Uppgift 1- Planera

Steg	Plan. tid	Beskrivning	Verkl. tid	Problem på vägen
1	15 m	**Planera uppgiften** Förstå alla uppgifter och planera dem.	15 m	Trodde att jag förstod allt direkt, men så var inte fallet. Förstod omfattningen bättre när jag kom till uppgifterna, speciellt planeringen.
2	15 m	**Skapa reposition på github**	10 m	Inga problem
3	30 m	**Uppgift 1 – Planera iterationer (grovplanering)**	4 t	Förstår inte att jag skrev 30 min här som planerad tid. Att planera alla iterationer var väldigt tidskrävande och krävde mycket tankearbete, fastän det var grovplanering.

Planering av Iteration 1

Bakgrund:

Eftersom i det första läget handlar om att bygga upp någonting viktigt att visa för kunden så fokuserar denna iteration på just detta. Under iterationen ska ett webbgränssnitt byggas upp efter Användningsfallen:

- Användningsfall 1: Skapa användare
- Användningsfall 2: Skapa tävlingstillfälle
- Användningsfall 3: Återställa inloggningsuppgifter

Dessa användningsfall fokuserar på enstaka funktioner i systemet och därav i princip kräver en färdig bas att stå; denna bas kommer inte att implementeras under denna iteration. Denna iteration fokuserar specifikt på att endast utforma de nödvändiga webbgränssnittsdelar som krävs för var och en av användningsfallen. Dock så ska dessa ha ett gemensamt tilltalande formspråk och dela på css stilmallar och eventuella bilder. All funktionalitet från användningsfallet kommer dock inte att realiseras. Datat kommer inte att sparas någonstans, utan det kommer bara att leva kvar i minnet tills fönstret stängs.

Funktionella krav (efter prioritet):

1. Rätta till ouppklarade fel i klasserna Competition och Event som upptäcktes i enhetstesterna när objekten Event och Competition interagerade med varandra.
2. Webbgränssnitt ska utformas och kopplas till befintlig funktionalitet skriven i klasserna: Competition och Event i användningsfallet 2: Skapa tävlingstillfälle. Competition och Event klasserna måste och ska fungera ihop med webbgränssnittet. Det är detta som ska visas för kund i slutet av iterationen (likt SCRUM).

Icke funktionella krav:

- Det nya webbgränssnittet ska vara uppfattas som tilltalande och proffsigt av kund.
- Webbgränssnittet ska gå fort att arbeta i.

Tidsåtgång under iteration:

11 timmar

Delmål för iterationen:

- I slutet av iterationen ska ett färdigt webbgränssnitt presenteras för kund med funktionalitet ifrån användningsfall nr 2.
- Alla fel är åtgärdade som framkom under tidigare enhetstest.

Planering av Iteration 2

Bakgrund:

Nu när vi har ett visuellt gränssnitt som vi kan utgå ifrån sedan den tidigare iterationen så kan vi tillämpa den även på andra användningsfall. I den här iterationen så realiserar vi användningsfall 1; Skapa användare. På den tillämpar vi sedan webbgränssnittet. All funktionalitet i användningsfallet kommer dock inte att skapas. Användaren skapas bara i minnet tills fönstret stängs. Systemet skickar inte heller ut e-post eller sms sill den nya användaren.

Funktionella krav (efter prioritet):

1. Administratör kan skapa användare efter typerna: administratör, domare, medlem av klubb.
2. De nya skapade användarna ska ha uppgifter om:
    a. typ av användare
    b. Användarnamn och lösenord
    c. E-post och mobilnummer

Icke funktionella krav:

- Det nya webbgränssnittet ska vara uppfattas som tilltalande och proffsigt av kund.
- Webbgränssnittet ska gå fort att arbeta i.

Tidsåtgång under iteration:

11 timmar.

Delmål för iterationen:

- Användningsfall 1 ska kunna presenteras för kund i ett färdigt webbgränssnitt.
- Nya klasser gällande användarfallet ska vara skapade.
- De skapade klasserna och webbgränssnittet ska vara ihopkopplade och fungera.

Planering av Iteration 3

Bakgrund:

I denna iterationen förverkligas användningsfall nummer 3; där användaren kan återställa sina inloggningsuppgifter. Det visuella webbgränssnittet tillämpas som gjordes efter iteration 1 tillämpas också här.

