27.6.2020



Time Management

Team 5

Christopher Gusenbauer

Projekt  
dokumentation

Inhaltsverzeichnis

[1. Einleitung 3](#_Toc40889930)

[2. Umgesetzte Anforderungen 3](#_Toc40889931)

[3. Überblick über das System aus Benutzersicht 3](#_Toc40889932)

[4. Überblick über das System aus Entwicklersicht 3](#_Toc40889933)

[4.1. Entwurf 3](#_Toc40889934)

[4.1.1. Überblick über das System 3](#_Toc40889935)

[4.1.2. Wichtige Design Entscheidungen 3](#_Toc40889936)

[4.2. Implementierung 3](#_Toc40889937)

[4.3. Code Qualität 3](#_Toc40889938)

[4.4. Testen 3](#_Toc40889939)

[5. Installationsanleitung 4](#_Toc40889940)

Versionshistorie

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Datum | Ersteller | Änderungen |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Tabelle 1: Versionshistorie

# Einleitung

*In diesem Projekt wurde ein Aufgabenplaner als Desktopapp implementiert. Mit diesem Programm („Tasky“) kann man Aufgaben verwalten, kategorisieren, filtern, Zeittracking betreiben, diese Daten sichern und importieren und Statistiken ansehen.  
  
Aufgrund der besonderen Situation des Wegfallens von zwei Teammitgliedern wurde das Projekt allein von Christopher Gusenbauer entwickelt und konzipiert. Der finale Code enthält 0% des Codes der ausgeschiedenen Mitglieder.   
  
Deswegen wurde ebenfalls vereinbart, die Anforderungen etwas abzuändern und das Programm von Projektmanagement mehr Richtung Taskmanagement zu entwickeln. Das wurde umgesetzt.*

# Umgesetzte Anforderungen

*Welche der Anforderungen haben Sie umgesetzt, wer aus dem Team war verantwortlich und welches Stundenausmaß ist damit verknüpft*

*Related issues*

*User Requirement #2394: TB-01*

*User Requirement #2395: TB-02*

*User Requirement #2396: TB-03*

*User Requirement #2401: TM-01*

*User Requirement #2403: TM-04*

*User Requirement #2401: TM-01*

*User Requirement #2403: TM-04*

*User Requirement #2978: TB-06: Aktivitäten können mit beliebig vielen Tags versehen werden*

*User Requirement #2979: TM-02: Zeittracking in der Applikation*

*User Requirement #2981: TF-05: Grafische Auswertung von Aktivitäten*

*User Requirement #2982: TF-01/02/03 Filtern und Sortieren der Aktivitäten*

*User Requirement #2983: TB-Neu-1: Intuitive Markierung von Aktivitäten als erledigt*

