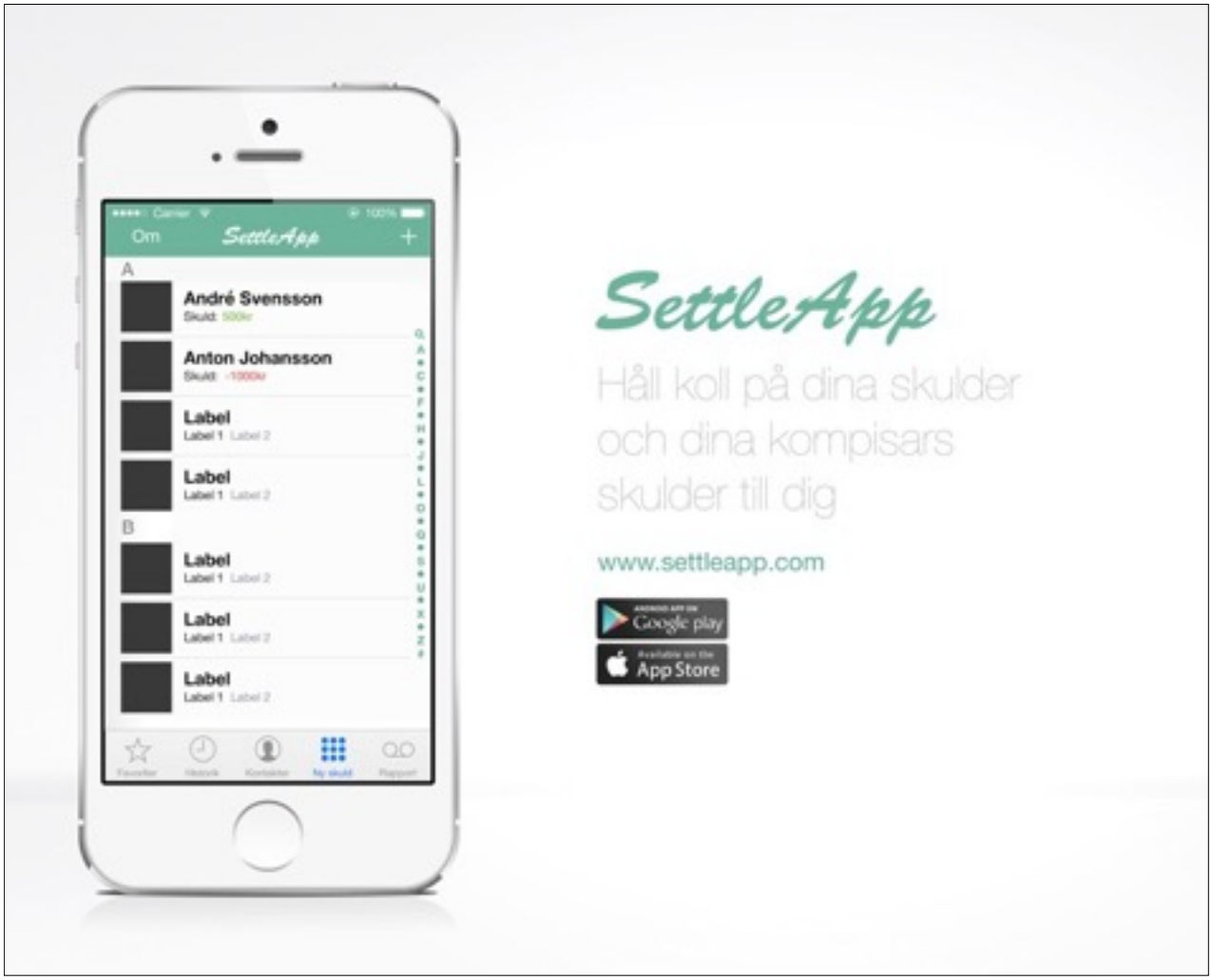


Figur 1 - Bilden är ett montage

Designpecifikation

Förberett för kursen TDDI02

Bearbetat av: Tobias Lundgren, Elias Resman,
Jonatan Sävenhed, Martin Weman

8 oktober 2014

Projektnamn: SettleApp

Handledare: Rebecka Geijer Michaeli

Version 3

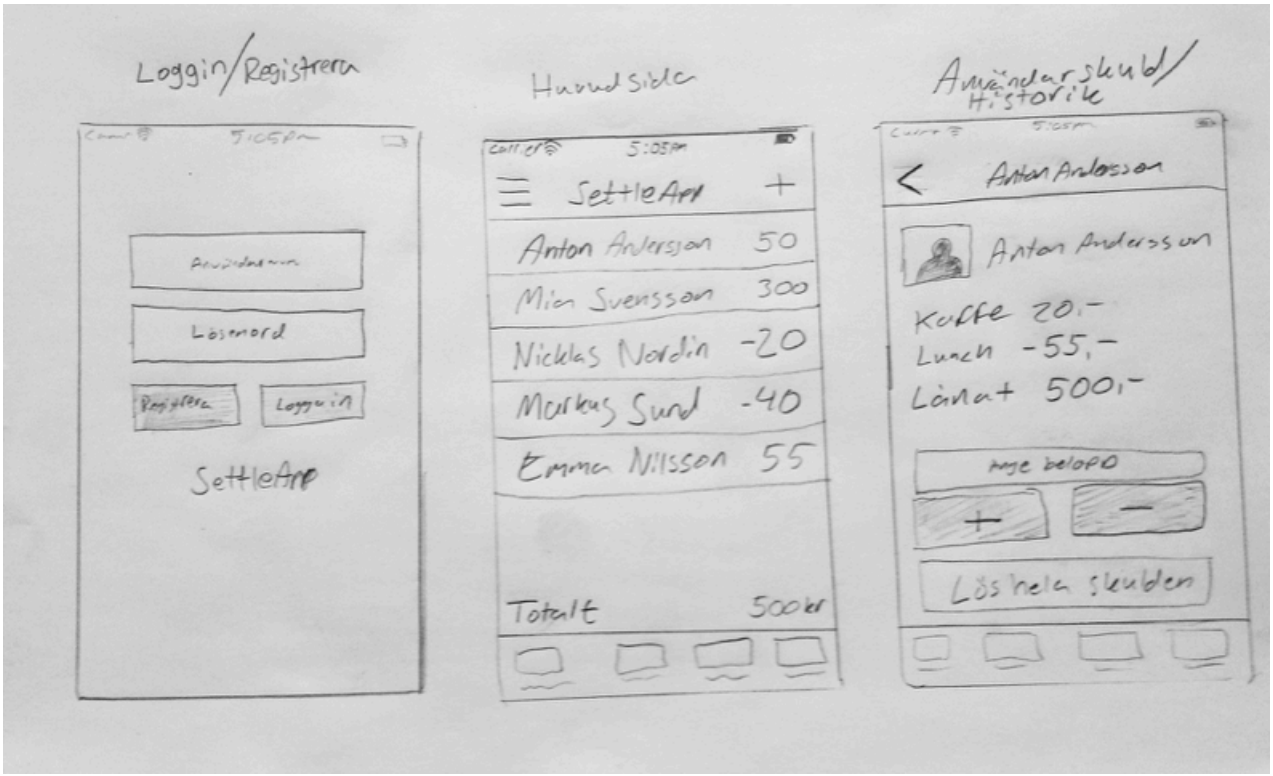
Innehåll

1. INLEDNING

Projektets syfte är att utveckla en iOS applikation som ska hålla koll på skulder mellan olika parter, se kravspecifikationen för mer information om applikationen och de krav som den ska uppfylla. I den här designspecifikationen följer grundlig och mer ingående information om applikationens klasser, klasser och dess funktioner. Designspecifikationen ger en lösning på de kraven som ska uppfyllas.

2. ANVÄNDARGRÄNSSNITT

Applikationens användargränssnitt kommer att efterliknas enligt vyerna i figur 2.