Funktionella krav (efter prioritet):

1. En sida finns där användare kan välja att återställa sina lösenord.
2. En fungerande captcha finns där användare kan validera sin mänsklighet.
3. Användaren anger sin e-postadress eller sitt mobilnummer och informerad om att ett meddelande skickats till dem med en återställningslänk. Denna skickas dock inte i praktiken i detta iterationssteg.

Icke funktionella krav:

- Det nya webbgränssnittet ska vara uppfattas som tilltalande och proffsigt av kund.
- Webbgränssnittet ska gå fort att arbeta i.

Tidsåtgång under iteration:

10 timmar.

Delmål för iterationen:

- Användningsfall 3 ska kunna presenteras för kund i ett färdigt webbgränssnitt.
- Nya klasser gällande användarfallet ska vara skapade.
- De skapade klasserna och webbgränssnittet ska vara ihopkopplade och fungera.

Iteration 1

Kravanalys:

Analys av krav från projektplanering

Funktionellt krav, prioritet 1:
”Rätta till ouppklarade fel i klasserna Competition och Event som upptäcktes i enhetstesterna när objekten Event och Competition interagerade med varandra. ”

- Kravet är rimligt och är grunden för hela iterationen och krävs för att kunden ska kunna få en fungerande demo. Prioritet 1.

Funktionellt krav, prioritet 2:
”Webbgränssnitt ska utformas och kopplas till befintlig funktionalitet skriven i klasserna: Competition och Event i användningsfallet 2: Skapa tävlingstillfälle. Competition och Event klasserna måste och ska fungera ihop med webbgränssnittet.”

- Kravet är rimligt och måste fungera. Ny kod för interaktion mellan webbgränssnitt och klasserna Competition och Event måste skapas.

Icke funktionella krav:

”Det nya webbgränssnittet ska vara uppfattas som tilltalande och proffsigt av kund.”
”Webbgränssnittet ska gå fort att arbeta i.”

- Webbgränssnittet måste vara snyggt och enkelt att använda.

Övergripande mål för iteration:

- Skapa en fungerande demo (att visa upp för kund).

Förtydligade krav efter prioritet

1. Klasserna Competition och Event ska gå igenom enhetstesterna.
2. Ett nytt webbgränssnitt ska designas som uppfattas som professionellt av kund.
3. Webbgränssnittet kopplas ihop med Competition- och Eventklasserna och fungera felfritt, ny interaktionskod måste skapas.
4. En körbar demo ska genereras som kan visas upp för kund.
5. Webbgränssnittet ska gå fort att arbeta i.
6. Enhetstesterna för integrationstestet ska gå igenom.
7. Nya tester för ny funktionalitet ska utföras.

