	**50**	**100**

Wir wünschen Ihnen viel Erfolg!

Einführung und Hinweise zum Projekt

Projektauftrag

Ihr Projektauftrag können Sie der Beilage „Projektauftrag“ entnehmen.

Rahmenbedingungen

- Das Projekt wird in einer Gruppe à 3 Personen durchgeführt. In Programmcode muss anhand der Javadoc-History ersichtlich sein, welcher Lernende, welcher Teil des Codes umgesetzt hat. Auch in den abgegebenen Dokumenten ist eine möglichst ausführliche Änderungskontrolle zu führen.
- Alle Dokumente und der Sourcecode müssen in einem GIT-Repository eingecheckt werden.
- Entwicklung der RESTful-API (Backend) muss in Java mit Spring Boot erfolgen.
- Umsetzung der Datenbank mit MySQL.
- Frontend-Technologie kann jede Gruppe selber auswählen.
- Der Projektname soll in der Anwendung sichtbar sein.

1. Analyse des Umfelds

1.1 Bewertungskriterien

100% Analyse der Anforderungen hinsichtlich Mehrbenutzeranforderungen

1.2 Handlungsziel

Einschätzen, ob eine Datenbank die Anforderungen der Multi-User-Fähigkeit erfüllt und allfällige Anpassungen dokumentieren

1.3 Vorgehen

Erstellen Sie ein Domänenmodel ihrer Domäne (ganze Domäne, in welcher sich die Applikation befindet). Setzten Sie dieses als UML Klassendiagramm um. Das Diagramm muss die Domänenklassen mit den wichtigsten Attributen (nur Name) und die Beziehung (mit Multiplizität und Bezeichnung) beinhalten. Es soll keine Datentypen, Sichtbarkeiten und Methoden beinhalten!

Wichtig: Ein Domänenmodell ist kein ERD und kein Design Klassendiagramm! Es stellt eine konzeptuelle Repräsentation der Dinge im Fachbereich dar und kein Konzept von Softwareobjekten.

Tipp: Erstellen Sie das Domänenmodell in einem ersten Schritt nur aus den Klassen und Beziehungen, jedoch ohne Attribute und Multiplizitäten.

1.4 Verlangte Ergebnisse

- Domänenmodell als UML-Klassendiagramm (Ganze Domäne, alle Anforderungen)

1.5 Massstab

Thema	Detailpunkte	Max. Pkt.	Err. Pkt.
Domänenmodell	Ist ein Domänenmodell vorhanden.	1	
	Ist das Domänenmodell vollständig, d.h. sind alle Domänenklassen, die wichtigsten Attribute und die Beziehungen (Multiplizität + Bezeichnung) vorhanden.	4	
	Total	**5**	

2. Analyse der Anforderungen

2.1 Bewertungskriterium

100% Analyse der Anforderungen hinsichtlich Mehrbenutzeranforderungen

2.2 Handlungsziel

Einschätzen, ob eine Datenbank die Anforderungen der Multi-User-Fähigkeit erfüllt und allfällige Anpassungen dokumentieren

2.3 Vorgehen

- Erstellen Sie ein UML Use-Case-Diagramm für alle Pflicht- und optionalen Anforderungen (Use-Cases und beteiligte Akteure).
- Beschreiben Sie die Akteure jeweils in 2 – 5 Sätzen.
- Erstellen Sie die Use-Case-Beschreibungen für 2 Use Cases der Pflicht-Anforderungen (Hauptablauf + Alternative Flows) gemäss folgender Standardschablone:
    - Name und ID des Use-Case
    - Kurzbeschreibung
    - Vorbedingungen
    - Akteur (Primary)
    - Akteur (Secondary)
    - Hauptablauf
    - Nachbedingungen
    - Alternative Flows

2.4 Verlangte Ergebnisse

Siehe unter Massstab.