*User Requirement #2984: TB-Neu-2: Anpassung der Eigenschaften einer Aktivität*

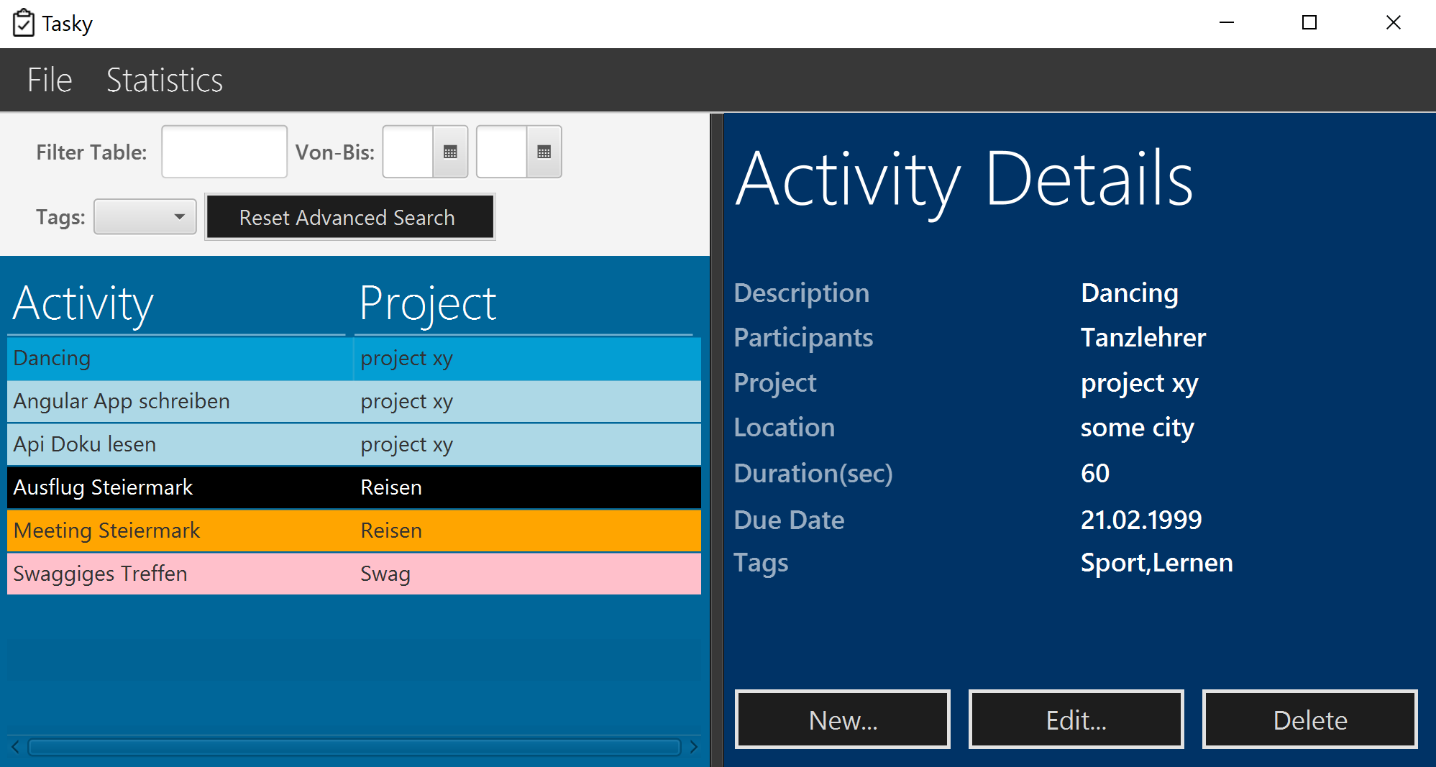
*User Requirement #2986: TB-Neu-3: Applikation als ausführbares Programm zur Verfügung stellen.*

