Användarfall – lägga upp ny användare
Systemet möjliggör uppläggning av nya användare. Användare kategoriseras efter följande behörigheter:

1. Administratör
2. Domare
3. Gymnast
4. Ledare

Följande uppgifter ska finnas kopplade till användare:

1. Förnamn
2. Efternamn
3. Personnummer
4. E-post/användarnamn
5. Lösenord
6. Behörighet

Följande krav finns kring administrationen av användare:

1. Lägga till användare
2. Ändra befintlig användare
3. Ta bort befintlig användare

Syftet med användare:

1. Inloggning.
2. Att ge olika behörigheter relevant åtkomst till specifika funktioner.
3. Att koppla individuella resultat till användare.

Användarfall scenario

Villkor:
Före: Användarkonto med behörighet ”Administratör”.
Efter: Användare är upplagd i systemet och kan logga in i systemet.

Primärt flöde
1. Loggar in.
2. Väljer funktionen lägga till ny användare.
3. Fyller i obligatoriska uppgifter. Se ovan krav på uppgifter kopplade till användare.
4. Förhandsgranskar ny användare.
5. Sparar användaren.

Alternativa flöden
1a. Administratören har fel behörighet och kommer inte åt funktionen lägga till ny användare. Administratören får kontakta ”masteradmin” eller utvecklarna för korrigering.
1b. Loggar in med fel användarnamn och/eller lösenord. Möjlighet att återställa lösenord.
4a. Något i uppgifterna är fel. Administratören väljer att avbryta och kommer tillbaka till formuläret.
5a. Användaren finns redan. Går inte att spara. Relevant meddelande visas för administratören.
5b. Uppgifter saknas. Går inte att spara. Relevant meddelande visas för administratören och möjlighet att komplettera uppgifterna.

Utökat flöde
Användaren finns i systemet. Administratören vill:

1. Radera användare
2. Ändra användarens uppgifter

Övergripande Planering iteration 1-3

Målet för iteration 1-3 är att möta kravet kring det primära flödet, det utökade flödet och det alternativa flödet enligt punkt 4a.

- Användare med behörighet Administratör ska kunna lägga upp, ändra och ta bort en användare av behörighet Domare/Ledare/Administratör.
- Ingen vikt kommer i detta skede att läggas på grafisk layout.
- Vissa UI element kommer att läggas in för att lättare få en bild av flödet.
- Användaruppgifter kommer att lagras i en databas. Databasen ligger i detta skede lokalt för att senare flyttas till server.
- Tester kommer ske löpande och loggas ut antingen i konsol till utvecklare eller till användare via meddelande i webbfönster. För mer detalj om testerna se respektive iterationsplanering.
- Systemet kommer att byggas som ett webbaserat verktyg med Javascript, HTML och CSS.

Övergripande planering för projektet

Uppgift	Tidsåtgång	Dag	Avvikelse
Användarfall	1 timme	MÅNDAG 6/1	OK
Övergripande planering	1,5 timme	MÅNDAG 6/1	OK
Övergripande reflektion	1 timme	TORS 15/1	Datum stämmer inte pga sjukdom.
Iteration 1	6,25 timmar	TISDAG 7/1	Använde en kvart av felmarginaltiden så den totala tidsåtgången blev 6 timmar.
Iteration 2	15 timmar	TORS 8/1-LÖRD 10/1	Använde inte mer tid, men fick ett helt annat resultat än planerat. Se reflektion.
Iteration 3	7,5 timmar	MÅND 12/1 - TISD 13/3	8 timmar. 0,5 timmar mer på reflektionerna. Hade ingen tid kvar att lägga till felmarginal. Datumen stämmer inte pga sjukdom.
Totalt	32,25 timmar		32,75

