Bosses Klausur Anwendungsentwicklung, Oberstufe, 1. Hj.

# Aufgabe 1 – Projektmanagement & graphische Benutzeroberflächen (8 Pkt.)

## 1a

**Welche Aussage beschreibt am besten den Unterschied zwischen User Interface (UI) und User Experience (UX) im Kontext von graphischen Benutzeroberflächen?**

UI bezieht sich auf die Struktur und Funktionalität eines Produkts, während UX sich mit der visuellen Gestaltung und Ästhetik beschäftigt.

UI umfasst alle interaktiven Elemente, die ein Benutzer sehen und verwenden kann, während UX das Gesamtgefühl und die Zufriedenheit des Benutzers mit der Verwendung des Produkts beschreibt.

UI ist ausschließlich auf mobile Anwendungen bezogen, während UX nur für Desktop-Anwendungen gilt.

UI konzentriert sich auf die Interaktion zwischen Softwareentwicklern, während UX die Interaktion zwischen Projektmanagern fokussiert.

## 1b

**Welche Aussage zu den Projektbeteiligten trifft nicht zu?**

Der Auftraggeber initiiert den Projektauftrag und entscheidet über Abbruch oder Fortführung des Projekts.

Der Lenkungsausschuss trifft in allen Projektphasen Grundsatzentscheidungen.

Stakeholder sind nur externe Personen, die vom Projekt betroffen sind.

Das Projektteam bearbeitet unter der Leitung der Projektleitung den Projektauftrag

## 1c

. **Welche Elemente der Projektorganisation nach DIN 69901 gehören nicht zu den klassischen Projektinstanzen?**

Lenkungsausschuss

Gesamtprojektleitung

Teilprojektleitungen

Projektbeteiligte

## 1d

**Welche Aussage zur Projektleitung trifft zu?**

Die Projektleitung legt die Regeln und Rollen im Projekt fest und bestimmt maßgeblich die Teambildung.

Die Projektleitung ist für die Durchführung des Projekts verantwortlich.

Die Projektleitung berichtet direkt an den Lenkungsausschuss.

Alle Aussagen sind richtig.

# Aufgabe 2 - Entity Relationship Model (10 Pkt.)

## ER-Modell für eine Bücherei

**Aufgabenstellung:**

Sie sind als Fachinformatiker in einer Stadtbücherei tätig. Die Leitung hat Sie beauftragt, ein ER-Modell für die Verwaltung der Bibliotheksbestände und -ausleihen zu erstellen, da das System bisher komplett auf Papier verwaltet wird. Nun soll es digitalisiert werden.

Folgende Anforderungen sollen in Ihrem ER-Modell berücksichtigt werden:

1. **Nutzer**: Jeder Nutzer hat eine eindeutige Mitgliedsnummer, einen Namen und eine Adresse.
2. **Bücher**: Jedes Buch hat eine ISBN, einen Titel und einen Autor.
3. **Ausleihen**: Für jede Ausleihe eines Buches wird ein Ausleihvorgang erstellt. Zu einem Ausleihvorgang gehören die Ausleih-Nr., das Ausleih- und Rückgabedatum sowie der ausleihende Nutzer.
4. **Mitarbeiter**: In der Bibliothek arbeiten Bibliothekare. Jeder Mitarbeiter hat eine Personalnummer und einen Namen.
5. **Zuordnungen**: Jede Ausleihe wird von einem Mitarbeiter bearbeitet.

Erstellen Sie ein ER-Modell, das die oben genannten Anforderungen abbildet. Achten Sie dabei auf die korrekten Kardinalitäten zwischen den Entitäten.