*Gesamtaufwand laut Redmine > 200 Stunden.*

*Verantwortlich für Alle Requirements: Christopher Gusenbauer*

# Überblick über das System aus Benutzersicht

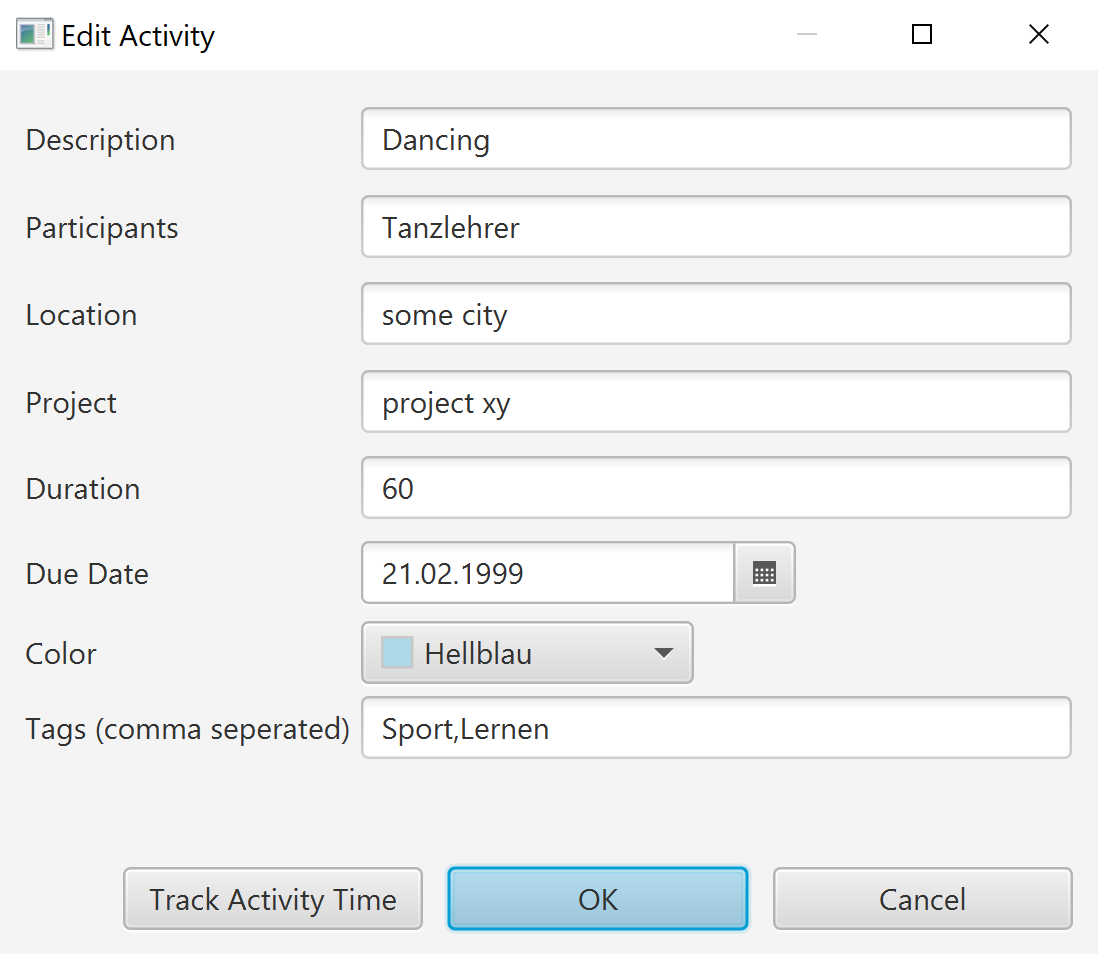
*Wie wurden die Anforderungen in der Benutzeroberfläche umgesetzt (Screenshots der Benutzeroberfläche und Beschreibung der Funktionalität anhand von Szenarien)*

*Wie wird das System verwendet?*

*Das ist Taskys Startoberfläche. Sie ist in vier Areale aufgeteilt. Oben befindet sich das Menü, über das die Daten von Tasky als XML gespeichert oder geladen werden können. Außerdem kann man dort die Statistikansichtöffnen. Auf der linken Seite befinden sich das Filtermenü und die Übersichtsansicht.*