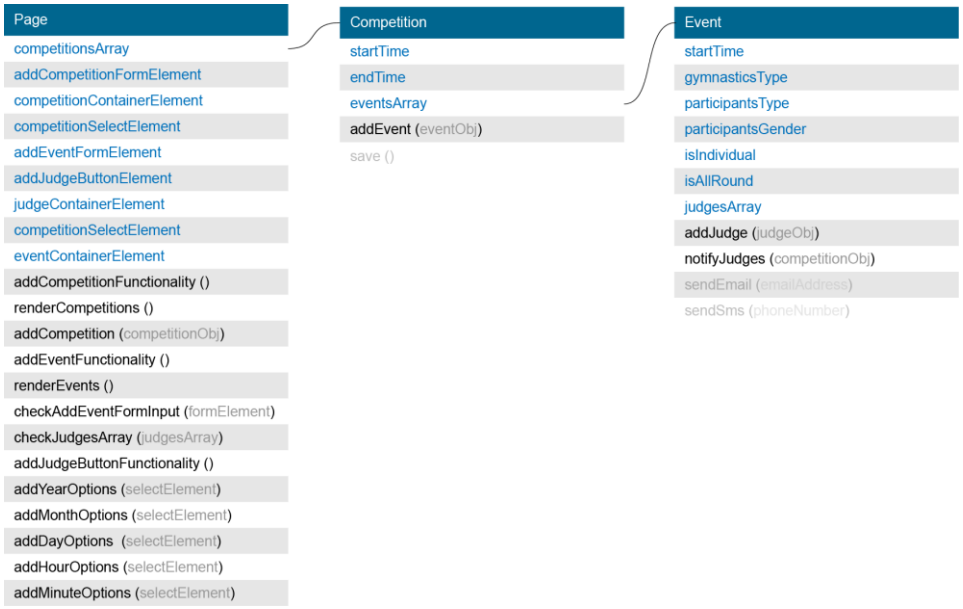
Detaljerad planering

Steg	Beskrivning	Planerad tidsåtgång	Verklig tidsåtgång	Problem på vägen
1	Analysera kraven	15 m	25 m	Gick rätt så bra.25
2	Rätta till ouppklarade fel i klasserna Competition och Event som upptäcktes i enhetstesterna.	1 t	1t 12 m	Stötte på problem som var större än väntat. Många loopar i domarjämförelse, vilka löste sig.

3	Designa ett tilltalande webbgränsnitt.	2 t	2 t 35 m	Flöt ihop med ihopkopplandet av klasserna. Designprocessen var kontinuerlig
4	Koppla ihop webbgränsnittet med funktionaliteten i Competition- och Eventklasserna.	2 t	6 t 30m	Svårt, webbgränsnittet har svårt att tillämpa klasserna på sättet som de är skrivna. En ny ”Page” klass måste skapas.
7	Testning av ny kod. Fungerar den som utlovat? Åtgärda eventuella fel.	2 t	X	Ingen testkod hann skrivas.
8	Skriv reflektion	30 m	15 m	

Klassdiagram

Page objektet med funktionalitet där Competition och Event objekten samverkar med html-sidan.



Utfall av integrationstesterna (klasserna Competition och Event)

Tidigare fanns det 5 integrationstester som inte gick igenom. Detta har nu åtgärdats i iterationen:

Test Case Id	**Förvillkor**	**Event objektet läggs till i Competition objekt genom Competition.addEvent(EventObj)**	**Förväntat resultat**	**Verkligt resultat**
1	Objekten Competition och Event skapade.	Event.startTime (1418811210000) är längre än Competition.startTime (1418821210000)	Ett felmeddelande kastas.	Ett felmeddelande kastas.
2	Objekten Competition och Event skapade.	Event.endTime (1418849210000) är längre än Competition.endTime (1418839210000)	Ett felmeddelande kastas.	Ett felmeddelande kastas.
3	Objekten Competition och Event skapade	Event objektet saknar viktig information i sina egenskaper.	Ett felmeddelande kastas.	Ett felmeddelande kastas.
4	Objekten Competition och Event skapade.	Competition objektet har redan ett identiskt Event objekt lagrat i sig.	Ett felmeddelande kastas.	Ett felmeddelande kastas.
5	Objekten Competition och Event skapade.	Competition objeketet har redan Event objekt med registerade domare i sig. Det nya Event objektet äger rum under samma tid och har samma domare registrerad. Domaren blir alltså dubbelbokad.	Ett felmeddelande kastas.	Ett felmeddelande kastas.

Reflektion

Det första som gjordes var att ett webbgränssnitt designades, detta gick relativt smärtfritt, men snart kom problemen. Vid den detaljerade planeringen av iterationen så hade jag ingen aning om att det skulle bli så mycket kod och arbete på att interagera Competition och Event objekten med sidan html-sidan. Till slut så blev jag tvungen att skapa ett nytt fristående objekt ”Page” som sköter detta istället för att baka in allt i Competition och Eventklasserna. När jag väl fick till Page klassen så fungerade saker relativt bra, men det tog mycket tid. Som klassdiagrammet ovan visar så blev slutligen Page objektet mycket större än de övriga objekten. Jag fick i princip klar en fungerande ”demo” i tid under iterationen; en demo som man kan demonstrera för kund och få feedback.
Den hittas här: http://pesola.se/webbprogrammerare/1DV404/Laboration%204/competition.html

Under arbetet så uppenbarade sig en hel del fel. Dessa har jag inte haft tid att åtgärda eller skriva testkod till. Dessutom så ser jag det som väldigt svårt att skriva testkod till stora delar av Page objektet som interagerar visuellt med html-koden. Den visuella testningen känns omätbar. Jag vet inte hur jag ska skapa mätbara tester för kod som samverkar med gränsnittet.

