Projektdokumentation

Samira Bingesser, Gani Nurceski, Juan Gutierrez

Inhaltsverzeichnis

[Projektbeschreibung 2](#_Toc87636214)

[Use-Case Diagramm 2](#_Toc87636215)

[Klassendiagramm 3](#_Toc87636216)

[Objektorientierte Konzepte 4](#_Toc87636217)

[Encapsulation 4](#_Toc87636218)

[Delegation 5](#_Toc87636219)

[Static/Final 5](#_Toc87636220)

[Exception 5](#_Toc87636221)

[API 6](#_Toc87636222)

[Git repository 6](#_Toc87636223)

[Testfälle 6](#_Toc87636224)

[Reflexion 7](#_Toc87636225)

# Projektbeschreibung

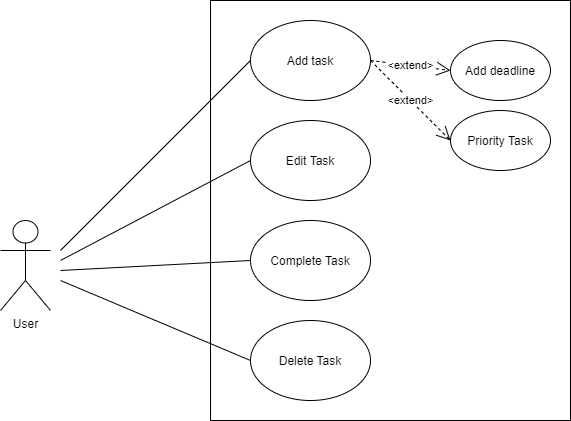
Als unser Projekt haben wir uns überlegt eine To Do App zu entwickeln. Es sollte möglich sein neue Tasks zu erfassen, bearbeiten, löschen, abschliessen (als erledigt markieren) und alle Tasks anzuzeigen. Zusätzlich kann man einem Task ein Enddatum bzw. eine Deadline und eine Priorität hinzufügen. Der User wird anhand eines Menüs durch die Konsolenapp geleitet. Die Anwendung ist eine C# Konsolenapplikation und wurde in Visual Studio in Community Edition entwickelt.

# Use-Case Diagramm

Altes Use-Case Diagramm:

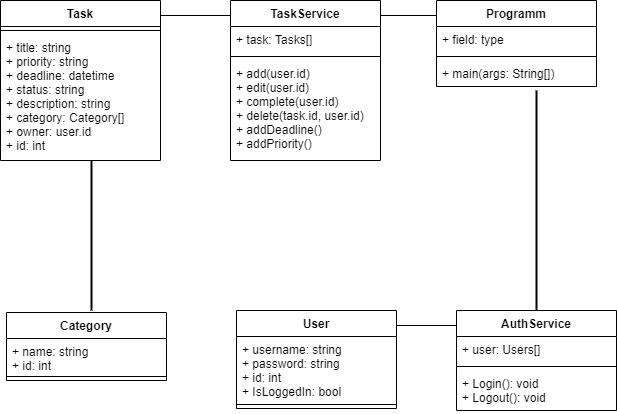


Neues Use-Case Diagramm:



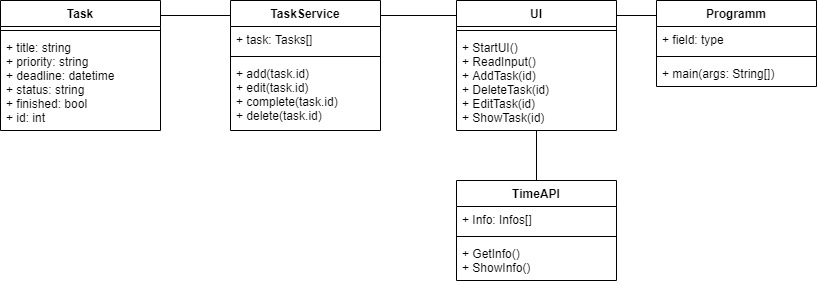
# Klassendiagramm

Hier unten sehen wir das alte sowie das neue Klassendiagramm vor und nach der Implementation.

Altes Klassendiagramm vor der Implementation:

Ein Bild, das Tisch enthält.

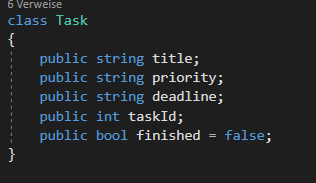
Automatisch generierte BeschreibungNeues Klassendiagramm vor der Implementation:

Klassendiagramm nach der Implementation:

# Objektorientierte Konzepte

## Encapsulation

In der Task Klasse, wo jeglich die Attribute gehalten werden, können die Attribute von aussen verändert werden, da diese den Bereich auf «Public» gesetzt wurden. Später in der UI werden die Werte in den Methoden Add und Edit angepasst, bzw. erstellt.



## Delegation

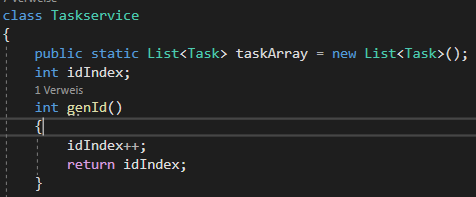
Im UI wird ein Task als abgeschlossen gezählt. Dazu geht es in der TaskService Klasse (hier als service instanziiert) und führt die Methode «Complete» aus (von der Klasse UI aus).