Figur 2 - Designskiss över programvyerna

Först visas inloggningsskärmen (Login/Registrera) där användaren kan logga in med befintligt användarkonto eller skapa ett nytt. Om man väljer att skapa en ny användare så får man fylla i följande fält: Användarnamn, Förnamn, Efternamn, E-postadress samt lösenord.

När man sedan loggat in så kommer man direkt till huvudsidan som är en översiktsvy över alla aktuella skulder samt den totala skulden.

Om man trycker på lägg till kontakt (+) uppe i högra hörnet enligt huvudsidan i figur 2 så kommer man till ett nytt fönster där man kan lägga till en kontakt med namn som man sedan kan debitera eller kreditera en skuld till.

Om man trycker på menyknappen (☰) enligt huvudsidan i figur 2 så visas en meny som innehåller: Kontoinformation, Kontakter, Lägg till kontakt, Om SettleApp samt Logga ut.

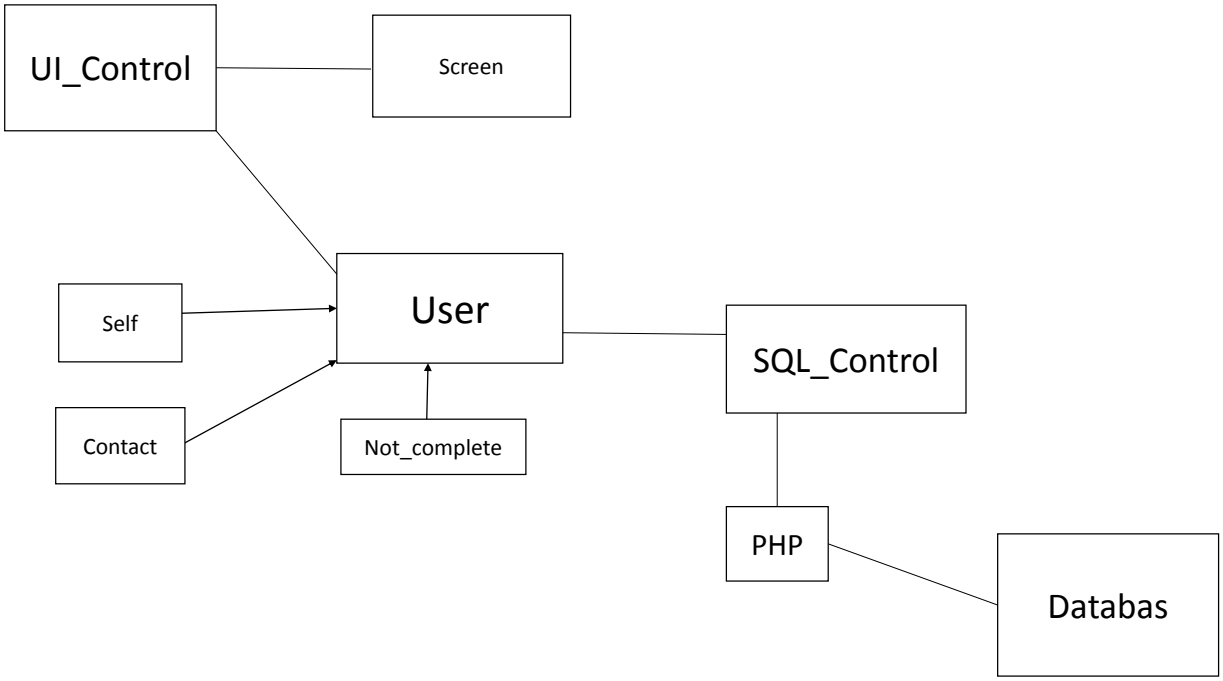
Om man trycker på ett av namnen i listan så kommer man till översiktsvyn över en vald kontakt. Där ser man all info som är relaterat till den valda kontakten som historik och total skuld samt knappar för att lägga till skuld, minska skuld eller lösa hela skulden enligt Användarskuld/Historik i figur 2.