*Hinweis: Verwenden Sie für die grafische Darstellung des ER-Modells die bekannten Konventionen, achten Sie auch darauf, starke und schwache Entitäten zu unterscheiden.*

Lösung als Bild hier einfügen:

# Aufgabe 3 - Datenbanken und SQL (10 Pkt.)

## Analyse von Verkaufsdaten

Sie haben Zugriff auf eine Datenbank eines Einzelhandelsunternehmens mit folgenden Tabellen:

**Tabelle: Produkte**

- produkt\_id

- produktname

- kategorie

- preis

**Tabelle: Verkäufe**

- verkaufs\_id

- produkt\_id

- kunde\_id

- verkaufsdatum

- menge

- umsatz

**Tabelle: Kunden**

- kunde\_id

- vorname

- name

- adresse

- email

**Ihre Aufgabe ist es, verschiedene Analysen zu den Verkaufsdaten durchzuführen und die Ergebnisse in SQL-Abfragen darzustellen.**

**1. Umsatz pro Produktkategorie**

- Erstellen Sie eine Abfrage, die den Gesamtumsatz für jede Produktkategorie ausgibt.

- Sortieren Sie das Ergebnis absteigend nach dem Gesamtumsatz

Selectfrom Produkte

foreach(Produkte.kategorie) sum(Produkte.preis) oder by desc

Joa, naja, ein Ansatz 😉

**2. Topprodukte nach Umsatz**

- Schreiben Sie eine Abfrage, die die 5 umsatzstärksten Produkte ausgibt.

- Zeigen Sie dafür den Produktnamen, den Preis und den Gesamtumsatz an.

Ja SQL ist meine große Schwäche. Gucke ich mir bis zur Prüfung nochmal an. Habe ich bis jetzt gott sei dank nur in ein paar Unterrichtsstunden gebraucht. Ist ok, hast ja noch etwas Zeit!

*Hinweise:*

* *Verwenden Sie geeignete JOIN-Befehle, um die Tabellen miteinander zu verknüpfen.*
* *Nutzen Sie Aggregatfunktionen wie SUM(), AVG(), COUNT() etc., um die gewünschten Berechnungen durchzuführen.*
* *Verwenden Sie Aliasnamen für eine übersichtliche Darstellung der Ergebnisse.*

# Aufgabe 4 – Programmierung (15 +3 Pkt.)

## Kundenverwaltungssystem

**Hintergrund**:

Sie arbeiten an einem Kundenverwaltungssystem für ein kleines Unternehmen. Das System soll in der Lage sein, Kundeninformationen zu speichern und eine einfache Suche nach Kunden durchzuführen.

**Aufgabenstellung**:

Implementieren Sie einen Pseudocode (oder Code in einer bekannten Programmiersprache) für die folgenden Anforderungen:

1. **Datenstruktur**:

Erstellen Sie eine Datenstruktur `Kunde`, die die folgenden Attribute enthält:

- `KundenID` (eindeutige Identifikationsnummer)

- `Name` (Name des Kunden)

- `Email` (E-Mail-Adresse des Kunden)

Public Class Kunde()

{

Public GUID KundenID { get; init;}

Public String Name { get; init;}

Public string KundenEMail { get; set;}

}

PS: Das „init;“ ist wie ein set; mit dem unterschied, das es nur einmal am Anfang gesetzt werden darf. Dadurch können PK und fixe Daten nicht im Nachhinein überschrieben werden.

2. **Kundenliste**:

Erstellen Sie eine Liste, um mehrere Kundenobjekte zu speichern.

Public List<Kunde> Kundendaten = new List<Kunde>();

3. **Kunde hinzufügen:**

Schreiben Sie eine Funktion `KundeHinzufuegen(KundenListe, KundenID, Name, Email)`, die einen neuen Kunden zur Liste hinzufügt. Überprüfen Sie, ob die `KundenID` bereits existiert. Wenn ja, geben Sie eine Fehlermeldung aus.

Public ValueTask<Kunde> neuenKundenErstellen (string Name, string Email)

{

Kunde testKunde = new Kunde()

{

testKunde.KundenID = new Guid(),

testKunde.Name = Name,

testKunde.KundenEMail = Email

};

this.listenService.Kundendaten.Add(testKunde);

