Project Requirements - Praktikum Software Engineering - SS 2020

https://github.com/jku-win-se/teaching.ss20.prse.prwiki

Problemstellung

Für die Meldung, Durchführung, Verwaltung und Abrechnung von Dienstreisen soll ein neues Reisemanagement Portal entwickelt werden.
Ein Team aus drei Entwickler soll das Projekt in mehreren Sprints interaktiv über einen Zeitraum von 4 Monaten realisieren und alle notwendigen Artefakte, Software, Architekturbeschreibung, Tests, Dokumentation etc. erstellen.

Requirements

1	*Basis:* Anlegen einer neuen Reise
2	*Basis:* Ändern einer bereits angelegten Reise
3	*Basis:* Löschen einer angelegten Reise
4	*Basis:* Reisen können wiederkehrend angelegt werden (jede Woche, Monat, etc… bis Enddatum)
5	*Basis:* Jede Reise kann mehrere Teilnehmer und Reiseziele haben.
6	*Basis:* Reisen und Teile einer Reise können als erledigt/abgesagt markiert werde
7	*Basis:* Ansicht der Reisen: Reisen werden in Listenform dargestellt.
8	*Basis:* Ansicht der Reisen: Reisen werden in Kalenderform dargestellt
9	*Attachments:* Jede Reise kann mehrere Attachments haben
10	*Import/Export:* Laden und speichern von/in Datei aller und ausgewählter Reisen (nicht csv)
11	*Import/Export:* Backup erstellen - Die Reisen können im CSV-Format exportiert werden.
12	*Kategorien:* Es können beliebig viele Kategorien für Reisen verwaltet werden.
13	*Kategorien:* Jede Reise kann mehreren Kategorien zugeordnet sein.
14	*Filter:* Reisen können nach Tags gefiltert werden
15	*Filter:* Reisen können nach Monat/Jahr gefiltert werden
16	Sortieren: Reisen Können nach Datum sortiert werden
17	*Sortieren:* Reisen koennen nach Status sortiert werden
18	Sortieren: Reisen können nach Kategorie sortiert werden
19	Reisekostenabrechnung: Für jede Reise kann eine Reisekostenabrechnung
20	*Reisekostenabrechnung:* Grafische Auswertung und Darstellung von Reisen, Reisekosten, etc..
21	*Logging:* Für Nachvollziehbarkeit der Aktionen sollen wichtige Ereignisse (Erstellen einer Reise, Abrechnung, Fehler etc.) in ein eigenes Log-File geschrieben werden

Software Engineering Kontext

Das Projekt bietet die Möglichkeit ein Software Produkt im vollen Umfang von der Planung bis zur fertigen Implementierung zu erstellen. Programmieren, sprich das schreiben von Code ist eine Wichtige Komponente, darüber hinaus gibt es jedoch eine Reihe von anderen AKtivitäten in der Praxis die für das erfolgreiche Gelingen eines Software Projekts von zentraler Bedeutung sind.

- Spezifizieren, Planen, und Entwerfen eines Softwareproduktes
- Objektorientierte Programmierung und Testen (Unit Tests)
- Arbeiten im Team
- Arbeiten mit SE Tools
- Versionsverwaltung (GitHub)
- Projektmanagement (Redmine)
- Buildmanagement (Maven + CircleCi)
- Planen von Sprints und Release Versionen
- Erstellen einer Systemdokumentation
- Architektur, Dokumentation von Testfällen

Zeitplan

März			April		Mai		Juni			Juli
05.03	**12.03**	**30.03**	**02.04**	**23.04**	**11.05**	**14.05**	**04.06**	**22.06**	**25.06**	**16.07**
	T078	**12.00**			**12.00**			**12.00**		
Vorbesprechung	Req. Workshop Sprint Planning 1	**R1**	Sprint Planning 2	Project Meeting	**R2**	Sprint Planning 3	Project Meeting (Code Review)	**R3**	Final Presentation	**Final Release+ Documen-tation**

	Gruppentermin, Praesentation (**Anwesenheitspflicht!**)
	Individuelle Gruppen Meetings (Termin nach Absprache)
	Abgabe mit eigenem Github-Branch

Releases & Deliverables

Für jedes Release gilt: Requirements, Tasks, Bugs, etc. werden in Redmine verwaltet.
Verantwortungen und Aufwände immer in Redmine eintragen!
Es muss mindestens 1x pro Woche Code in GitHub committed werden. Bei jedem commit immer die jeweilige id eintragen (#TaskNr). - Jedes Teammitglied muss Code schreiben und committen!
Am Ende jedes Sprints müssen die jeweiligen Tasks, Requirements, Bugs, etc. geschlossen werden und den Releases zugeordnet sein.

Release 1 - Fokus: UI Prototyp und OO Design

1. Erstes Konzept zum Aufbau der Applikation (Welche Features, Komponenten, Klassen)
UML Klassendiagramm mit den wichtigsten Klassen (Klassennamen, Hierarchien, Methoden...)
Zur Erstellung des Klassendiagramms muss ein entsprechendes UML Tool verwendet werden (keine von Hand gezeichneten Klassendiagramme!)
2. UI Prototype
3. Lauffähige Builds in CircleCI
4. Status Präsentation 1 (für Sprint Planning Meeting)

Die Abgabe erfolgt in eigenem branch: '`version-0.1`', Alle Dokumente (als pdfs) sollen in einem Ordner '`designdocuments`' gesammelt werden

Release 2 - Fokus: Prototyp Implementierung und Unit Tests

1. Erweiterte/aktualisierte UML Diagramme
2. Prototyp Implementierung:
    - Erste lauffähige Version des User Interfaces
    - Teile der Features funktionsfähig
3. Unit Tests fuer einzelne (wichtige) Klassen
4. Use Case Beschreibungen (siehe Use Case Template)
5. Status Präsentation 2 (Für Sprint Planning Meeting)

Die Abgabe erfolgt in eigenem branch: '`version-0.2`', Alle Dokumente (als pdfs) sollen in einem Ordner '`designdocuments`' gesammelt werden

Release 3 - Fokus: Dokumentation & Code Qualität

1. Erweiterte/aktualisierte UML Diagramme
2. Erweiterte Unit Tests
3. Implementierung:
    - User Interfaces
    - Großteils lauffähige Version (alle Features vorhanden)
4. Analyse der Codequalität mit PMD, Findbugs, etc.
5. Erste Version der Projektdokumentation
6. Status Präsentation 3 (Für Sprint Planning Meeting), Live Demo/Screencast der Applikation

Die Abgabe erfolgt in eigenem branch: '`version-0.3`', Alle Dokumente (als pdfs) sollen in einem Ordner '`designdocuments`' gesammelt werden

Finales Produkt

1. Finale Projektdokumentation
2. Lauffähige, finale Version der Applikation
3. Github Dokumentation (Readme mit Installationsanleitung, etc.)
4. Javadoc für wichtige Klassen, Interfaces und Methoden

Die Abgabe erfolgt in eigenem branch: '`version-1.0`', Alle finalen Dokumente sollen in einem Ordner '`documentation`' gesammelt werden