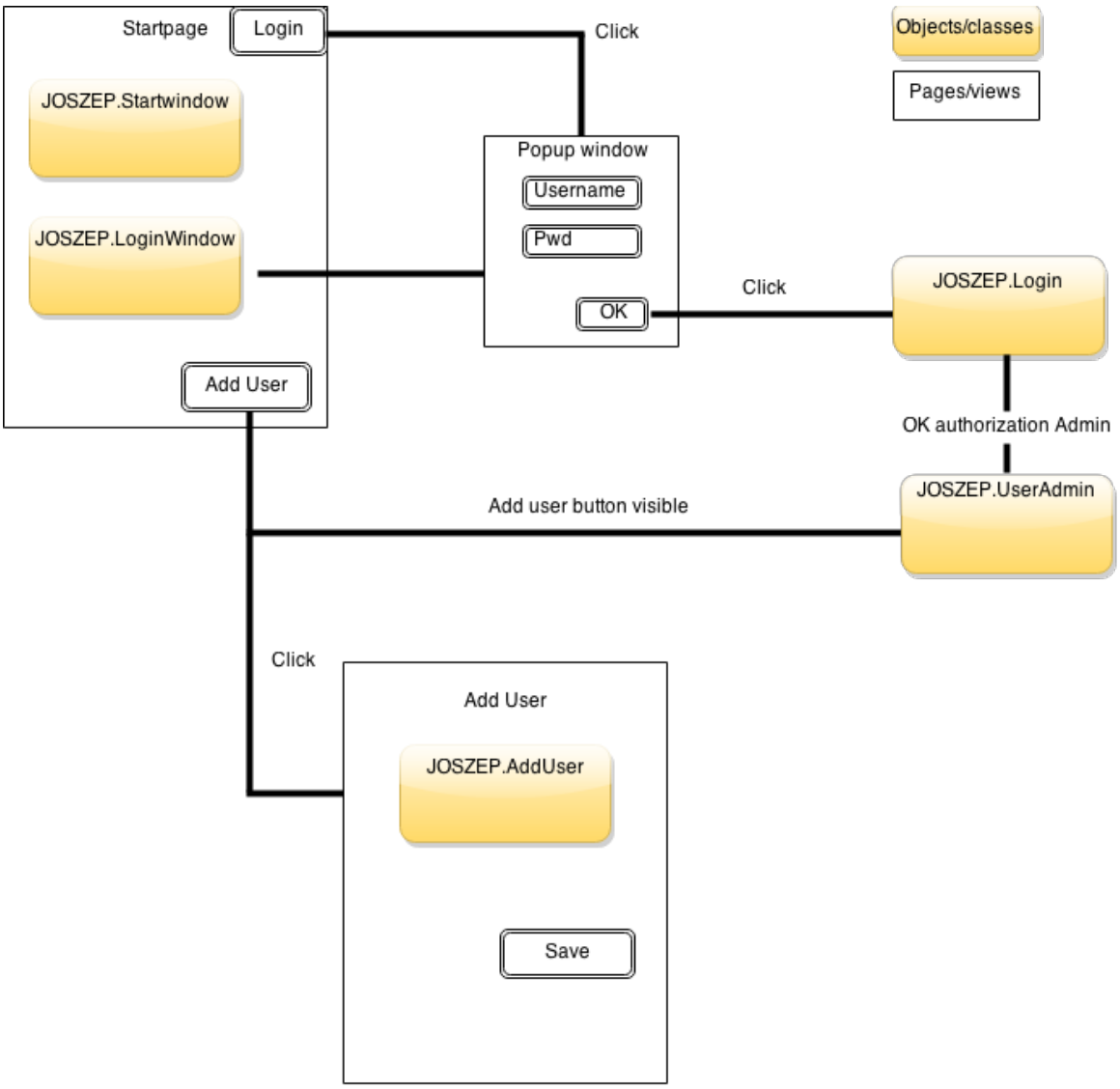
Iteration 1
- Skapa alla objekt/klasser enligt designdiagrammet.
- Lägga in grundläggande kod i alla filer.
- Skapa HTML struktur för index.html och adduser.html
- Inga databaser kommer att skapas i denna iteration. Login-funktionen testas mot ett statiskt objekt.
Data: Användarnamn: js@powerapp.se Lösenord: szepanski1

Index.html
Html-element: Knapp LOGIN.
Eventhanterare LOGIN: När användaren klickar på LOGIN visas ett popupfönster.