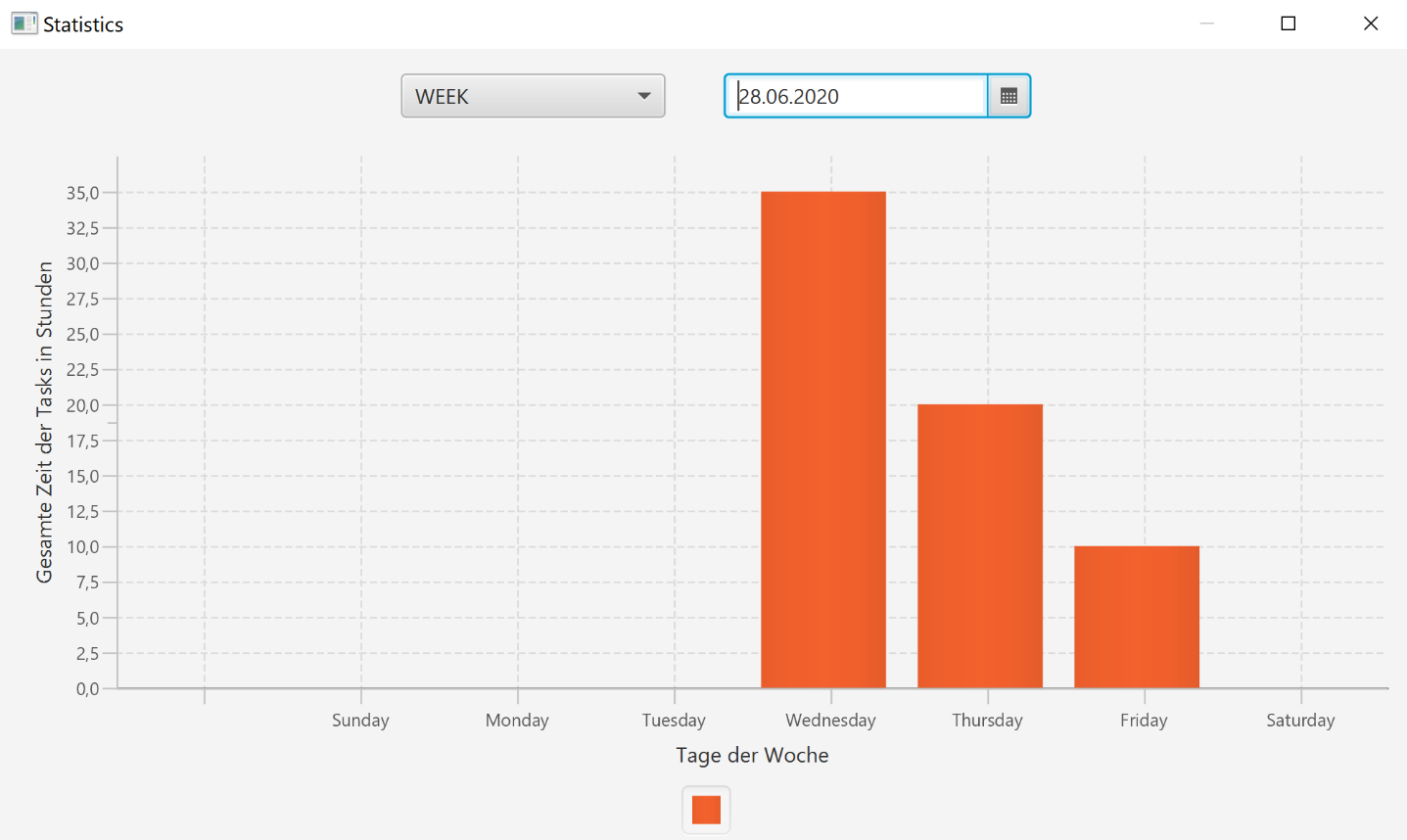
*In der Übersichtsansicht werden die Tasks tabularisch dargestellt und können mittels Klick sortiert werden und via Drag & Drop in ein eigens erscheinendes DeleteArea die Tasks löschen. In der Filteransicht kann nach textuellen Inhalten der Aktivitäten, nach einem Zeitraum des Fälligkeitsdatums und nach den inkludierten Tags gefiltert werden. Außerdem können die Suchkriterien auch zurückgesetzt werden.*

