INTERNATIONALE HOCHSCHULE

FALLSTUDIE

Aufgabenstellung zum Kurs: IPWA02-01 – Programmierung von industriellen Informationssystemen mit Java EE

INHALTSVERZEICHNIS

1. AUFGABENSTELLUNG

Wähle für Deine Bearbeitung einer der folgenden Fallstudien aus.

Bitte berücksichtige bei Deiner Bearbeitung die in der jeweiligen Fallstudie selbst beschriebene Aufgabenstellung.

Hinweis zum Urheberrecht und zur Plagiatsprüfung:

Es wird darauf hingewiesen, dass der IU Internationale Hochschule GmbH das Urheberrecht der Prüfungsaufgaben/Aufgabenstellungen obliegt. Einer Veröffentlichung der Aufgabenstellungen auf Drittplattformen wird ausdrücklich widersprochen. Im Falle einer Zuwiderhandlung stehen der Hochschule u.a. Unterlassungsansprüche zu. Zudem weisen wir darauf hin, dass jede eingereichte schriftliche Ausarbeitung mittels einer Plagiatssoftware überprüft wird. Wir empfehlen daher auch, keinesfalls ausgearbeitete Lösungen zu teilen, da dies den Verdacht eines Plagiates begründen kann.

1.1. Aufgabenstellung 1: Like Hero To Zero

Du arbeitest im Web-Development-Team einer PR-Agentur, die verschiedene Naturverbände und Vereine unterstützt. Ein Großspender hat Geld für eine App bereitgestellt, mit dem Ziel, eine Web-Frontend für ein Nachhaltigkeitsprojekt mit dem Namen „Like Hero To Zero“ zu entwickeln. Es soll öffentlich zugängliche Daten zu weltweiten CO2-Emissionen darstellen (ohne Login). Außerdem soll die Webanwendung eine Backend-Oberfläche (mit Login) besitzen, mit der registrierte Wissenschaftler:innen neue Daten hinzufügen oder Datenfehler korrigieren können.

Der Requirements Engineer hat sich bereits im Vorfeld mit ausgewählten Mitgliedern aus NABU, BUND und weiteren Vereinen ausgetauscht und dazu die folgenden User Stories im Product Backlog hinterlegt und nach der MoSCoW-Methode priorisiert:

1. MUST: Als umweltpolitisch interessierte:r Bürger:in will ich den aktuellsten im Datensatz verfügbaren CO2-Ausstoß des Landes nachlesen können, über dessen Staatsbürgerschaft ich verfüge.
2. MUST: Als registriere:r, Daten beitragende:r Wissenschaftler:in will ich die jüngsten Daten aus meiner Klimaforschung in dem Datensatz hinterlegen.
3. COULD: Als Herausgeber:in des Datensatzes will ich sicherstellen, dass Ergänzungen oder Änderungen an den Daten erst freigegeben werden müssen.

Das Projekt ist agil konfiguriert und Du bist Teil eines interdisziplinären Teams aus vielen Entwickler:innen.

Aufgabenstellung:

1. Richte ein öffentliches Code-Repository für Dein Projekt ein, z. B. in GitHub.

2. Entwickle einen Prototyp für die Webanwendung, der die MUST-Funktionen anbietet (inkl. persistenter Datenhaltung). Nutze dabei den Technologiestack aus dem Skript:

- JSF und CDI/Beans, optional dazu: andere Komponentenbibliotheken (z. B. PrimeFaces);
- JPA mit einem Persistenz-Provider (z. B. Hibernate);
- eine relationale Datenbank (z. B. MySQL);
- Alternative: Verwende Spring bzw. Spring Boot mit Spring Beans anstelle von JSF mit CDI Beans. Nutze auch in diesem Falle JPA oder Spring Data JPA zusammen mit einem JPA-Provider und einer relationalen Datenbank.

3. Dokumentiere die Entwicklung der Webanwendung im Fallstudienbericht. Im Hauptteil sollten das Design und die Umsetzung beschrieben werden. Dabei soll jeweils im Entwurf und anhand von Screenshots in der tatsächlichen Umsetzung gezeigt werden, wie …

- … die einzelnen Webseiten gestaltet sind, und wie man zwischen den Webseiten hin- und herspringen kann,
- wie die Datenbankstruktur beschaffen ist und
- wie die softwaretechnische Architektur gestaltet ist (mind. ein UML-Strukturdiagramm inkl. aller verwendeter Beans).