Html-element: Popupfönster. Visas när användaren klickat på knappen login. Består av ett fönster med ett inputfält för användarnamn, ett inputfält för lösenord och en knapp OK.
Eventhanterare OK: När användaren klickar OK valideras uppgifterna mot ett temporärt skapat objekt som håller en användare. Är något fält tomt meddelas användaren och ges ett nytt försök. Är något fält fel meddelas användaren och ges ett nytt försök. Är uppgifterna korrekta kommer användaren in i systemet. Popupfönstret försvinner och uppgifterna i inputelementen raderas.

Html-element: Efter lyckad login kommer användare av typen Administratör se en knapp LÄGG TILL ANVÄNDARE.
Eventhanterare LÄGG TILL ANVÄNDARE: Tar användaren vidare till sidan där han/hon kan administrera användare (adduser.html).

Adduser.html
Html-element: Formulär för inmatning av uppgifter kopplade till ny användare. Ska innehålla följande element; Förnamn, Efternamn, Personnummer, Email, Lösenord, Behörighet och knapp SPARA.
Lista med befintliga användare. Tillhörande knappar ÄNDRA och RADERA.
Eventhanterare: Inga eventhanterare kommer skapas i denna iteration.


Startpage
Login
Click
Objects/classes
Pages/views
JOSZEP.Startwindow
Popup window
Username
Pwd
JOSZEP.LoginWindow
OK
Click
JOSZEP.Login
Add User
OK authorization Admin
JOSZEP.UserAdmin
Add user button visible
Click
Add User
JOSZEP.AddUser
Save

Planering/tidslogg iteration 1			
Uppgift	**Tidsåtgång**	**Dag**	**Avvikelse**
Planering/krav	1 timme	TISDAG 7/1	OK
Design	1,5 timme	TISDAG 7/1	OK
Skapa js-filer och html-filer med grundläggande kod	0,45 timmar	TISDAG 7/1	Skulle klarat mig på mindre tid, men fastnade på att jag inte kunde komma vidare från startsidan till loginsidan. Visade sig vara ett tekniskt problem med cloud9
Startsidan html element	1 timme	TISDAG 7/1	OK
Eventhantering login + popup fönster	0,25 timmar	TISDAG 7/1	OK
Eventhantering Lägg till användare	0,25 timmar	TISDAG 7/1	OK
adduser html element	1 timme		Tog 1,25 timmar. Hade inte så bra koll på att skapa ett formulär.
Felmarginal	0,5 timmar		Använde 15 minuter till ovanstående
Totalt	6,25 timmar		6 timmar

Testning

Testning har skett i samband med varje moment. I denna iteration har det gått ut på att logga ut knapparnas eventhantering och var i systemet användaren befinner sig. Detta för att säkerställa att alla klasser/objekt/sidor fungerar och läses in korrekt. Se konsolen för utloggning av meddelanden.

Ex på testmeddelanden:
console.log("Starting page that can be viewed by everyone");
console.log("You´ve pressed the login button");
console.log("Sending logininformation");
console.log("You´re viewing this page as Admin");
console.log("You´ve pressed add user button");

Reflektion

Den här iterationen gick ungefär som beräknat. Just kodningen av dessa delar känner jag mig relativt hemma på och det gjorde det enklare att förutse hur lång tid allt skulle ta. När jag avsatte tid för varje moment i iterationen tyckte jag att jag tog i lite väl, men av erfarenhet jag fått under kursens gång så vet jag att ting tar tid. Även enklare saker som att lägga upp filer och skriva grundläggande kod. Med andra ord var min tidsuppskattning realistisk. För mig är det väldigt bra att börja med att lägga in UI element i koden. På så sätt kunde jag lättare se flödet och snabbt fånga upp vad jag missat eller vad som är onödigt. I det här fallet hade jag bara tänkt ha ett loginobjekt, men det slutade med att jag delade upp det i ett loginfönsterobjekt och ett loginobjekt för själva funktionen. Sen är det ju alltid det där med oförutsedda händelser, men tack vare att jag la in tid för felmarginal klarade jag iterationen på planerad tid.

Iteration 2
Under denna iteration kommer jag fokusera på datastrukturen i systemet och kraven som rör att kunna spara, ändra och ta bort en användare.