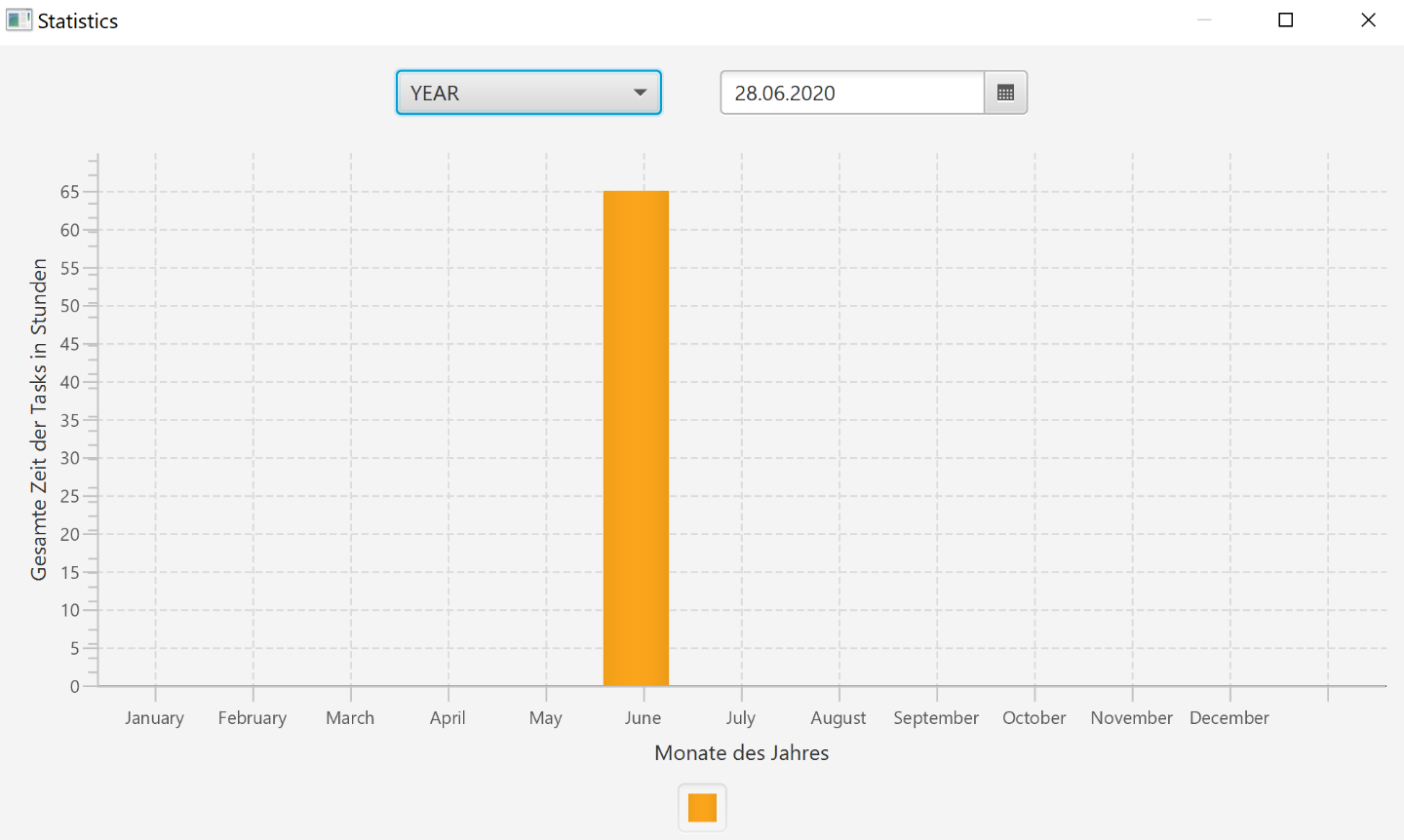
*Rechts sehen wir eine Detailansicht der jeweilig ausgewählten Aktivität und ihrer Attribute. Mit den Buttons „New“, „Edit“, „Delete“ wird die jeweilige Option für die ausgewählte Aktivität ausgeführt. Mit dem Edit oder New Button gelangt man in folgende Ansicht:*



*Hier kann eine Aktivität verändert werden. Zusätzlich zu den „normalen“ Eigenschaften gibt es auch besonderheiten. Ein Due Date wird mittels einem Date Picker gewählt. Mit einem Color-Picker kann man die Farbe auswählen, in der die Aktivität in der tabularischen Ansicht dargestellt wird. Es können mehrere Tags angegeben werden, welche automatisch gefiltert und getrennt werden und in der Filteransicht als Drop-Down verfügbar sind. Mit einmaligen Klick auf „Track Activity Time“ wird die Duration Eigenschaft im Sekundentakt hochgezählt, bis der Button erneut geklickt wird.*

*Auf der folgenden Seite ist noch die Statistikansicht zu sehen, welche bei gewähltem Zeitpunkt entweder die verbrauchte Zeit aller Aktivitäten pro Monat eines Jahres oder Tag einer Woche (des Zeitpunkts) anzeigt.*