Under iterationen hanns inte testkod skrivas för Page klassen, detta får skjutas över till nästa iteration.

Iteration 2

Analys av föregående iteration:

- Tester på Page objektet hanns inte utformas eller köras, detta måste ske under denna iteration.

Kravanalys:

Funktionellt krav, prioritet 1:

”Administratör kan skapa användare efter typerna: administratör, domare, medlem av klubb.”

- Kravet är rimligt. En ny klass/objekt; User måste skapas.

Funktionellt krav, prioritet 2:

”De nya skapade användarna ska ha uppgifter om: typ av användare, Användarnamn och lösenord, E-post och mobilnummer”

- User objektet ska ha egenskaperna: type, username, password, email och cellphone.

Nya identifierade funktionella krav ifrån användarfallet, prioritet 3:

- Det går inte att skapa en användare med ett användarnamn som redan är taget.

Icke funktionella krav:

”Det nya webbgränssnittet ska vara uppfattas som tilltalande och proffsigt av kund.”
”Webbgränssnittet ska gå fort att arbeta i.”

- Dessa krav tillfredsställdes till största del i den föregående iterationen.

Övergripande mål för iteration:

- En demo som kan visas för kund. I demot ska man kunna skapa användare i webbgränssnittet som utvecklades i den tidigare iterationen.

Förtydligade krav efter prioritet

1. Skapa enhetstester och få dem att gå igenom för Page objektet/klassen från föregående iteration.
2. Skapa ny User klass/objekt med egenskaperna: type, username, password, email och cellphone
3. User klassens egenskaper ska varna ifall man sätter ett ogiltigt värde.
4. Webbgränssnittet kopplas ihop med User klassen och fungera felfritt, ny interaktionskod måste skapas i ett Page objekt.
5. En körbar demo ska genereras som kan visas upp för kund.
6. Enhetstester skapas för User klassen/objektet och dessa ska gå igenom.

Detaljerad planering

Steg	Beskrivning	Planerad tidsåtgång	Verklig tidsåtgång	Problem på vägen
1	Analysera kraven	15 m	10 m	Inga svårigheter
2	Skapa enhetstester för Page klassen från föregående iteration	1 t	1 t	Fick istället komplettera i testerna för Event och Competition objektet, eftersom fel upptäcktes i webbgränsnittet. Endast ett test för Page objektet
3	Rätta till eventuella fel som upptäcks under testerna	1 t	50 m	Fel som upptäcktes rättades till i Competition, Event och Page objekten.
3	Skapa User objektet	2 t	2 t 8 m	Gick (förhoppningsvis) smärtfritt.
4	Koppla ihop webbgränsnittet med funktionaliteten i User objektet (skapa i Page objektet/klassen)	2 t	6 t 17 m	Fick skapa unika Page objekt till varje sida. (Omdöpt till UserPage i nästa iteration) Ny funktionalitet i User objektet, mycket krångel med att spara användardata i localstorage.
7	Testning av ny kod. Fungerar den som utlovat? Åtgärda eventuella fel.	2 t	X	Hann inte med detta, då ett fungerande gränsnitt var viktigare.
8	Skriv reflektion	30 m	15 m	Inga problem

Utfall av tester (komplettering av föregående iteration)

Event:

Test Case Id	Förvillkor	Event objekt skapas genom konstruktorn	Förväntat resultat	Verkligt resultat
1	Inga	Event.endTime (1418811210000) är lägre än Event.startTime (1418839210000)	Ett felmeddelande kastas.	Event objektet skapas utan klagomål

- *Testfixturer – ogiltiga värden: /spec/fixtures/bad_values.json*
- *Testfixturer – giltiga värden: /spec/fixtures/event_good_values.json*

Competition:

Test Case Id	Förvillkor	Competition objekt skapas genom konstruktorn	Förväntat resultat	Verkligt resultat
1	Inga	Event.endTime (1418811210000) är lägre än Event.startTime (1418839210000)	Ett felmeddelande kastas.	Competition objektet skapas utan klagomål

- *Testfixturer – ogiltiga värden: /spec/fixtures/bad_values.json*
- *Testfixturer – giltiga värden: /spec/fixtures/competition_good_values.json*