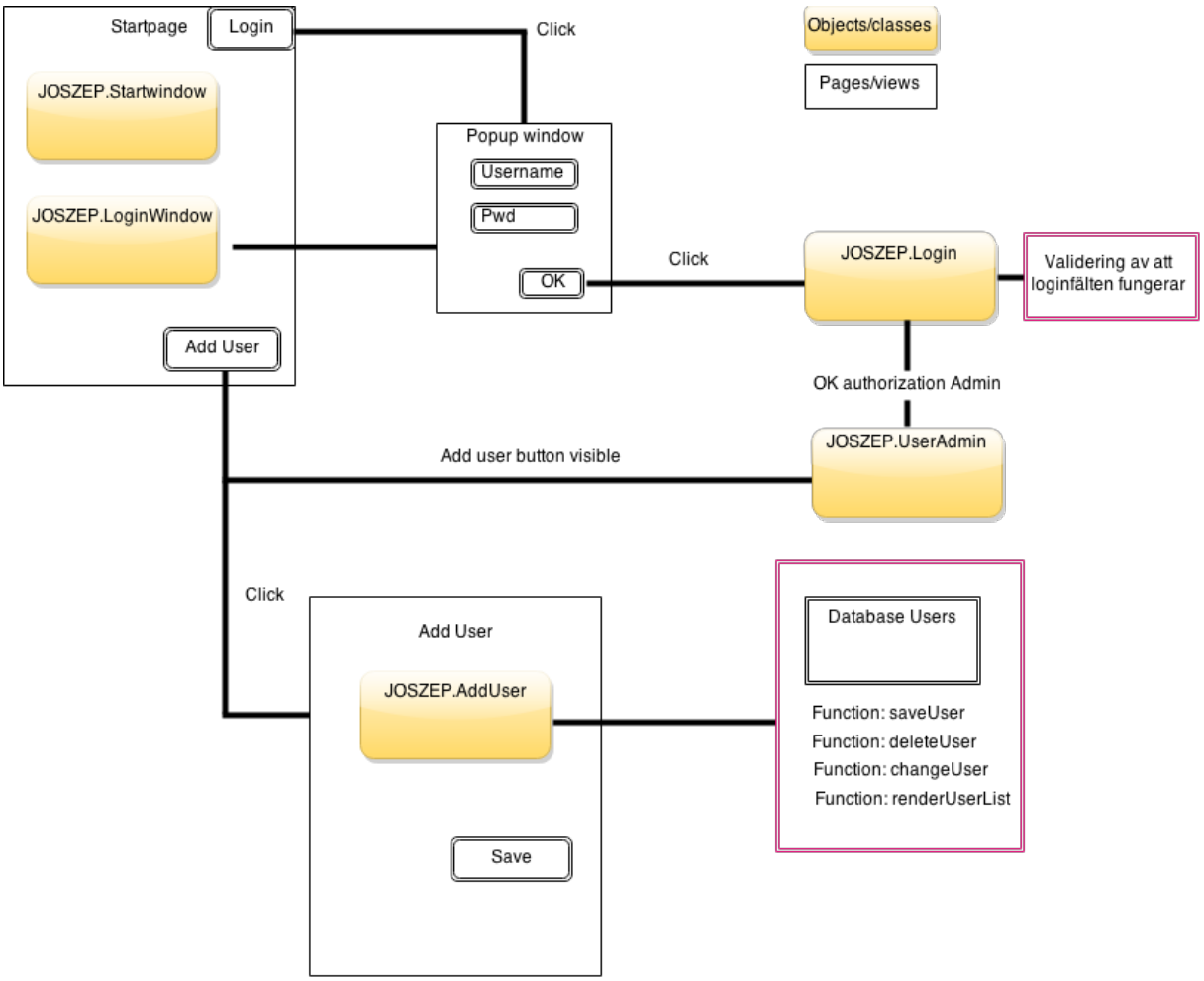
- Skapa en databas baserad på uppgifter i formuläret.
- Skapa eventhantering till knapparna
- Ingen validering/kontroll av inmatade uppgifter
- Ingen kontroll om användaren redan finns i systemet.
- Validera användaruppgifter mot databas vid login.