2.5 Massstab

Thema	Detailpunkte	Max. Pkt.	Err. Pkt.
UML Use-Case-Diagramm	Ist ein Use-Case-Diagramm nach UML Standard vorhanden?	1	
	Sind alle Akteure definiert und benannt?	1	
	Sind alle Use-Cases der Pflicht- und optionalen Anforderungen im Diagramm vorhanden und den Akteuren zugewiesen.	2	
Use-Cases	Sind mindestens zwei essenzielle Use-Cases der Pflichtanforderungen mit strukturiertem Text beschrieben und gemäss obigen Punkten definiert?	6	
	Total	**10**	

3. User Interface Design

3.1 Bewertungskriterien

100% Benutzerinterface für mehrere Benutzer implementieren

Handlungsziele

Applikation entwerfen und mittels Transaktionen Multi- User-Fähigkeit sicherstellen.

User Interfaces, Datenbankanpassungen und Transaktionen implementieren.

3.2 Vorgehen

Erstellen Sie ein Storyboard der Android-App, welche Sie erweitern werden. Halten Sie den aktuellen Stand der App fest. Dabei handelt es sich nicht mehr um einen Mockup, sondern um eine Dokumentation der bestehenden App als Storyboard. Der Aufbau des GUIs, die verwendeten Elemente, sowie die Abläufe müssen sichtbar sein (Erfolgs- und Fehlerfälle). Berücksichtigen Sie dabei die Akteure, welche im GUI verwendet werden.

3.3 Verlangte Ergebnisse

Ansichten der bestehenden App sind als Storyboard dokumentiert.

3.4 Massstab

Thema	Detailpunkte	Max. Pkt.	Err. Pkt.
GUI Design	GUI Designs sind dokumentiert im Rahmen eines Storyboards (Scan der Skizzen auf Papier oder elektronischer Mockup)	4	
	GUI Designs berücksichtigen die unterschiedlichen Akteure	1	
	Total	**5**	

4. Datenhaltung spezifizieren

4.1 Bewertungskriterien

50% Mehrbenutzeranforderungen und deren Fehlerbehandlung implementieren

50% Transaktionen und deren Fehlerbehandlung implementieren

4.2 Handlungsziel

Applikation entwerfen und mittels Transaktionen Multi- User-Fähigkeit sicherstellen.

4.3 Vorgehen

Erstellen Sie ein Entity-Relationship-Diagramm mit den erforderlichen Entitäten und Beziehungen, inkl. den Multiplizitäten für die Datenhaltung in einem RDBMS. Das ERD muss nur die Entitäten der Pflicht-Anforderungen beinhalten. Es muss jedoch so aufgebaut sein, dass eine einfache Erweiterung für die Umsetzung der optionalen Anforderungen möglich ist.

4.4 Verlangte Ergebnisse

- Entity-Relationship-Diagramm mit allen für das Projekt erforderlichen Entitäten, deren Eigenschaften und Beziehungen (Nur Pflicht-Anforderungen)

4.5 Massstab

Thema	Detailpunkte	Max. Pkt.	Err. Pkt.
Datenhaltung	Ist ein Entity-Relationship-Diagramm vorhanden?	1	
	Ist das ERM vollständig, d.h. erforderliche Entitäten, Beziehungen mit Multiplizitäten vorhanden und korrekt?	3	
	Ist die Datenhaltung für die benötigten Entitäten (Pflicht-Anforderungen) vollständig spezifiziert, d.h. Tabellenname, Felder, Datentypen, Grösse/Länge?	3	
	Sind die Entitäten, Attribute, Beziehungen, Multiplizitäten, Datentypen, Grössen/Länge sinnvoll (den Anforderungen angepasst) gewählt.	3	
	Total	**10**	

5. Testing

5.1 Bewertungskriterium

100% Testspezifikation schreiben und Tests durchführen

5.2 Handlungsziele

Testspezifikation für funktionale und nicht-funktionale Aspekte der Multi-User-Fähigkeit definieren, Applikation testen und Tests protokollieren.