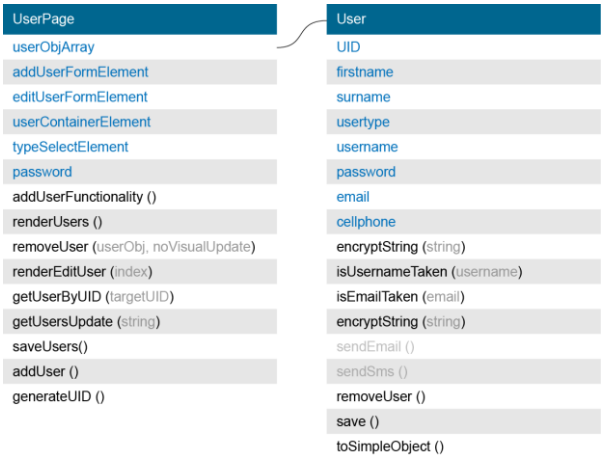
Page: (Omdöpt till UserPage i nästa iteration)

Test Case Id	Förvillkor	Ett till Comptetition objekt läggs till i Page objektet	Förväntat resultat	Verkligt resultat
1	Ett Page objekt existerar med ett Competition objekt i sig.	Event.startTime (1418811210000) Event.endTime (1418839210000) är exakt samma värden som i det tidigare inlagda Competition objektet	Ett felmeddelande kastas.	Competition objektet läggs till utan klagomål

- *Testfixturer – ogiltiga värden: /spec/fixtures/bad_values.json*
- *Testfixturer – giltiga värden: /spec/fixtures/user_page_values.json*

Klassdiagram

Dessa klasser/objekt har uppstått under iterationen: User objektet och UserPage objektet med funktionalitet där User objekten samverkar med html-sidan.



Reflektion:

När testningen Page objektet påbörjades (kvarleva från föregående iteration) så lyckades jag endast klura ut ett test för det objektet, men vid närmare visuell testning av gränssnittet så upptäckte jag två ytterligare fel i Event och Competition objekten vilka det skrevs tester för och sedan åtgärdades.

Det första jag insåg när jag väl började skapa User objektet var att jag behöver ytterligare ett Page objekt som kopplar ihop User objektets funktionalitet med webbgränssnittet. Detta gjorde att Page objektet för tävlingssidan döptes om till CompetitionPage och objektet UserPage skapades.

Utformningen av Userobjektet var till en början simpel med ett fåtal egenskaper och metoder, men när UserPage objektet skapades uppstod även nya behov för metoder och egenskaper i User objektet.

Jag låg bra till i tid och skapade därför även funktionalitet att ändra på användare och även sedan möjlighet att användarna sparas i webbläsarens localstorage. Här började problemen hopa sig. Eftersom objekten tappar alla metoder då de sparas i webbstorage så fick jag skriva om en del funktionalitet i båda klasserna för att få webbgränsnittet att fungera, detta gjorde att tiden rann iväg och inga tester hanns skrivas för de nya objekten. Jag fick i nöd och näppe klart ett fungerande demo i slutet av iterationen, därav nåddes målet för iterationen.

Demot hittas här: http://pesola.se/webbprogrammerare/1DV404/Laboration%204/user.html

Iteration 3

Analys av föregående iteration:

- Tester på UserPage eller User objekten hanns inte utformas eller köras, detta måste ske under denna iteration.

Kravanalys:

Funktionellt krav, prioritet 1:
” En sida finns där användare kan välja att återställa sina lösenord.”

- Kravet är vettigt och grunden för iterationen.

Funktionellt krav, prioritet 2:
” En fungerande captcha finns där användare kan validera sin mänsklighet.”

- Här finns det risk att eventuella problem dyker upp vid implementation.

Funktionellt krav, prioritet 3:
”Användaren anger sin e-postadress eller sitt mobilnummer och blir informerad att ett meddelande skickats till dem med en återställningslänk. Denna skickas dock inte i praktiken i detta iterationssteg.”

- En grundläggande funktion på sidan, en server krävs för att skicka e-post eller meddelanden. Ingår inte i denna iteration.

Icke funktionella krav:
”Det nya webbgränssnittet ska vara uppfattas som tilltalande och proffsigt av kund.”
”Webbgränssnittet ska gå fort att arbeta i.”