}

PS: Das hinzufügen funktioniert hier nur, wenn die Liste in der gleichen Klasse vorhanden ist. Am besten benutzt man aber wenn man eine DB dahinter hat, gleich die eigentliche Tabelle wie eine Liste und fügt den Kunden direkt zur DB hinzu. Sonst kann es sein, dass es Listen mit verschieden Zuständen gibt und die Integrität der Daten gefährdet wird. Sollte dies nicht möglich sein muss die Liste über einen Service erreichbar gemacht werden, wodurch sie sich so verhält, als wenn sie in der Klasse wäre. Ich gehe davon aus das die hier der Fall ist. Der Service, der mir die Liste bereit stellt heißt ListenService. Dieser Service verwaltet die Liste und sorgt dafüür das der grade beschriebene Fall nicht eintritt. Um auf den Service zugreifen zu können, wie ich es grade tue, muss er in C# einmal in der Programm.cs registriert werden und in jeder Klasse, die ihn nutzen soll injected werden. Das injecten sähe wie folgt aus:

Ok, geht klar, aber verhinderst Du damit auch schon, dass der Kunde doppelt eingetragen würde? Das sollte ja verhindert werden!

[Inject]

Public ListenService listenService;

4. **Kunde suchen:**

Schreiben Sie eine Funktion `KundeSuchen(KundenListe, KundenID)`, die einen Kunden anhand seiner `KundenID` sucht und die Informationen des Kunden zurückgibt. Wenn der Kunde nicht gefunden wird, geben Sie eine entsprechende Nachricht aus.

Public ValueTask<Kunden> KundenSuche( Guid KundenID)

{

try

{

return this.listenService.Kundendaten.Where( () => x.KundenID == this.KundenID);

}

Catch( NullReferenceExeption ex)

{

Return $“ Es gibt keinen Kunden mit der angegebenen ID. {ex}“;

}

}

5. **Kunde entfernen:**

Schreiben Sie eine Funktion `KundeEntfernen(KundenListe, KundenID)`, die einen Kunden anhand seiner `KundenID` aus der Liste entfernt. Geben Sie eine Bestätigung aus, wenn der Kunde erfolgreich entfernt wurde, oder eine Fehlermeldung, wenn der Kunde nicht gefunden wurde.

Public ValueTask KundeEntfernen(KundenID)

{

alterKunde = this.listenService.Kundendaten.Where( () => x.KundenID == this.KundenID);

this.listenService.Kundendaten.Remove(alterKunde);

return;

}

6. **Zusatzfrage (+ 3 Pkt.)**:

Welche bekannte Datenstruktur ist besonders gut geeignet für die Realisierung dieser Aufgabe und warum?

Besonders gut für die Anforderungen dieser Aufgabe eignen sich Listen in objektorientierten Sprachen, da diese sich sehr ähnlich zu SQL basierten Datenbanken verhalten. Der größte Unterschied ist, dass die Listen nicht persistent gespeichert werden können und je nach Programmiersprache noch mehr mögliche Methoden haben als DB-Tabellen. Die Liste wird durch die Klassen ergänzt. Die Klassen stellen die Spalten der DB Tabelle da und die Liste mit ihren verschieden Einträgen die Zeilen. Da das so gut funktioniert gibt es zumindes bei C# und Pyhton (vermutlich bei mehr aber da weiß ich es nicht genau) sogenannte ORM´s, die wie eine Art Übersetzer zwischen Code und SQL funktioniert. In C# ist da Entity Framework Core, das sehr zu empfehlen ist, da es einem wirklich jede Menge Arbeit abnimmt, wenn man sich einmal eingearbeitet hat.

(Ps: EFCore ist aber auch der Grund warum ist Aufgabe 3 verkackt habe, einfach weil ich im Betrieb nie SQL benutzt habe.) 😉

OK, überzeugt mich – Dictionary oder HashMap hätten mir sonst auch gereicht (Zugriffszeiten O(1) für viele Aufgaben)

*Hier ist ein einfaches Beispiel, wie die Funktionen verwendet werden können sollen:*

*// Hauptprogramm*

*KundenListe = []*

*KundeHinzufuegen(KundenListe, 1, "Max Mustermann", "max@beispiel.de")*

*KundeHinzufuegen(KundenListe, 2, "Erika Mustermann", "erika@beispiel.de")*

*KundeSuchen(KundenListe, 1) // Sollte Max Mustermann finden*

*KundeEntfernen(KundenListe, 1) // Sollte Max Mustermann entfernen*

*KundeSuchen(KundenListe, 1) // Sollte "Kunde nicht gefunden" ausgeben*