5.3 Vorgehen

Aufgrund der zeitlichen Verhältnisse wird auf eine vollständige durch Tests abgedeckte Qualitätsprüfung verzichtet. Es soll jedoch ein Teilbereich getestet werden.

Definieren Sie ein Testkonzept mit den im Unterricht behandelten Punkten.

Erstellen Sie JUnit-Tests, welche mindestens zwei Funktionen Ihrer Anwendung testen.

Spezifizieren Sie die Testfälle der Systemtests für die beiden beschriebenen Use-Cases der Pflicht-Anforderungen. Führen Sie die Systemtests aus und protokollieren Sie die Ergebnisse der Tests in ein Testprotokoll.

5.4 Verlangte Ergebnisse

Lauffähige und erfolgreich ablaufende Unit Tests mit Projekt der Entwicklungsumgebung. Dokumentierte Ergebnisse (Screenshot).

Testkonzept und zwei Testfälle zum Systemtest.

Testprotokoll zu den Systemtests.

5.5 Massstab

Thema	Detailpunkte	Max. Pkt.	Err. Pkt.
Testkonzept	Testkonzept ist vorhanden und beinhaltet die im Unterricht behandelten Punkte.	3	
Unit-Tests	Es ist mindestens 1 Funktion über Unit-Tests abgedeckt und diese wurden korrekt und erfolgreich getestet (Screenshot)	5	
Systemtest	Die zwei Testfälle sind mit klar definiertem Sinn und Zweck, Input und Output definiert (wie im Unterricht behandelt).	8	
	Die zwei Testfälle wurden ausgeführt und protokolliert	4	
	Total	**20**	

6. Implementation

6.1 Bewertungskriterien

Benutzerinterface für mehrere Benutzer implementieren

Mehrbenutzeranforderungen und deren Fehlerbehandlung implementieren

Transaktionen und deren Fehlerbehandlung implementieren

6.2 Handlungsziele

Applikation entwerfen und mittels Transaktionen Multi- User-Fähigkeit sicherstellen

User Interfaces, Datenbankanpassungen und Transaktionen implementieren.

Transaktionen und deren Fehlerbehandlung implementieren

6.3 Vorgehen

Verwenden Sie ein GIT-Repository für die Versionskontrolle und checken Sie mindestens täglich Ihren möglichst lauffähigen Code mit einem sprechenden Kommentar ein. Der Kommentar soll den aktuellen Stand der Entwicklung kurz zusammenfassen.

Verwenden Sie zur Entwicklung entweder die Spring Tool Suite (STS), Eclipse oder IntelliJ. Der Unterricht und die Beispiele wurden mit der Spring Tool Suite erstellt.

Erstellen Sie mit Spring Boot eine RESTful API, welche die Daten in einer MySQL-Datenbank persistiert. Verwenden Sie JPA für die Datenbankanbindung und das Mapping von Tabellen auf die Java-Objekte. Die RESTful-API soll mindestens folgende Schichten beinhalten: Persistence Layer (Datenbank, JPA), Data Layer (POJO) und Logik Layer (Business Logic).

Es müssen DB-Transaktionen verwendet werden.

Kommentieren Sie den Code der RESTful-API mit JavaDoc. Dazu gehört auch eine ausführliche History, welcher Studierende wann, was, gemacht hat.

Laden Sie ihre Spring Boot RESTful-API auf Heroku hoch.

App-Erweiterung:

Binden Sie die RESTful-API an die im ÜK 335 erstellte Android-App an. Die App muss aus Zeitgründen nicht kommentiert werden.

Erstellen Sie ein Klassendiagramm der realisierten RESTful-API (darf auch generiert werden).

6.4 Verlangte Ergebnisse

GIT-Repository mit gesamter Dokumentation, Sourcecode aller Anwendungen und einem SQL-Export Ihrer Datenbank.