3. SYSTEMARKITEKTURER

I det här kapitlet kommer arkitekturen på systemet att tas upp. I nivå 1 beskrivs systemet övergripligt om hur de olika modulerna och klasserna kommunicerar. Nivå 2 beskriver hur modulerna och klasserna fungerar mer detaljerat, den tar även upp vilka variabler och funktioner som varje klass förväntas ha.

3.1. NIVÅ 1

Nivå 1 beskriver systemets alla olika moduler/klasser. Modul 1 består av ”UI_Control”-klassen som kommunicerar med skärmen. I modul 2 finns ”User”-klassen och dess underklasser (”Self”, ”Contact och ”Not_Complete”), de tar emot information från ”UI_Control”, hanterar den, och ser till att ”SQL_Control” används på rätt sätt. Modul 3 består av ”SQL_Control”-klassen, PHP-scriptet och den faktiska databasen. ”SQL_Control” kommer att få information från någon av ”User”-klasserna och uppdatera databasen med informationen.



Figur 3 - Schema över systemet

3.1.1. UI_CONTROL

”UI_control” är systemets main-program som tar emot input och dirigerar anropen vidare och sedan skriver ut resultatet.

”UI_control” är den klassen som beskriver allt som ska visas på skärmen för användaren. Det inkluderar knappar, textfält, bilder, ikoner och hur de ska visas. Klassen tar emot data från användaren till exempel inloggningsuppgifter eller ändring av skuld och skickar vidare denna data till ”User”-klassen.

3.1.2. USER

”User” är en abstrakt klass som innehåller alla funktioner som sköter logiken bakom användaren och dess kontakter, det vill säga inloggning, ändring av skuld, returnera kontaktuppgifter och dylikt. Den tar emot information från ”UI_control” och utför önskad operation. Om informationen rör databasen så skickas kommandon till ”SQL_control” som returnerar önskat värde. ”User” har tre underklasser:

- ”Self” innehåller information och funktioner som bara rör användaren själv.
- ”Contact” hanterar användarinformation som berör användarens kontakter.
- ”Not_complete” används som en temporär ”Self” innan man loggat in eller skapat en ny användare och har bara två funktioner, en för att logga in samt en för att skapa en ny användare.

3.1.3. SQL_CONTROL

De enda uppgifterna ”SQL_control” har är att läsa samt skriva till databasen och returnera färdig data till ”User”, alltså ett helt objekt.

3.1.4. DATABAS

Databasen lagrar användarinformation och skulder till andra användare för alla användare. Ett PHP-dokument som används som en kommunikationstolk mellan ”SQL_control” och databasen för att förenkla mängden kod i ”SQL_control”, den innehåller även inloggningsuppgifter till databasen. När en ny användare skapas ges den ett id-nummer som beskriver dess position i databasen.

3.2. NIVÅ 2

I nivå 2 beskrivs ingående detaljerad information om de olika klasserna.

3.2.1. USER

Här under följer en mer detaljerad information om klassen ”user’s” funktioner, variabler och underklasser.

3.2.1.1. VARIABLER

User ska innehålla dessa variabler:

- String username, kommer att innehålla användarnamnet.
- String name, kommer att innehålla förnamnet.
- String surname, kommer att innehålla efternamnet.
- Integer ID, kommer att representera personens plats i databasen.
- Lista med skulder som är bundna till andra användare.

3.2.1.2. FUNKTIONER

User kommer att innehålla följande funktioner:

- Sting name() tar inga parametrar och returnerar en sträng med användarens för och efternamn.
- String username() tar inga parametrar och returnerar en sträng med användarnamnet.
- get_debts() tar inga parametrar och returnerar en pekare till listan med skulder.
- Bool change_debt(string, double) tar en sträng och en double som parametrar och returnerar en bool. Funktionen är virtuell, en abstrakt funktion och kommer att definieras i underklasserna.

3.2.1.3.UNDERKLASSER

User ska också ha 3 stycken underklasser som definieras här nedan.

