2023

Projektdokumentation

JAVAFX TODO APP

ANTON HORN, TOBIAS METZGER, LUIS KRONENBITTER, TOMISLAV ZECEVIC

SOFTWAREENTWICKLUNG 2 | Sommersemester 23

1 ABKÜRZUNGSVERZEICHNIS

GUI - Graphical User Interface (Grafische Benutzeroberfläche)

FXML - JavaFX Markup Language (XML-basierte Beschreibungssprache für JavaFX)

JDBC - Java Database Connectivity (Java-Datenbankanbindung)

UML - Unified Modeling Language (Vereinheitlichte Modellierungssprache)

JVM - Java Virtual Machine (Java-Virtual-Maschine)

JUnit - Java Unit Testing Framework (Java-Testframework für Unittests)

API - Application Programming Interface (Programmierschnittstelle)

MVC - Model-View-Controller (Modell-Präsentation-Steuerung)

CRUD - Create, Read, Update, Delete (Erstellen, Lesen, Aktualisieren, Löschen)

OOP - Object-Oriented Programming (Objektorientierte Programmierung)

IDE - Integrated Development Environment (Integrierte Entwicklungsumgebung)

XML - Extensible Markup Language (Erweiterbare Auszeichnungssprache)

CSS - Cascading Style Sheets (Gestaltungssprache für Webseiten)

JSON - JavaScript Object Notation (Datenformat zur Serialisierung von Daten)

SQL - Structured Query Language (Strukturierte Abfragesprache)

2 EINLEITUNG

Die Projektdokumentation bietet einen umfassenden Überblick über die javafx-todo-app. Sie beschreibt das Ziel der Anwendung, die Teammitglieder und die Aufteilung der Arbeiten. Zudem werden die Funktionalitäten, die Roadmap und die verwendeten externen Bibliotheken vorgestellt.

3 KURZBESCHREIBUNG DES PROJEKTS

Die javafx-todo-app ist eine benutzerfreundliche Anwendung im ansprechenden Pixel-Art-Stil, die es Benutzern ermöglicht, ihre Aufgaben effektiv zu erstellen und zu verwalten. Mit Gamification-Elementen wie dem Sammeln von Punkten/Coins für abgeschlossene Aufgaben wird das Erledigen von Aufgaben zu einem unterhaltsamen Erlebnis. Die App bietet eine Vielzahl von Funktionen, darunter das Hinzufügen, Anzeigen, Löschen, Terminieren und Priorisieren von Aufgaben. Darüber hinaus können Aufgaben nach Kategorien sortiert und gefiltert werden, um eine optimale Übersichtlichkeit zu gewährleisten. Mit der javafx-todo-app haben Benutzer die Möglichkeit, ihre Aufgaben einfach und effizient zu organisieren.

4 BESONDERHEITEN

4.1 PIXEL-ART-STIL

Die Anwendung verwendet ein visuelles Design im Pixel-Art-Stil. Das sorgt für einen Spielbezogenen Nostalgischen Look.

4.2 GAMIFICATION

Durch das Sammeln von Punkten/Coins für abgeschlossene Aufgaben wird eine spielerische Motivation geschaffen. Umgesetzt wurde dies durch Trophäen, die durch das Punktsammeln freigeschalten werden.

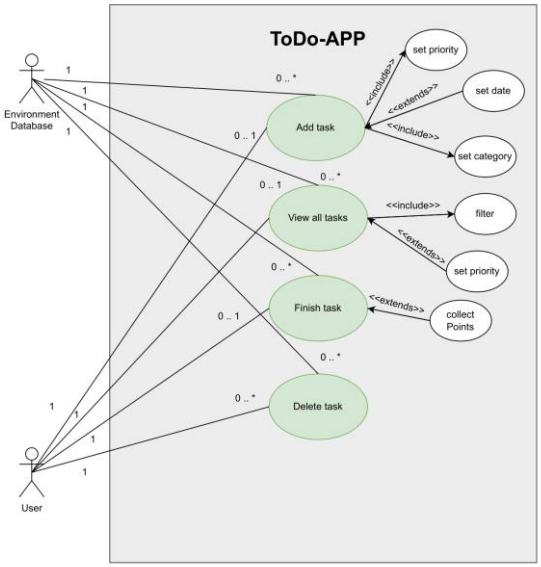
4.3 VERWENDUNG VERSCHIEDENER EXTERNER BIBLIOTHEKEN

JUnit Jupiter für Unit-Tests, Log4j für Logging, SQLite JDBC für die Datenbankanbindung und weitere Bibliotheken zur Unterstützung der Funktionalität.