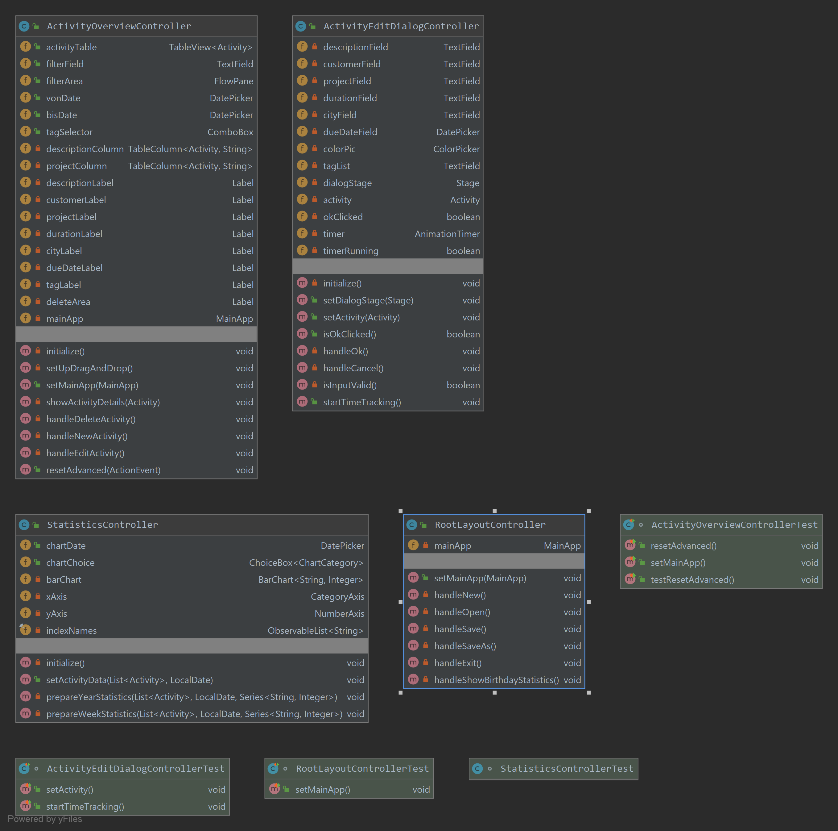
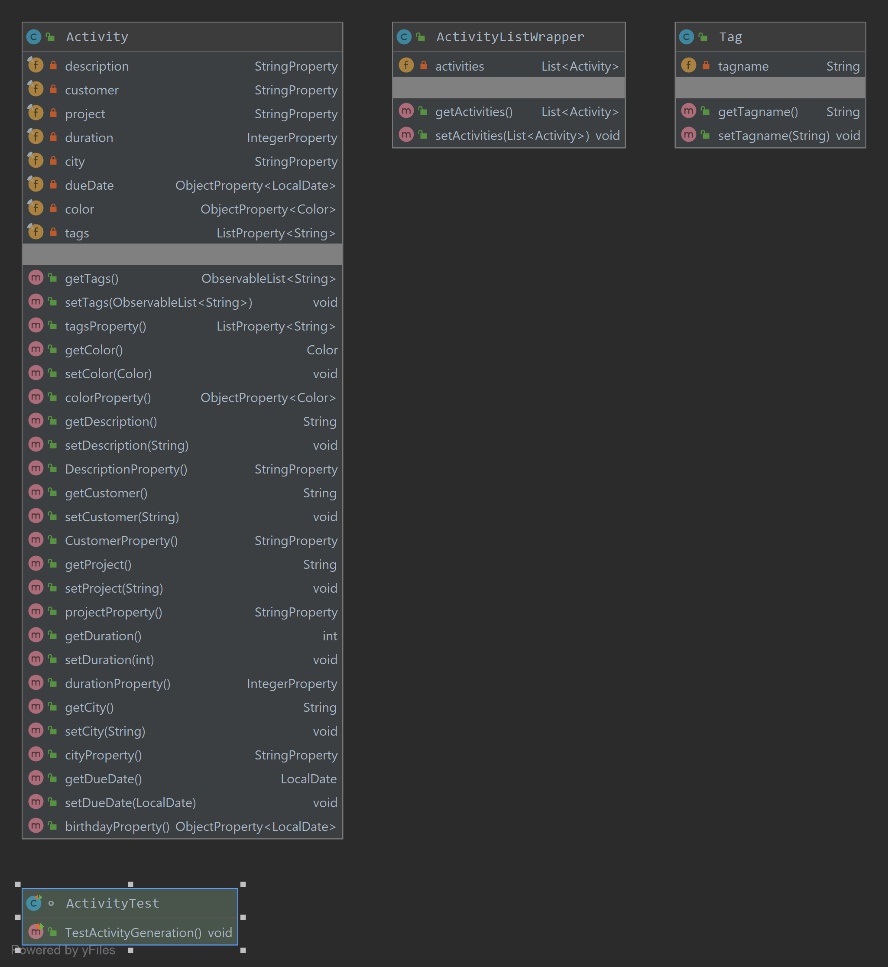