Self kommer att ha följande variabler:

- String email, innehåller användarens emailadress.
- Double total_debt, ska lagra den totala skulden för användaren.
- Lista med berörda kontakter för update, updatelista.
- Ärvda variabler och funktioner från ”User”.

Self skall innehålla följande funktioner

- Email, tar inga parametrar och ska returnera en sträng med mailadressen.
- Total, tar inga parametrar och returnerar en double med den totala skulden.
- Refresh, tar inga parametrar och returnerar en bool om det gick bra eller inte. Den ber ”SQL_control” att hämta information om skulderna och uppdaterar total_debt, sedan returnerar den true. Om ingen kontakt kunde nås med databasen returneras false.
- Update, tar inga parametrar och returnerar en bool. Update kallar ”SQL_control::update” för alla kontakter i updatelistan. True returneras om det gick bra att skriva över skuldfältet, annars false.
- Change_debt, tar en double och en sträng som parametrar. Dessa ska representera förändring av skuld samt användarnamn. Först söker igenom nuvarande skulder och ser om användarnamnet redan finns bland skulderna. Om användarnamnet finns så ändras den befintliga skulden och lägger till kontakten i update-listan, om skulden blir 0 så tas den bort, sedan kallas update och true returneras. Om användarnamnet inte finns bland skulderna ber den ”SQL_control” att söka i databasen efter användarnamnet, hittas användarnamnet läggs en ny skuld till i skuldlistan, kontakten läggs även till i update-listan. Update kallas och true returneras, i övriga fall returneras false.

Not_complete skall innehålla följande variabler:

Not_complete skall inte ha några egna variabler utöver dem som ärvs av ”User”.

Not_complete skall innehålla följande funktioner:

- Login kommer att ta två strängar som parametrar och returnerar en ”User”. Dessa skall vara användarnamnet och dess lösenord. Om de båda stämmer överens med databasen returneras en ”Self” från ”SQL_control”, annars returneras en ”Not_complete”.
- Create tar en parameter för varje fält i databasen(inte id) och returnerar en ”User”. Om användarnamnet eller email-adressen inte redan används i databasen kommer ”SQL_control” bes att skapa en ny användare och en ”Self” returneras. Annars returneras en ”Not_complete”.

Contact skall ha följande variabel:

- Double debt håller koll på nuvarande skuld till kontakten.
- Ärvda variabler och funktioner från ”User”.

Contact skall innehålla följande funktioner:

- Change_debt, tar en double och en sträng som parametrar. Om strängen är tom så kommer debt att uppdateras med doubeln’s värde och true returneras, annars returneras false.

3.2.2. SQL-CONTROL

SQL-control skall innehålla följande variabler:

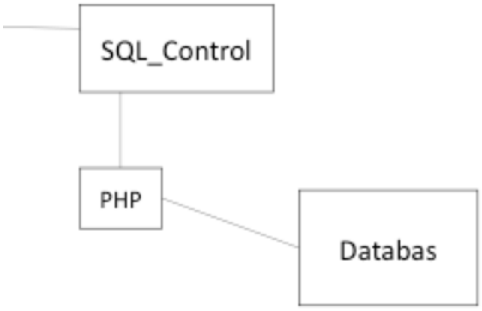
- String URL, kommer lagra sökvägen till PHP-scriptet.

SQL-control skall innehålla följande funktioner:

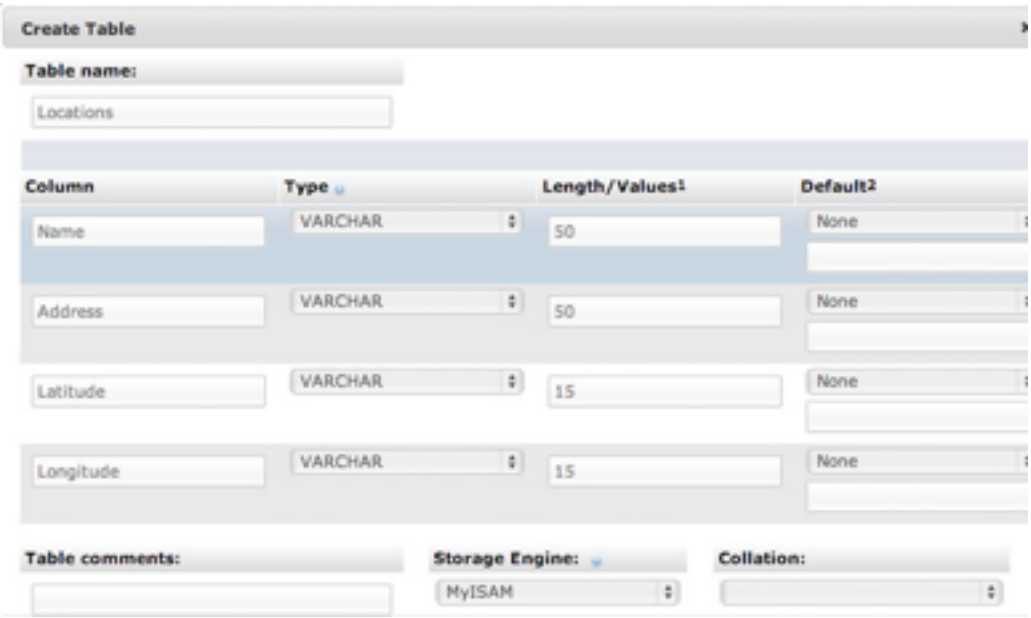
- Update, tar en "user" som parameter och uppdaterar alla fälten i databasen för just den user'n. True returneras om kontakten med databasen gick bra, annars returneras false.
- Refresh är två funktioner, en som tar en integer och en som tar en sträng som parameter. Refresh returnerar alltid en "self" som skapas av informationen som finns på platsen i databasen där parametern stämmer överens.
- Search, tar in en sträng som parameter, där strängen representerar användarnamnet eller emailadressen beroende på om ett @ eller inte finns med. En bool kommer returneras.
- Login, tar två strängar som parametrar och returnerar en user. Strängarna ska representera användarnamnet och dess lösenord. Kommandon för att söka igenom databasen och jämföra lösenordet skickas till PHP-skriptet, om det stämmer överens skapas en "self" genom att refresh kallas med användarnamnet och detta returneras. Annars returneras en ”not_complete”.
- Create, tar in en parameter för varje fält i databasen och returnerar en ”user”. Search används för att kolla om användarnamnet och emailadressen är lediga i databasen. Om ledigt skickas kommandon till PHP-skriptet om vilka värden som ska ingå. Sedan kallas en refresh för användarnamnet som returneras.

3.2.3. DATABAS

En databas av typen MySQL kommer att användas för att lagra data externt för att varje användare ska kunna komma åt respektive data oavsett på vilken iOS-enhet som används. Applikationen skickar förfrågningar till ett PHP-script som som finns på samma server som MySQL-databasen ligger på.



PHP-scriptet innehåller inloggningsuppgifterna till databasen samt funktioner för att skicka förfrågningar till databasen. Det gör att man enkelt kan kommunicera med databasen genom PHP-kommandon som fungerar med MySQL och underlättar programmeringen. Datan som skall lagras i kolumnerna (Under ”Column” i Figur 4) är uppgifter om varje användare i form av: användare-id (int), användarnamn(string), förnamn och efternamn(string), e-postadress(string) och lösenord(krypterad string).



Figur 4 - En MySQL-tabell som innehåller variabler
Källa: http://codewithchris.com/iphone-app-connect-to-mysql-database/

Det kommer lagras i tabeller i MySQL-databasen för att enkelt kunna komma åt rätt data för rätt användare. Tabellens namn kommer vara ”Users” där alla användare kommer att lagras tillsammans med respektive data.

Om tid återstår (Börkrav, se kravspecifikationen) skall även användarens skuldhistorik lagras i databasen så man kan få en överblick över alla skulder man haft tidigare.

Databasen kommer att lagra värdena med rätt typ så att den direkt kan användas av applikationen.