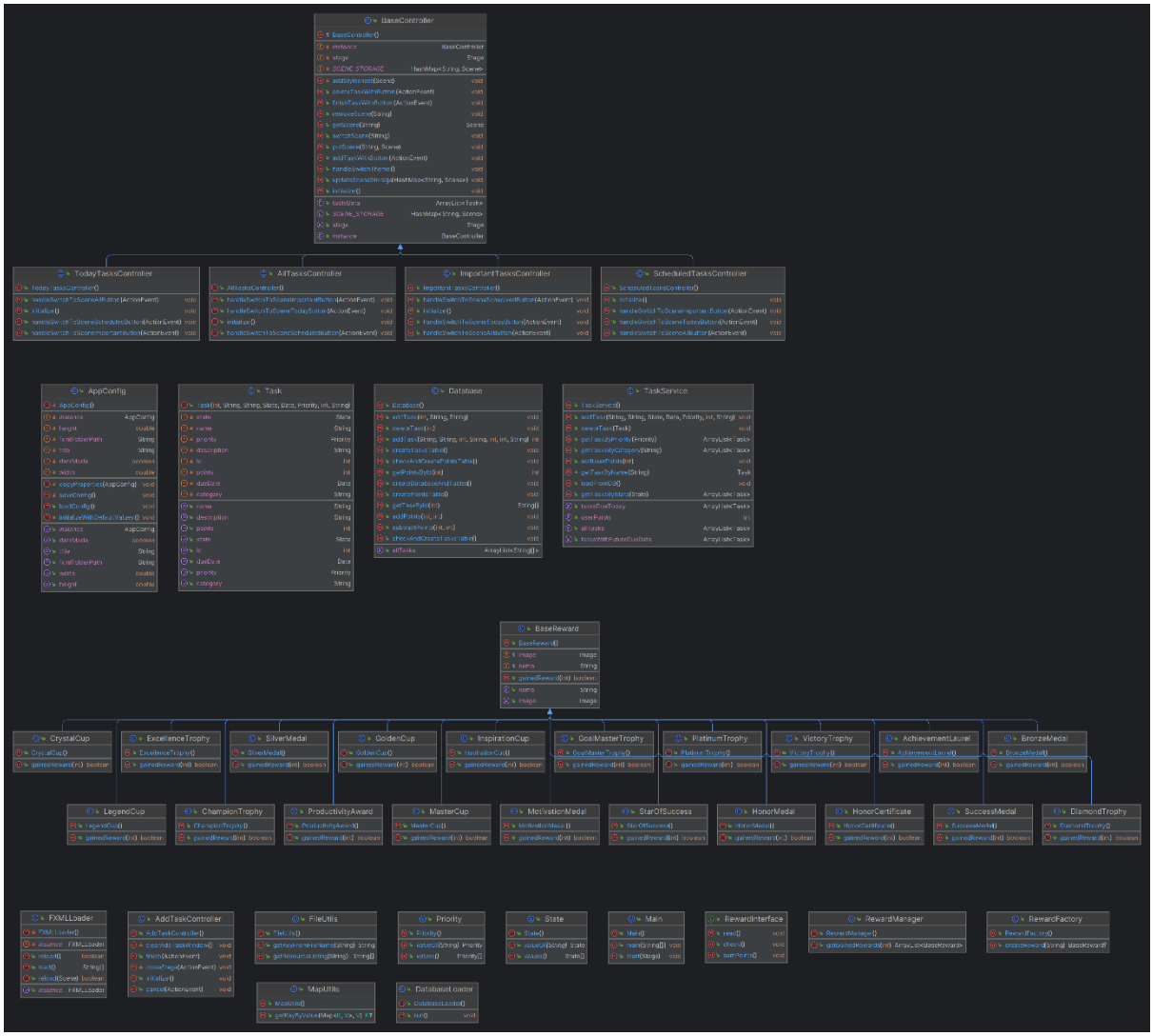
5 ARCHITEKTUR

Die Architektur des Projekts folgt einem klar strukturierten Schichtenmodell. Das Backend ist für die Datenverwaltung zuständig, während das Frontend die Benutzeroberfläche bereitstellt. Die Klasse “TaskService” bietet dem Frontend eine Schnittstelle an die Datenbank. Sie konvertiert Daten, gruppiert diese und besitzt Helfermethoden, die das Arbeiten mit diesen Daten vereinfacht. Diese Aufteilung ermöglicht eine gute Trennung der Verantwortlichkeiten und eine bessere Wartbarkeit des Codes.