# Überblick über das System aus Entwicklersicht

## Entwurf

### Überblick über das System

*UML Diagramm mit Erläuterungen*

*Verwendete Design Muster (z.B. Model-View-Controller)*

*Das MVC Design Muster wurde leicht abgewandelt gewählt. Die View entspricht den FXML Dateien, das Model & der Controller (Package Controller) sind aber vermischter als üblich, da die App sehr kompakt auf nur einer Hauptseite dargestellt werden muss. Des weiteren gibt es eine Main Klasse die die Orchestrierung der App übernimmt.*

### Wichtige Design Entscheidungen

*Beschreibung der 3-5 wichtigsten Design Entscheidungen nach folgenden Schema*

*Entscheidung:*

*Begründung:*

*Alternativen, die in Betracht gezogen wurden:*

*Annahmen:*

*Konsequenzen:*

## Implementierung

*Beschreibung wichtiger Aspekte der Implementierung (eventuell mit ausgewählten Codestücken), Projektstruktur, Abhängigkeiten, verwendete Bibliotheken*

## Code Qualität

*Beschreibung der Verwendung von PMD, Beschreibung der Findings und welche davon behoben wurden*

*Viele Findings, aber viele davon falsche Treffer wegen JavaFX, teils sinnlose Findings und einige deren Benefit den Aufwand für mich nicht Wert war, bzw. die unlösbar sind. Z. B. meine FilterLogik, da viele Filter aufeinmal filtern ist die Komplexität einfach recht hoch.*

*In Fällen wo die Implementationszeit verdreifacht werden würde, wurde ebenfalls nichts geändert, da der Benefit hier für mich nicht wirtschaftlich erscheint.*

*85 Findings wurden behoben.*

## Testen

Es wurde immenser Aufwand betrieben um sinnvolle Tests für JavaFX zu schreiben, wobei die Schlussfolgerung aber darauf hinauslief, dass es nicht mit dem derzeitig im Web verfügbaren Wissenstand möglich ist, tatsächlich nützliche, über absolute Basictests hinausgehende Tests zu schreiben. Aufgrund der Tatsache, dass es sich um eine GUI-App handelt und praktisch keine alleinige „Model-Logik“ verfügbar ist, sondern hauptsächlich Controller und GUI-Logik ist mit JUNIT alleine nur eine sehr geringe Anzahl an sinnvollen Tests schreibbar. Es wurde intensive Recherche betrieben, eine Lösung zu finden. TestFX <https://github.com/TestFX/TestFX> wäre eine mögliche Lösung. Leider existiert vermutlich aufgrund der geringen Popularität von JavaFX im Internet kaum Dokumentation oder Anleitungen, die über das reine Setup von TestFX hinausgehen. Die meisten Anleitungen sind bereits Outdated. Schlussendlich hat sich auch einfach herausgestellt, das der Aufbau meiner App für Code-Tests nicht gut geeignet ist, und tatsächliche GUI-Tests hier fast ein must-have wären, weil aufgrund einiger Anforderungen von Spezialfällen eine teils starke Verbindung der Klassen notwendig war.  
  
Um etwaige Fehler identifizieren zu können, wurde deswegen versucht sehr genau mit Enduser Tests, also manuellen GUI-Tests und dem Ausprobieren von Szenarien das Programm zu testen. Mit dieser Methode wurden einige Fehler entdeckt und ausgebessert, wie beispielswiese ein Bug mit der Generation von Tags

# Installationsanleitung

*Benötigt: Maven, Java, OpenJFX oder ein JDK mit OpenJFX inkludiert, z.B. Liberica JDK*

*Via IDE: Rechtsklick auf Launcher.java, Ausführen*

*Via Maven:*

mvn clean javafx:run

Via JLINK:

Mvn javafx:jlink

Im Projektdirectoryroot in der Shell:  
target/image/bin/java -m at.gusenbauer.taskycompanion/at.gusenbauer.taskycompanion.Launcher

In der aktuellsten Projektversion wird auch ein .bat Script generiert, welches automatisch die App ausführt.

Dieses generierte Image ist Plattformabhängig und muss je OS neu generiert werden, im Github Repository inkludiert sind die Windows Dateien.