Tipp: Verwende einen öffentlich verfügbaren Datensatz, z. B. diesen:

Rearc (2022). *CO2 Emissions (kt) | World Bank Open Data.* https://aws.amazon.com/marketplace/pp/prodview-qf3r4b6jpivte#usage

1.2. Aufgabenstellung 2: Require4Testing

Du arbeitest im Web-Development-Team eines kleinen Startups. Es soll ein neues Projekt mit dem Namen „Require4Testing“ gestartet werden, mit dem Ziel, eine Web-App zur Organisation manueller Anwendertests zu entwickeln. Der Requirements Engineer hat sich bereits im Vorfeld mit Personen aus ausgewählten Projekten ausgetauscht, die für die Planung, Erstellung und Durchführung von Tests verantwortlich sind. Im Ergebnis hat er die folgenden User Stories im Product Backlog hinterlegt und nach der MoSCoW-Methode priorisiert:

1. MUST: Als Requirements Engineer möchte ich zu testende Anforderungen erstellen können.
2. MUST: Als Testmanager:in möchte ich Testläufe anlegen können.
3. MUST: Als Testfallersteller:in möchte ich zu einer Anforderung Testfälle erstellen können.
4. SHOULD: Als Testmanager:in möchte ich einem Testlauf verschiedene Testfälle und eine:n Tester:in zuordnen können.
5. SHOULD: Als Tester:in möchte ich mir zugeordnete Testfälle mit einem Ergebnis versehen können.
6. COULD: Als Tester:in möchte ich einen Überblick über mir zugewiesene Testfälle haben.
7. COULD: Als Testmanager:in möchte ich einen Überblick über den Status aller Testdurchführungen haben.
8. COULD: Als Testfallersteller:in möchte ich zu Testfällen einzelne Testschritte erfassen.

Aufgabenstellung:

1. Richte ein öffentliches Code-Repository für Dein Projekt ein, z. B. in GitHub.

2. Entscheide Dich für fünf Anforderungen aus dem Product Backlog, die im ersten Sprint prototypisch umgesetzt werden sollen.

3. Entwickle einen Prototyp für die Webanwendung, der die ausgewählten Funktionen rudimentär anbietet. Achte dabei auf die persistente Datenhaltung. Nutze dabei den Technologiestack aus dem Skript:

- JSF und CDI/Beans, optional dazu: andere Komponentenbibliotheken (z. B. PrimeFaces);
- JPA mit einem Persistenz-Provider (z. B. Hibernate);
- eine relationale Datenbank (z. B. MySQL);
- Alternative: Verwende Spring bzw. Spring Boot mit Spring Beans anstelle von JSF mit CDI Beans. Nutze auch in diesem Falle JPA oder Spring Data JPA zusammen mit einem JPA-Provider und einer relationalen Datenbank.

4. Dokumentiere die Entwicklung der Webanwendung. Im Hauptteil sollten das Design und die Umsetzung beschrieben werden. Dabei soll jeweils im Entwurf und anhand von Screenshots in der tatsächlichen Umsetzung gezeigt werden, wie …

- … die einzelnen Webseiten gestaltet sind, und wie man zwischen den Webseiten hin- und herspringen kann,
- wie die Datenbankstruktur beschaffen ist und
- wie die softwaretechnische Architektur gestaltet ist (mind. ein UML-Strukturdiagramm inkl. aller verwendeter Beans).

Hinweis: Von einem Prototyp erwartet der Product Owner eine App, die eine pragmatische, eher funktional ausgerichtete Gestaltung der Nutzeroberfläche aufweist. Das Ergebnis muss also keinen Design Award gewinnen.

1.3. Aufgabenstellung 3: Ghost Net Fishing

Du arbeitest im Web-Development-Team der großen Non-Profit-Organisation „Shea Sepherd“. Es soll ein neues Projekt gestartet werden, mit dem Ziel, eine Web-App für den das Melden und Bergen von sogenannten Geisternetzen zu entwickeln. Geisternetze sind herrenlose Fischernetze, die im Meer treiben. Der Requirements Engineer hat sich bereits im Vorfeld mit ausgewählten Stakeholdern in der Organisation und der Fischerei ausgetauscht und dazu die folgende Anforderung in das Fachkonzept geschrieben:

Ein Geisternetz hat die folgenden Eigenschaften:

- Standort (GPS-Koordinaten),
- geschätzte Größe und
- Status.

Der Status eines Geisternetzes kann folgende Ausprägungen haben:

- Gemeldet (Eine meldende Person hat das Geisternetz im System erfasst.)
- Bergung bevorstehend (Eine bergende Person hat die Bergung angekündigt.)
- Geborgen (Eine bergende Person hat das Geisternetz erfolgreich geborgen.)
- Verschollen (Eine beliebige Person hat festgestellt, dass das Geisternetz am gemeldeten Standort nicht auffindbar ist.)

