Danmarks Tekniske Universitet

Eksaminations Projekt

Project Planner

Gruppemedlemmer:
Carsten Michael Rosenkilde Nielsen s123062
Christian Mathias Rohde Kiær s123812
Jonathan Binner Becktor s123094

12. maj 2013

Indledning

Til opgave har vi fået stillet, at designe en planlægningsapplikation. Denne applikation skal skabes for et softwarehus, der mangler et system til at planlægge deres projekter. Projekter skal kunne oprettes efter de informationer, der haves på oprettelses tidspunktet. Dette skal kunne inkludere navn, start/sluttidspunkt og projektleder. Alle disse oplysninger skal også kunne tilføjes eller ændres senere hen. Projekter skal kunne tildeles aktiviteter, som tilføjes med et navn, en beskrivelse,en start/slut dato, antal allokerede timer og det aktuelle timeforbrug. Udviklere skal kunne tildeles aktiviteter, og derfra registrere deres timeforbrug på den pågældende aktivitet. Til løsning af problemerne, har vi opstillet 6 Use Cases som vi vil implementere testdrevent. Følgende Use Cases har vi valgt at kigge på:

- Create projects
- Manage projects
- Define Activities
- Manage developers
- Hours worked on project
- Seek Assistance

Vi har implementeret alle Use Cases test drevent, og det er gjort i forbindelse og uden konflikt med opgavebeskrivelsen.

Indhold

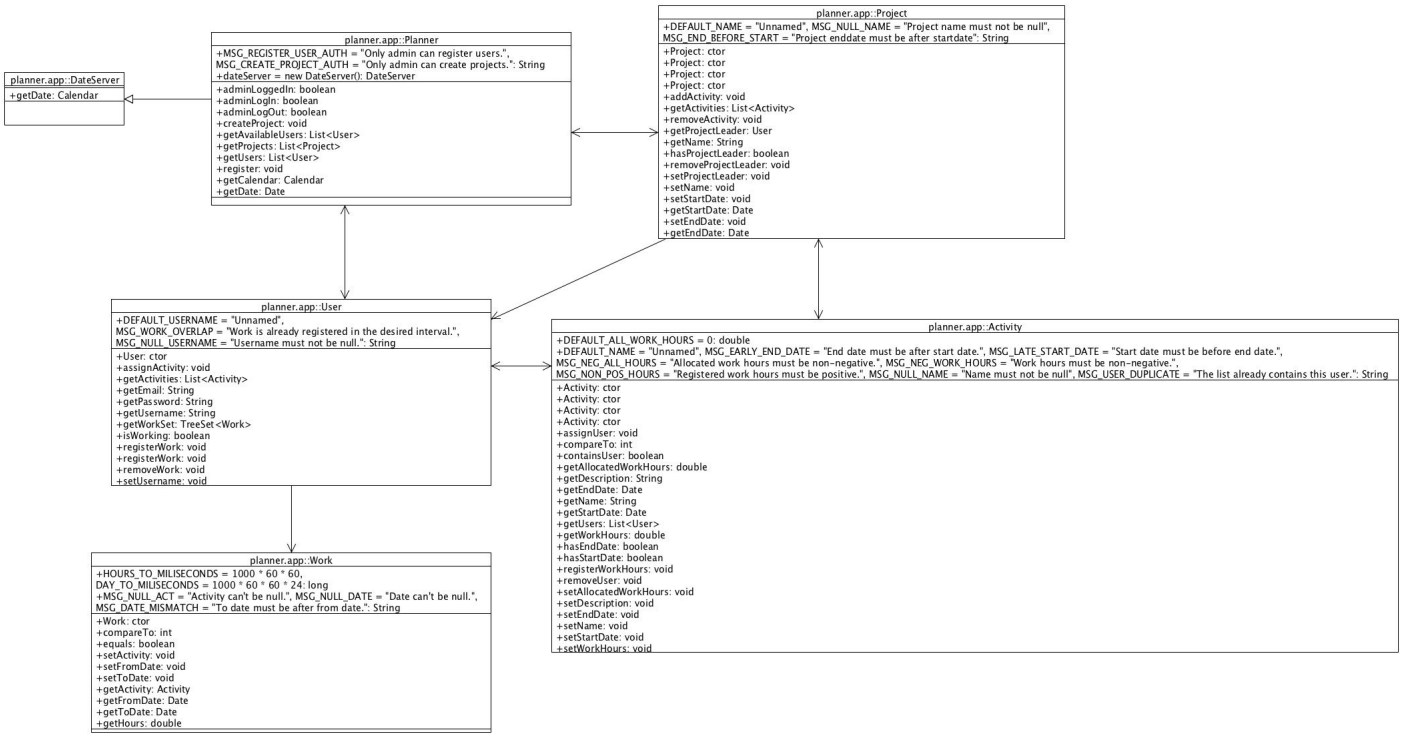
1 Kravspecifikation

1.1 Væsentlige begreber

1.2 Use cases

2 Programdesign

2.1 Klasser



Figur 1: Klassediagram

Ovenfor er vores klassediagram opstillet. Vi har valgt kun at vise de public metoder, for alle vores klasser. Derudover viser diagrammerne hvilke funktioner, der kan hente oplysninger fra hvad.