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

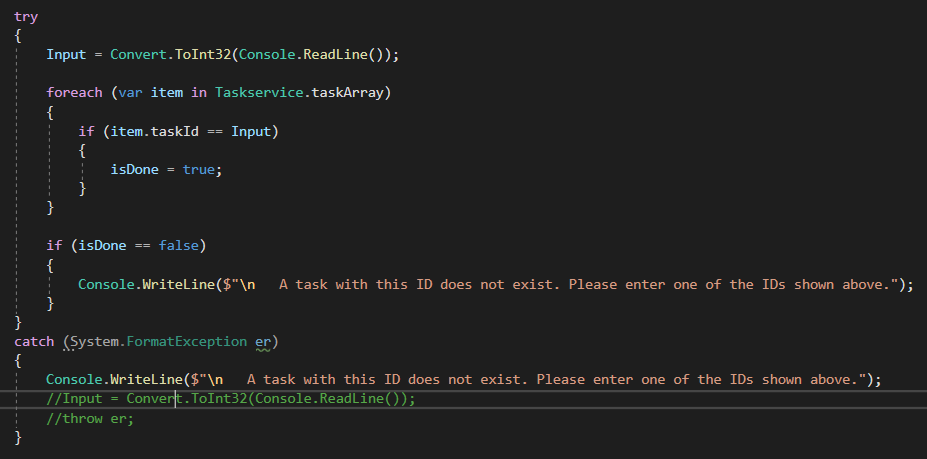
## Static/Final

In der Taskservice Klasse wurde ein Array erstellt. Darin werden alle Tasks gespeichert.



## Exception

Im UI in der CompleteTask Methode wurde ein Try und Catch Befehl eingebaut. Im Catch wird eine Rückmeldung für den Benutzer in der Konsole geschrieben.



# API

Mithilfe einer API wird beim Öffnen des Programmes (Build) der Benutzer begrüsst und ihm wird die Zeit und das Datum angezeigt. Zudem könnten wir weitere Daten, welche in der API erhalten sind, auslesen und in der Konsole ausgeben.

API: <https://worldtimeapi.org/api/timezone/europe/zurich>

# Git repository

Während des ganzen Projektes haben wir mit GitHub gearbeitet. Und unsere Fortschritte regelmässig ins Repository, welches wir ebenfalls erstellt haben, hochgeladen.

Hier ist der Link zu unserem repository: https://github.com/JuanGut01/ToDo

# Testfälle

|  |  |
| --- | --- |
| Bezeichnung | T001 |
| Voraussetzung | Der User gibt eine andere Zahl als 1, 2 oder 3 ein oder ein Sonderzeichen. |
| Erwartetes Resultat | "Letters and special characters are not permitted. Please choose one of the numbers shown above." Wird in der Konsole wiedergeben. |
| Tatsächliches Resultat | "Letters and special characters are not permitted. Please choose one of the numbers shown above." Wird in der Konsole wiedergeben. |
| Status | Bestanden |

|  |  |
| --- | --- |
| Bezeichnung | T002 |
| Voraussetzung | Die eingegebene Zahl stimmt mit den im Menü angegebenen überein. |
| Erwartetes Resultat | Folgendes sollte in der Konsole angezeigt werden:  Bei 1: Add task clicked  Bei 2: Remove task clicked  Bei anderer Eingabe als 1, 2 oder 3: That option is incorrect, please try again |
| Tatsächliches Resultat | Folgendes sollte in der Konsole angezeigt werden:  Bei 1: Add task clicked  Bei 2: Remove task clicked  Bei anderer Eingabe als 1, 2 oder 3: That option is incorrect, please try again |
| Status | Bestanden |

# Reflexion

Unser Ziel war es, eine funktionstüchtige ToDo Applikation mit einem GUI mithilfe von C# zu entwickeln. Zuerst haben wir geplant unsere To Do App mit einer Login - Funktion und WPF (Windows Presentation Foundation) zu realisieren. Da wir aber aus zeitlichen Gründen gemerkt haben, dass dies nicht machbar ist, entschieden wir uns für eine Konsolenapp ohne Login - Funktion.

Die Zusammenarbeit im Team verlief sehr gut. Was wir jedoch bemerkten, war dass wir gegen Ende des Projektes mehr Zeitdruck hatten als zu Beginn, aber wir dies zu unserem Vorteil nutzten und dementsprechend auch in der Schule die Logik des Programmes vollständig implementieren konnten. Zuhause beschäftigten nur mit dem UI für die Konsole. Nebst dem konnten wir Git kennenlernen, indem wir via Github zusammen gearbeitet haben.

Schlussendlich konnten wir ein funktionstüchtiges Programm abgeben. Zu Beginn des Projektes waren unsere Programmierkenntnisse ziemlich mager. Nach diesem Projekt können wir uns alle mit dem OOP Konzept besser auseinandersetzen.