6 UML-DIAGRAMM



7 CLASS-DIAGRAMM



8 STELLUNGNAHMEN

8.1 CLEAN CODE

Der Code des Projekts folgt den Clean-Code-Prinzipien, ist gut strukturiert und leicht lesbar. Namenskonventionen und Kommentare sind konsistent angewendet, was die Wartbarkeit des Codes verbessert. Es wurde darauf geachtet, dass Daten möglichst konsistent und effizient gekapselt sind. D.h. es wurde vermieden, dass Klassen-Daten als “public" zugriffsmodifiziert sind. In der Klasse “BaseReward” wurde der Zugriffsmodifizierer “protected” genutzt, da wir hier keinen Konstruktor verwenden wollten um das Factory-Pattern optimal umzusetzen.

8.2 TESTS

Es wurden umfangreiche Unit-Tests mit JUnit Jupiter erstellt, um die Funktionalität der Komponenten zu überprüfen und Fehler frühzeitig zu erkennen. Dadurch wird die Robustheit und Stabilität des Projekts gewährleistet.

8.3 GUI (JAVAFX)

Die Benutzeroberfläche wurde mit JavaFX erstellt und bietet eine ansprechende und benutzerfreundliche Darstellung der Aufgaben. Es gibt einen Basiscontroller, von dem alle anderen Controller erben. Dies verhindert Code Duplikation und gibt dem GUI-Code eine Struktur. Die GUI selbst ist in unterschiedliche Szenen eingeteilt.

8.4 LOGGING UND EXCEPTIONS

Log4j wurde verwendet, um wichtige Ereignisse und Fehler zu protokollieren. Die Anwendung behandelt Exceptions und bietet dem Benutzer geeignete Fehlermeldungen. Die unterschiedlichen Log-Stufen werden logisch eingesetzt.

8.5 UML, THREADS UND STREAMS

UML-Diagramme wurden erstellt, um die Projektstruktur zu visualisieren. Mehrere Threads werden beim Start der Applikation verwendet. Wir laden/erstellen die Datenbank zeitgleich mit dem Laden der FXML-Dateien. Dies ermöglicht einen schnellen Start der Applikation. Streams werden in der Klasse “TaskService” eingesetzt. Z.B. beim Suchen der Tasks nach einer bestimmten Kategorie. Der Einsatz der Streams hat den Code verkürzt und ihn deutlich lesbarer gemacht

8.6 FACTORIES

Das Factory-Pattern wird im Kontext der Belohnungen eingesetzt. Wir haben eine Basisklasse (BaseReward). Jede Belohnung/Trophäe erbt von der Basisklasse. In der Klasse “RewardFactory” kann nun nur anhand des Namens das richtige Objekt erstellt werden. Dies macht das System sehr dynamisch und würde uns ermöglichen noch weitere Belohnungen leicht hinzuzufügen.

8.7 AUFGABENVERTEILUNG

Gleich zu Beginn des Projekts, nachdem wir die Ziele definiert hatten und die Datenbankstruktur mit UML geplant war, haben wir die Aufgaben auf die vier Teammitglieder verteilt. Wir haben uns in zwei Teams aufgeteilt: das Backend-Team und das Frontend-Team. Diese Aufteilung hat zu einem

schnellen Fortschritt in unserem Projekt geführt, da wir in kurzen Meetings regelmäßig den aktuellen Stand besprechen und uns abstimmen konnten.

Durch diese klare Aufgabenverteilung konnten wir effizient arbeiten und den Fortschritt des Projekts sicherstellen. Die regelmäßigen Meetings haben uns ermöglicht, uns abzustimmen, unsere Fortschritte zu präsentieren und bei Bedarf gegenseitige Unterstützung anzubieten. Dank dieser Zusammenarbeit konnten wir produktiv arbeiten und das Projekt erfolgreich vorantreiben.

9 FAZIT

Das Projekt hat uns einen Einblick in die Softwareentwicklung gegeben. Wir haben gelernt mit verschiedenen Techniken wie z.B. Logging, UML und Tests umzugehen und diese so einzusetzen, dass sie und das Entwickeln vereinfachen. Zusätzlich konnten wir uns mit zentralen Konzepten der Softwareentwicklung vertraut machen. Gerade über das Thema Threading konnten wir alle viel lernen.

Die meiste Probleme hatten wir bei der Erstellung der GUI. Oftmals wurden die FXML-Dateien nicht richtig geladen. Dies verzögerte die Entwicklung maßgeblich, da bei unserer ToDo-App die GUI eine der Hauptaufgaben war. In Zukunft würden wir mit einer anderen Bibliothek zur GUI-Erstellung arbeiten.