2.2 Sekvensdiagrammer

2.3 Diskussion

Vi som gruppe diskuterede vi hvad der skulle lægges fokus på. Til at starte med var vi enige om at der skulle være GUI, persistency og god funktionalitet. Dog da vi var et godt stykke inde valgte vi at fokusere på programmets funktionalitet og de tilhørende tests for at få så god "code coverage"som muligt. Da alle vores use cases kun beskæftiger sig med funktionalitet af

programmet, kunne vi derfor opnå at færdiggøre dem alle. Dette medfører også, at vores use cases har tests for de fleste scenarioer.
Oplægget til opgaven skaber en del uklarheder rundt omkring, der dog let kan ændres på. Specielt de forskellige brugers rettigheder i programmet, var vi i tvivl om. Derfor er programmet meget åbent, og kun nogle få metoder kræver brugerrettigheder.

3 Systematiske test

Alle vores tests er i bund og grund lavet for at kunne få så høj "code coverage"som muligt. Med dette som bagrund har vi lavet tests til mere eller mindre alt i programmet. Vi har taget brug af EclEmma for at finde hvor vi manglede test og hvad der ikke var covered. Nogle af de oftest funde test er tests for getters & setters hvilket var forholdsvist let at få testet man laver en værdi som sættes i en setter hvorefter vi bruger assertEquals for at se om den værdi vi har sat i setteren er den samme vi får fra getteren når vi spørger efter den. En anden test der ofte skulle testes var hvis fx. en setter ikke kunne være null og hvis den var skulle kaste en fejl. Her bruger vi try/catch i testsne for at fange de fejl som programmet giver os og sammenligner dem med de ønskede fejlmeddelelser.

Hvis vi kigger på usecase1 først tester vi om admin er logget ind. Dette gør vi ved at have en boolean value som sættes til true når det rigtige password er blevet skrevet adminLogIn() metoden. Derefter kan administratoren lave projekter. Hvis administratoren laver et projekt uden navn vil programmet tildele den et. Dette testes ved at lave et program uden navn og derefter teste ved getName metoden om navnet er blevet til "default name". Denne slags test bliver brugt flere gange i denne usecase hvis der ikke skrives start dato bliver den af default sat til dags dato. dette tjækkes på samme måde. Hvis projektet bliver lavet med end date før start date kaster programmet en fejl. Denne fanger vi med en try/catch som tjekker om fejlen og fejlmeddelelsen er det samme.

4 Konklusion

Vi har fået skabt et program, der lever op til de kravspecifikationer vi har fået stillet. Programmet er gennemtestet med mange scenarioer, og derfor har vi prøvet at sikre en høj kvalitet. Der er plads til udvidelse, hvis dette skulle være at ønske. Af egenskaber vi ikke har nået, som dog ikke

er en del af kravspecifikationerne, kan nævnes et persistency layer og et UI. Dette er selvfølgelig påkrævet for et fuldt funktionelt program. Vi har dog opnået at få skabt det grundlæggende, for en planlægningsapplikation.
Projektet har forløbet fint. Dog har der grundet andre kurser været funktioner vi har måtte udelade. Derfor har vores fokus været på de grundlæggende funktioner, og ting funktioner vi havde tiltænkt, som GUI og persistency, blev udeladt. Dette er dog noget vi i sidste ende har været glade for, da det har givet os bedre tid til at løse vores use cases. I forhold til vores oprindelige projektplan, kom vi forsent igang allerede fra uge 1. Dette medførte de fornævnte fravalg. Dette var dog noget vi blev enige om forholdsvist hurtigt. Ellers vil vi mene at vi har forsøgt at følge planen, og har prøvet at have så få overskridelser af tiden som muligt.
Alt i alt har dette givet os en mulighed, for at prøve at arbejde mod en deadline. Vi har fået indblik i hvilke valg der skal tages, og har indset at ikke alle de ideer man har gjort sig nødvændigvis kan nås.