Programrapport – Frederik Spang Thomsen

Status

Programmet er tilnærmelsesvis fuldendt. På nuværende tidspunkt mangler blot en udvidelse af Events til at håndtere mere dynamisk subscription til en User med lav saldo, der kunne bruges i et større system til fx at sende SMS, Email eller lignende advarsel eller Notifikationssystem. I udgangspunktet er der advarsler på, når man henter status og netop har købt noget, lige ligesom Rejsekort. Du får en advarsel, når du har købt noget, og din balance er endt under 50kr.

I forhold til udvidelser har jeg valgt at omstrukturere Stregsystem.cs, og dennes interface, der primært stod for hjælpefunktioner såsom GetProductById ved at implementere en BaseModel-klasse, der står for de basale ting, som alle Datamodeller har til fælles, herunder All, Where, Find og FindBy metoder. Disse har jeg implementeret ved at lave en List<T> i BaseModel, med et nedarvende koncept med et Constraint på Typen fra nedarvning. Det tænkte jeg ville spare tid i det lange løb, og give mindre redundant kode, ift. at implementere førnævnte fællesmetoder, for at kunne søge og slå op i listen. Ligeledes ville dette gøre en Database implementation lettere, da det i grove træk ville være implementation af disse til SQL kald, i stedet for Listeopslag, såsom .All skulle lave et ”SELECT * FROM {this.TableName}”, eller noget tilsvarende.
Til dette har jeg brugt C#’s indbyggede System.Reflection, for at få nogle funktioner til Metaprogrammering, såsom .GetProperty og lignende, for at hente funktioner, værdier mm dynamisk ud fra tekststrenge, så jeg har lavet en metode som User.FindBy(”Username”, value), men der virker med de properties der findes.

Alt i alt giver dette mig en struktur, der ligner MVC – Model View Controller. Altså StregsystemCLI, som View, og StregsystemController som Controller.
På sigt ville dette også gøre en implementation af fx Windows Forms lettere, da vi blot skulle udskifte CLI’et, på samme måde, som førnævnte Databaseintegration.

Noget, som jeg ville have ændret eller fortsat, givet at jeg havde længere tid til opgaven, ville være at følge konceptet ”Fat Models, Slim Controllers”, der er et ofte brugt koncept i Ruby – At holde logik og funktionalitet i Modeller og Service-klasser, og bibeholde læsbare Controllere. Dette er til dels opnået, i og med, at Controlleren primært står for at vælge en visning, (UI.DisplayX) og Model.Funktion(), alt efter disse.
Dette er i overensstemmelse med den gængse stil for MVC-opbygninger.

Programmet er udviklet i Visual Studio for Mac, og kan derfor have nogle enkelte fejl, men kører upåklageligt i et Unixmiljø, og jeg har derfor vedlagt bin/Debug mappen i .zipfilen. Jeg har vedlagt Data-mappen i output, der indeholder .csv-filerne, og automatisk opretter en Logs-mappe inden i, til at logge transaktioner. Denne data er persistent, men dog er ID’er ikke persistente på tværs af programeksekveringer.
Jeg har også forsøgt, at teste isoleret på en Windowsmaskine, ved hjælp af mine gruppekammerater, og ligeledes brugt System.IO.Path.DirectorySeperatorChar, for at bibeholde \ kontra / som mappe-seperator i Logging, samt Load i StregsystemLoader-klassen.

Kilder

- http://regex101.com til at validere Regex, før implementation.
- Tidligere arbejde hos Place2Book til Email Regex.
- Designmønsteret i undermodulet ActiveModel fra Ruby on Rails.