Lauffähige Anwendung, wobei mindestens die Use Cases der Pflicht-Anforderungen umgesetzt sein müssen.

Klassendiagramm der realisierten Klassen der RESTful-API (darf auch generiert werden).

6.5 Massstab

Thema	Detailpunkte	Max. Pkt.	Err. Pkt.
Versionsverwaltung	Sourcecode ist in einem GIT Repository täglich eingecheckt worden, Kommentar ist sprechend	3	
RESTful-API:	OR Mapping wurde mit JPA realisiert.	2	
	JPA wurde korrekt angewendet (Dependencies, Annotations, Repositories, Dependency Injection, Beziehungen, Primär-/Fremdschlüssel)	6	
	DB-Transaktionen wurden eingesetzt und deren Fehlerbehandlung korrekt implementiert.	4	
Client (App)	REST-Requests funktionieren und die erhaltenen Daten werden im GUI angezeigt.	5	
	GUI ist voll funktionsfähig.	6	
RESTful-API (Backend)	Ist gemäss Use Cases Pflichtanforderungen voll funktionsfähig.	9	
	Code Qualität entspricht den Java und Spring Best Practises (Variablenbenennung, Formatierung, Annotations, Sichtbarkeiten, Clean Code, Vererbung, etc.)	3	
	Beinhaltet mindestens die 3 oben genannten Schichten.	3	
Code-Dokumentation	Alle Klassen der RESTful API sind mit JavaDoc dokumentiert. Dazu gehört die Beschreibung der Klasse, die Version der Klasse, sowie die History der Änderungen (Version, Datum, Person, Was wurde geändert).	2	
	Alle wichtigen Methoden (ohne getter und setter) und alle Konstanten der RESTful API sind mit JavaDoc dokumentiert.	2	
Klassendiagramm	UML-Klassendiagramm der realisierten Klassen der RESTful API ist vorhanden. Attribute, Operationen, Beziehungen, Datentypen, Sichtbarkeiten sind vorhanden und korrekt.	5	
	Total	**50**	

Gratuliere, Sie haben alle Aufgaben gelöst!

7. Anhang

Hinweis zur Punkteaufteilung auf die Bewertungskriterien:

Thema	Detailpunkte	Kriterium 3	Kriterium 4	Kriterium 5
Versionsverwaltung	Sourcecode ist in einem GIT Repository täglich eingecheckt worden, Kommentar ist sprechend	1	1	1
RESTful-API:	OR Mapping wurde mit JPA realisiert.			2
	JPA wurde korrekt angewendet (Dependencies, Annotations, Repositories, Dependency Injection, Beziehungen, Primär-/Fremdschlüssel)			6
	DB-Transaktionen wurden eingesetzt und deren Fehlerbehandlung korrekt implementiert.			4
Client (App)	GUI ist voll funktionsfähig.	6		
	REST-Requests funktionieren und die erhaltenen Daten werden im GUI angezeigt.	5		
RESTful-API (Backend)	Ist gemäss Use Cases Pflichtanforderungen voll funktionsfähig.	2	7	
	Code Qualität entspricht den Java und Spring Best Practises (Variablenbenennung, Formatierung, Annotations, Sichtbarkeiten, Clean Code, Vererbung, etc.)		3	
	Beinhaltet mindestens die 3 oben genannten Schichten.		3	
Code-Dokumentation	Alle Klassen der RESTful API sind mit JavaDoc dokumentiert. Dazu gehört die Beschreibung der Klasse, die Version der Klasse, sowie die History der Änderungen (Version, Datum, Person, Was wurde geändert).		2	
	Alle wichtigen Methoden (ohne getter und setter) und alle Konstanten der RESTful API sind mit JavaDoc dokumentiert.		2	
Klassendiagramm	UML-Klassendiagramm der realisierten Klassen der RESTful API ist vorhanden. Attribute, Operationen, Beziehungen, Datentypen, Sichtbarkeiten sind vorhanden und korrekt.	1	2	2