<u>Eventhantering till följande element:</u>

SPARA – När användaren klickar på knappen sparas inmatade uppgifter i databasen. Personnummer används som primary key. I detta skede kommer inga inmatade uppgifter att kontrolleras/valideras utan jag förutsätter att allt är inmatat korrekt.

ÄNDRA – När användaren klickar på knappen syns vald användares uppgifter i inputfälten i formuläret. Användare kan ändra någon/några uppgifter och sedan SPARA användaren igen. Ändrade uppgifter ändras i databasen.

TA BORT – När användaren klickar på knappen syns en dialogruta. Användaren bekräftar att användaren ska raderas. Vid bekräftelse raderas användaren. Posten raderas från databasen.


Startpage
Login
Click
Objects/classes
Pages/views
JOSZEP.Startwindow
Popup window
Username
Pwd
JOSZEP.LoginWindow
OK
Click
JOSZEP.Login
Validering av att loginfälten fungerar
Add User
OK authorization Admin
JOSZEP.UserAdmin
Add user button visible
Click
Add User
Database Users
JOSZEP.AddUser
Function: saveUser
Function: deleteUser
Function: changeUser
Function: renderUserList
Save

Planering/tidslogg iteration 2			
Uppgift	**Tidsåtgång**	**Dag**	**Avvikelse**
Planering/krav	0,5 timmar	TORSDAG 8/1	OK
Förberedelser/reesearch	3 timmar	TORSDAG 8/1	OK. Hade kunnat lägga mer tid, men var tvungen att sätta igång. Research handlade mest om lokala databaser som jag aldrig provat på förr.
Design	0,5 timmar	TORSDAG 8/1	OK
Hämta värden i alla inputfält i	1 timme	TORSDAG 8/1	OK
Skapa en databas och hantering för att spara användare i databasen	4 timmar	FREDAG 9/1	Samlad avvikelse för allt som hörde till databashantering. Fick inte alls ordning på det och när jag använt alla timmar utan att lyckas tog jag ett beslut om att tänka om. Se reflektion.
Funktion för att radera användare	1 timme	LÖRDAG 10/1	Se ovan.
Funktion för att ändra användare	1 timme	LÖRDAG 10/1	Se ovan.
Validering av loginuppgifter	2 timmar	LÖRDAG 10/1	Se ovan.
Felmarginal	2 timmar	LÖRDAG 10/1	Använde dessa två timmar för att ersätta databasen med en annan lösning.
Totalt	15 timmar		15 timmar

Testning

Skedde precis som i iteration 1 genom att logga ut i konsolen. Testningen har skett löpande efter varje nytt tillägg för att bekräfta funktionen. I denna iteration blev det inte mycket bekräftelse förrän jag la om taktiken.

Reflektion

Ja, det här var en intressant upplevelse. Har aldrig tidigare i sådana här sammanhang misslyckats med att klara det jag satt mig för. Trots att jag vid min research inte tyckte att det verkade speciellt komplicerat att fixa en databaslösning så visade det sig vara det. I alla fall för mig med min erfarenhet. Med facit i hand skulle jag ha testat databashanteringen på en väldigt enkel nivå INNAN jag planerade iterationerna. Då hade jag kommit till den slutsatsen att jag antingen behövde avsätta mer tid till denna iteration eller ta in någon annan med mer erfarenhet. Som det är nu kommer ”kunden” ändå kunna ta ställning till om det är rätt användaruppgifter som ska matas in och om delvis lösta funktionen fungerar som de tänkt. Vid närmare eftertanke skulle jag tänkt så här från början. Att bara spara uppgifterna tillfälligt i ett objekt för att kunden ska ha något att ta ställning till innan jag lägger för mycket tid på att bygga upp en databashantering. Databaslösning kommer ju att behövas på flera ställen i systemet och då är det en planeringsprocess i sig.

Iteration 3
I den här iterationen ska vi lägga till kraven att det ska renderas en användarlista som syns på Add User sidan. Admin ska också kunna förhandsgranska inmatade uppgifter innan användaren sparas eller gå tillbaka och göra förändringar.

- Förhandsgranskning av inmatade uppgifter.
- Spara eller ändra inmatade uppgifter
- Rendera alla användare i en lista synlig på sidan.
- Ingen validering av inmatade uppgifter.

