PROJEKTDOKUMENTATION

Samira Bingesser, Gani Nurceski, Juan Gutierrez

Samira Bingesser, Gani Nurceski, Juan Gutierrez
BI18a

Inhaltsverzeichnis

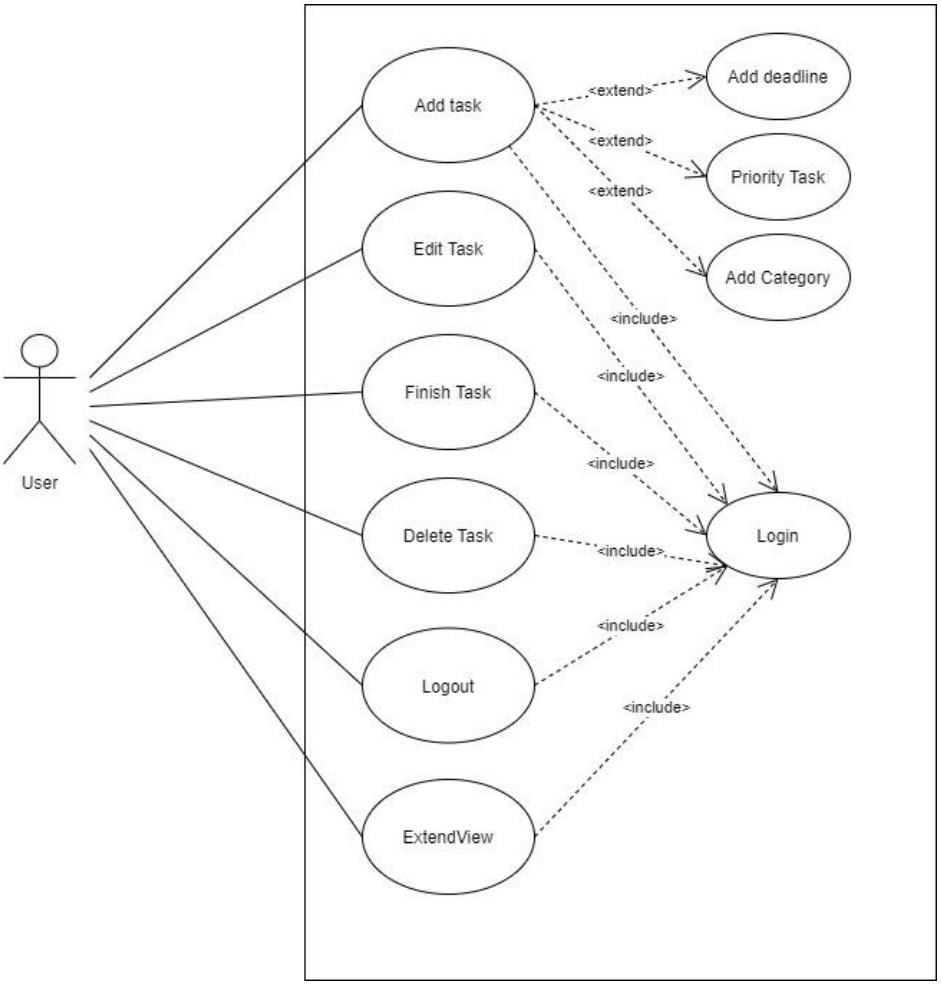
Samira Bingesser, Gani Nurceski, Juan Gutierrez
BI18a

Projektbeschreibung

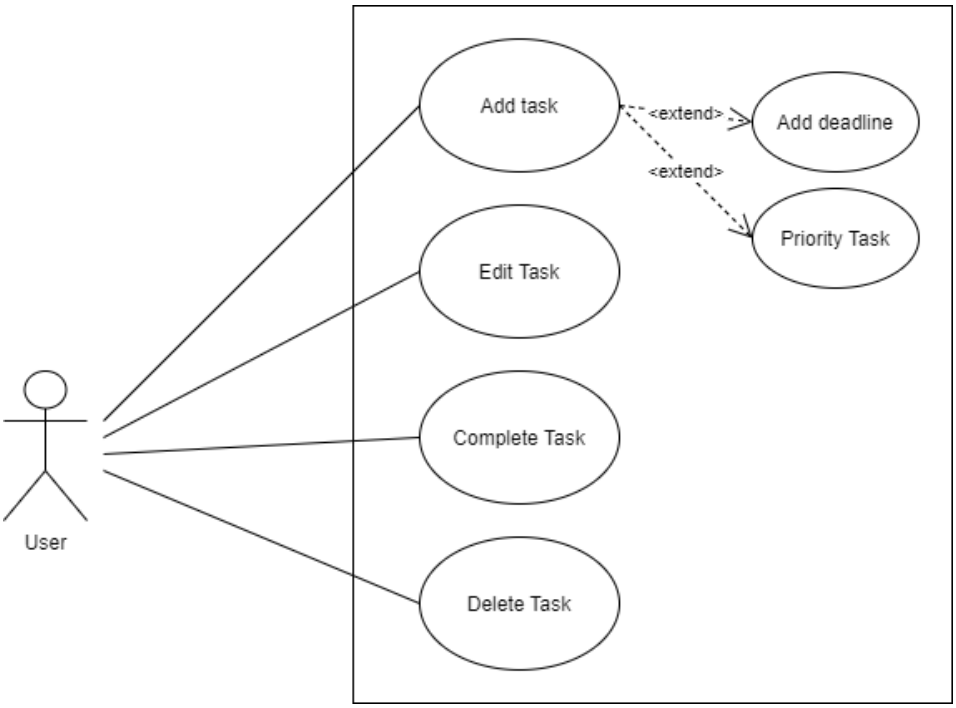
Als unser Projekt haben wir uns überlegt eine To Do App zu entwickeln. Es sollte möglich sein neue Tasks zu erfassen, bearbeiten, löschen, abschliessen (als erledigt markieren) und alle Tasks anzuzeigen. Zusätzlich kann man einem Task ein Enddatum bzw. eine Deadline und eine Priorität hinzufügen. Der 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:



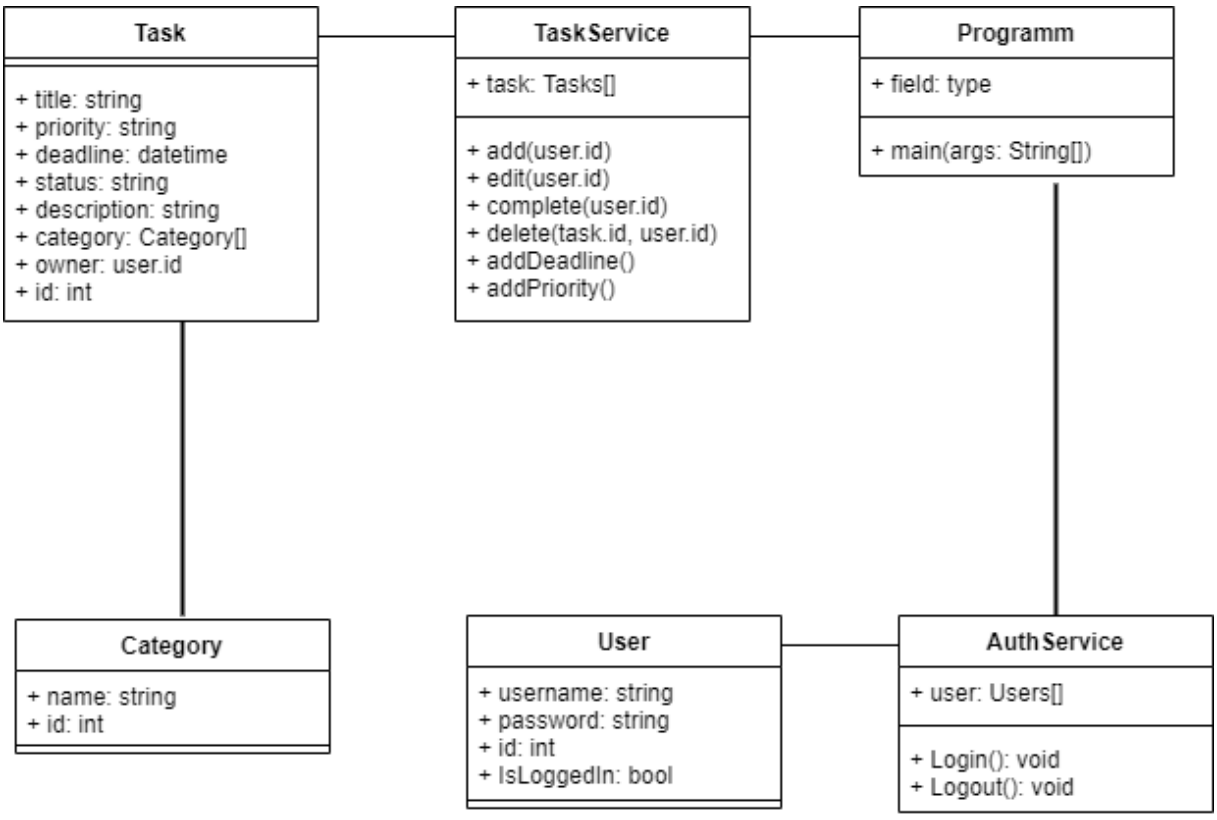
Neues Use-Case Diagramm:



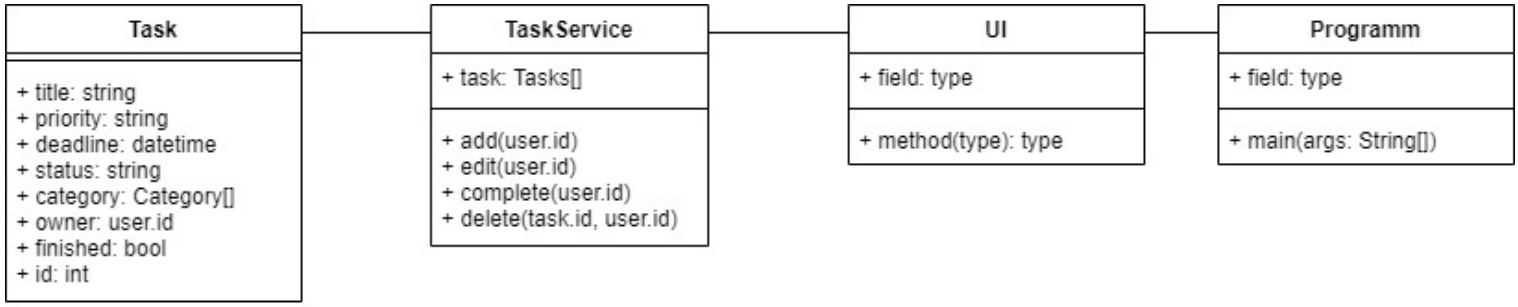
Klassendiagramm

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

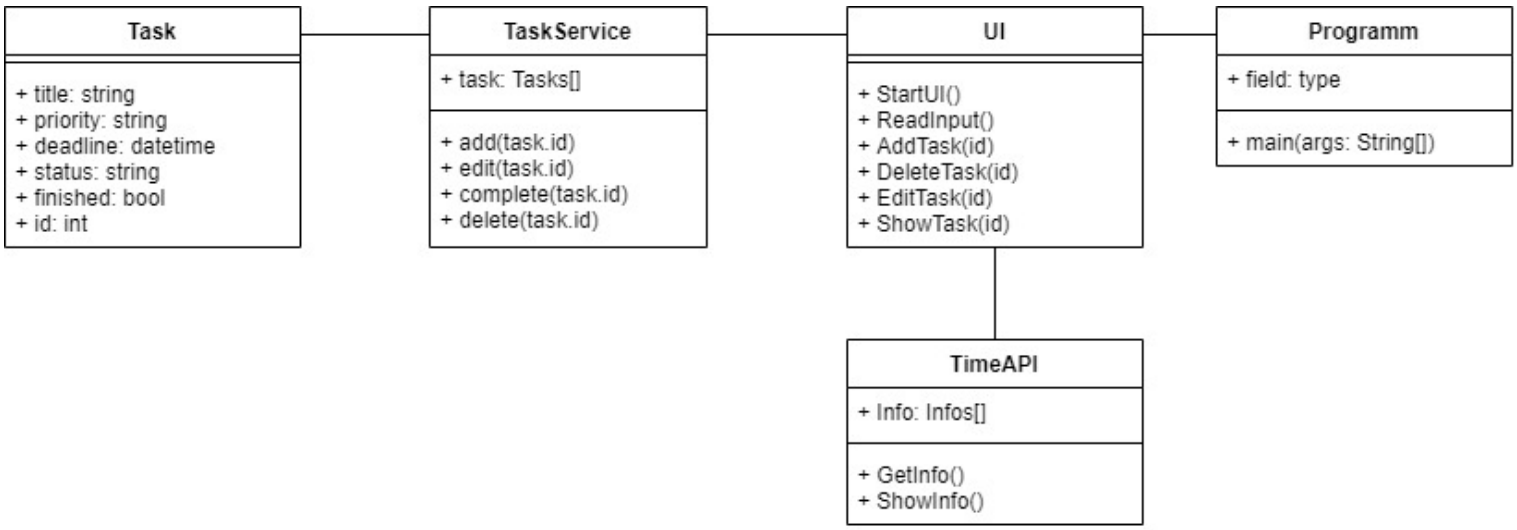
Altes Klassendiagramm vor der Implementation:



Neues 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.

```
6 Verweise
class Task
{
    public string title;
    public string priority;
    public string deadline;
    public int taskId;
    public bool finished = false;
}
```

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).

```
bool returned = service.complete(Input);
ShowMenu();
if (returned)
{
    Console.WriteLine("\n    Task was marked as completed.");
}
else
{
    Console.WriteLine("\n    Task was not marked as completed.");
}
```

Static/Final

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

```
class Taskservice
{
    public static List<Task> taskArray = new List<Task>();
    int idIndex;
    1 Verweis
    int genId()
    {
        idIndex++;
        return idIndex;
    }
```

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.

```
try
{
    Input = Convert.ToInt32(Console.ReadLine());

    foreach (var item in Taskservice.taskArray)
    {
        if (item.taskId == Input)
        {
            isDone = true;
        }
    }

    if (isDone == false)
    {
        Console.WriteLine($"\n   A task with this ID does not exist. Please enter one of the IDs shown above.");
    }
}
catch (System.FormatException er)
{
    Console.WriteLine($"\n   A task with this ID does not exist. Please enter one of the IDs shown above.");
    //Input = Convert.ToInt32(Console.ReadLine());
    //throw er;
}
```

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.