Es gibt meldende und bergende Personen. Beide Arten von Personen sind natürliche Personen mit einem Namen und einer Telefonnummer für Rückfragen. Meldende Personen können anonym bleiben und brauchen dann keine Telefonnummer. Man kann Netze aber nicht anonym als verschollen melden. Um unnötige Bergungsfahrten und Missverständnisse zu vermeiden, können Geisternetze immer nur maximal einer bergenden Person zugeordnet werden. Bergende Personen können allerdings mehr als ein Geisternetz gleichzeitig bergen.

Dazu hat er die folgenden User Stories im Product Backlog hinterlegt und nach der MoSCoW-Methode priorisiert:

1. MUST: Als meldende Person möchte ich Geisternetze (anonym) erfassen können.
2. MUST: Als bergende Person will ich mich für die Bergung eines Geisternetzes eintragen können.
3. MUST: Als bergende Person möchte ich sehen, welche Geisternetze noch zu bergen sind.
4. MUST: Als bergende Person möchte ich Geisternetze als geborgen melden können.
5. COULD: Als bergende Person möchte ich die noch nicht geborgenen Netze auf einer Weltkarte sehen.
6. COULD: Als bergende Person möchte ich sehen können, wer welche Geisternetze bergen möchte, um sich ggf. abzustimmen und die Bergungen umzuverteilen.
7. COULD: Als beliebige Person möchte ich Geisternetze als verschollen melden können.

Das Projekt ist agil konfiguriert und Du bist Teil eines interdisziplinären Teams aus vielen Entwickler:innen.

Aufgabenstellung:

1. Richte ein öffentliches Code-Repository für Dein Projekt ein, z. B. in GitHub.

2. Entscheide Dich für fünf Anforderungen aus dem Product Backlog, die im ersten Sprint prototypisch umgesetzt werden sollen.

3. Entwickle einen Prototyp für die Webanwendung, der die ausgewählten Funktionen rudimentär anbietet. Achte dabei auf die persistente Datenhaltung. Nutze dabei den Technologiestack aus dem Skript:

- JSF und CDI/Beans, optional dazu: andere Komponentenbibliotheken (z. B. PrimeFaces);
- JPA mit einem Persistenz-Provider (z. B. Hibernate);
- eine relationale Datenbank (z. B. MySQL);
- Alternative: Verwende Spring bzw. Spring Boot mit Spring Beans anstelle von JSF mit CDI Beans. Nutze auch in diesem Falle JPA oder Spring Data JPA zusammen mit einem JPA-Provider und einer relationalen Datenbank.

4. Dokumentiere die Entwicklung der Webanwendung. Im Hauptteil sollte das Design und die Umsetzung beschrieben werden. Dabei soll jeweils im Entwurf und anhand von Screenshots in der tatsächlichen Umsetzung gezeigt werden, wie …

- … die einzelnen Webseiten gestaltet sind, und wie man zwischen den Webseiten hin- und herspringen kann,
- wie die Datenbankstruktur beschaffen ist und
- wie die softwaretechnische Architektur gestaltet ist (mind. ein UML-Strukturdiagramm inkl. aller verwendeter Beans).

Hinweis: Von einem Prototyp erwartet der Product Owner eine App, die eine pragmatische, eher funktional ausgerichtete Gestaltung der Nutzeroberfläche aufweist. Das Ergebnis muss also keinen Design Award gewinnen.

2. ZUSATZINFORMATIONEN ZUR BEWERTUNG DER FALLSTUDIE

Bei der Analyse der Fallstudie und ihrer Bearbeitung müssen die im Prüfungsleitfaden aufgeführten Bewertungskriterien und Erläuterungen berücksichtigt werden.

3. BETREUUNGSPROZESS

Für die Betreuung der Fallstudie stehen grundsätzlich mehrere Kanäle offen. Die jeweilige Inanspruchnahme liegt dabei im eigenen Verantwortungsbereich. Die Tutor:innen stehen für fachliche Rücksprachen zur Themenwahl einerseits sowie für formale und allgemeine Fragen zum wissenschaftlichen Arbeiten andererseits zur Verfügung. Eine Abnahme von Gliederungen, Textteilen oder –entwürfen durch die Tutor:innen ist hierbei jedoch nicht vorgesehen, da die eigenständige Erstellung Teil der zu erbringenden Prüfungsleistung ist und in die Gesamtbewertung einfließt. Es werden jedoch Hinweise zu Gliederungsentwürfen gegeben, um den Einstieg in die Strukturierung einer wissenschaftlichen Arbeit zu erleichtern.