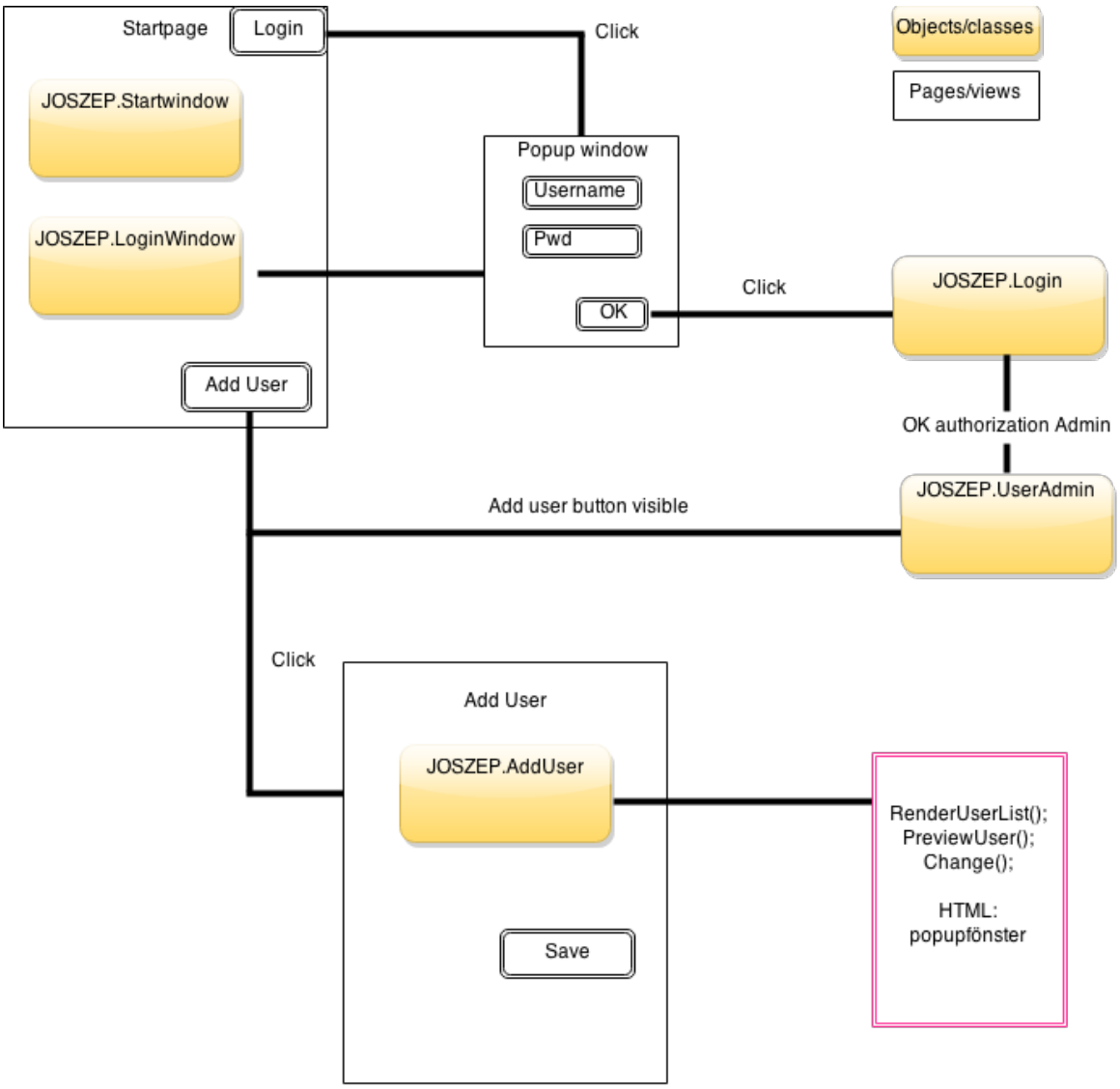
Html-element: Popupfönster som visar inmatade användaruppgifter. Knappen SPARA byts ut mot FÖRHANDSGRANSKA. Popuppfönstret ska innehålla alla personuppgifter på en användare, kapp ÄNDRA och KNAPP spara.

Eventhantering:
FÖRHANDSGRANSKA: När användaren har fyllt i alla uppgifter och klickar på FÖRHANDSGRANSKA visas ett popupfönster med alla uppgifter.

ÄNDRA: Om användaren väljer att ÄNDRA försvinner popupfönstret och formulär syns med inmatade uppgifter.

SPARA: Popupfönstret försvinner och alla inputfält töms. Listan med användare uppdateras.

RenderUserList(); Alla användare i systemet visas i en lista på sidan.


Startpage
Login
Click
Objects/classes
Pages/views
JOSZEP.Startwindow
Popup window
Username
Pwd
JOSZEP.LoginWindow
OK
Click
JOSZEP.Login
Add User
OK authorization Admin
JOSZEP.UserAdmin
Add user button visible
Click
Add User
JOSZEP.AddUser
RenderUserList();
PreviewUser();
Change();
HTML:
popupfönster
Save

Planering/tidslogg iteration 3			
Uppgift	**Tidsåtgång**	**Dag**	**Avvikelse**
Planering/krav	1 timme	MÅNDAG 12/1	OK
Design	0,25 timmar	MÅNDAG 12/1	OK
HTML popupfönster	1,5 timmar	MÅNDAG 12/1	OK
Funktion förhandsgranskning	1 timme	MÅNDAG 12/1	OK
Funktion ändra från förhandgranskning	0,5 timmar	TISDAG 13/1	OK
Funktion RenderUserList	1,75 timmar	TISDAG 13/1	Fick lite problem med att ta bort rätt användare från listan annars hade det gått fortare.
Sammanställa reflektioner alla iterationer	1,5 timmar	TISDAG 13/1	2 timmar
Totalt	7,5		8

Testning

Skedde precis som i iteration 1 genom att logga ut i konsolen. Testningen har skett löpande efter varje nytt tillägg för att bekräfta funktionen. I denna iteration blev det inte mycket bekräftelse förrän jag la om taktiken.

Reflektion

Det var lite mentalt knepigt att ställa om efter misslyckandet med att lösa databasproblemet. För syftet att kunden skulle kunna ta ställning till funktionerna fungerade en icke persistent array lika bra som en databas, men det blev svårare att programmera eftersom huvudet var inställt på en annan lösning från början. Annars gick det bra och i stort sett enligt tidsplan. I denna iteration jobbade jag med lösningar vi har jobbat en del med i webbteknik så jag var övertygad om att det inte skulle ställa till allt för stora problem för mig.

Övergripande reflektion
Generellt sett skulle jag velat ha gjort användarfallet mer komplett över de tre iterationerna med validering av inmatade uppgifter. Det hade också varit roligt att klara av databasdelen även om det inte hade passat in i ett så tidigt skede av projektet. Men för egen del hade det varit tillfredsställande att lösa det. Med så begränsad tid och begränsad erfarenhet fick jag helt enkelt lägga mig på en nivå jag realistiskt skulle klara av.

Det är också knepigt att vara ensam. Man inser verkligen styrkan i att vara flera utvecklare i ett projekt med olika kompetenser. Lyckan vore ju att kunna plocka in de personerna med för projektet rätt kunskaper. Vilken styrka det skulle ge och jag kan tänka mig större lönsamhet.

Allt eftersom jag jobbade på dök det upp nya saker som skulle kunna förbättra systemet. Vilket är en stor del av syftet med att jobba i den här processmodellen. Det var intressant att upptäcka att det faktiskt fungerar i verkligheten. Tex så kändes det helt plötsligt som ett krav att det måste finnas möjlighet att söka bland upplagda användare. Administratören kan ju inte gå igenom en hel lista varje gång de ska ändra eller ta bort en användare.

Det var också svårt att hitta rätt nivå på iterationslagren. Vad är att ta ett för stort steg och vad är ett för litet. Speciellt svårt blir det när man har längre tid i varje iteration. På tre veckor hinner man rätt långt om man är en grupp utvecklare. Själv skulle jag föredra kortare iterationer.