- Dessa krav tillfredsställdes till största del i den första iterationen.

Tidsåtgång under iteration:

8 timmar.

Övergripande mål för iteration:

- En demo som kan visas för kund. I demot ska användare i webbgränssnittet kunna återställa sitt lösenord.

Förtydligade krav efter prioritet

1. Skapa enhetstester och få dem att gå igenom för UserPage och User objekten från föregående iteration.
2. Skapa ny ResetPasswordPage klass/objekt med interaktionskod så användaren genom webbgränssnittet kan återställa sitt lösenord.
3. ResetPasswordPage klassen ska varna ifall användaren anger felaktiga uppgifter.
4. En körbar demo ska genereras som kan visas upp för kund.
5. Enhetstester skapas för ResetPasswordPage där det är möjligt. Dessa ska gå igenom.

Detaljerad planering

Steg	Beskrivning	Planerad tidsåtgång	Verklig tidsåtgång	Problem på vägen
1	Analysera kraven	15 m	12 m	Inga svårigheter
2	Skapa enhetstester för UserPage och User objekten från föregående iteration	2 t	3 t 56 m	Blev fler tester än väntat, fanns logisk funktionalitet i UserPage som kunde testas.
3	Rätta till eventuella fel som upptäcks under testerna	2 t	1 t 52m	Vissa fel i testet som skrevs rättade.
3	Skapa ResetPasswordPage objektet	2 t	2 t 15m	Svårt att interagera med befintliga objekt och göra ett nytt logiskt objekt.
4	Testning av ny kod. Fungerar den som utlovat? Åtgärda eventuella fel.	2 t		Inga tester kunde identifieras
5	Skriv reflektion	30 m	15 m	Inga problem

Utfall av tester (komplettering av föregående iteration)

User: (Konstruktor)

ID	Angiven UID	Angiven firstname	Angiven surname	Angiven usertype	Angiven username	Angiven password	Angiven email	Angiven cellphone	Förväntat resultat	Verkligt resultat
1	42f945dc-ec71-436d-830e-...	Anders	Persson	user	anders.persson	mysupersecretpassword	anders@persson.se	0731234567	Användare skapas	Användaren skapas
2	42f945dc-ec71-436d-830e-...	Anders	Persson	user	anders.persson	mysupersecretpassword	anders@persson.se	1.343634233	Användaren får fel-meddelande	Användaren får fel-meddelande
3	42f945dc-ec71-436d-830e-...	Anders	Persson	user	anders.persson	mysupersecretpassword	i3489s@!¤#02349032334co#=956?´		Användaren får fel-meddelande	Användaren får fel-meddelande
4	42f945dc-ec71-436d-830e-...	Anders	Persson	user	anders.persson	-24			Användaren får fel-meddelande	Användaren får fel-meddelande
5	42f945dc-ec71-436d-830e-...	Anders	Persson	user	1.343634233				Användaren får fel-meddelande	Användaren får fel-meddelande
6	42f945dc-ec71-436d-830e-...	Anders	Persson	1.343634233					Användaren får fel-meddelande	Användaren får fel-meddelande
7	42f945dc-ec71-436d-830e-...	Anders	2424						Användaren får fel-meddelande	Användaren får fel-meddelande
8	42f945dc-ec71-436d-830e-...	2424							Användaren får fel-meddelande	Användaren får fel-meddelande
9	1.343634233								Användaren får fel-meddelande	Användaren skapas

- *Testfixturer – ogiltiga värden: /spec/fixtures/bad_values.json*
- *Testfixturer – giltiga värden: /spec/fixtures/user_good_values.json*

User: (andra tester)

Test Case ID	Förvillkor	Följande sker	Förväntat resultat	Verkligt resultat
1	User objeket är skapat	User.UID = 42f945dc-ec71-436d-830e-161138f48e49	Ett felmeddelande kastas.	Ett felmeddelande kastas.

2	User objekt är skapat och sedan borttaget med metoden removeUser();	new User() med giltiga argument	Användaren skapas	Ett felmeddelande kastas.
3	User-objektet är skapat	Användarens sparas i localstorage med User.save()	Användaren sparas i localstorage	Användaren sparas inte i localstorage

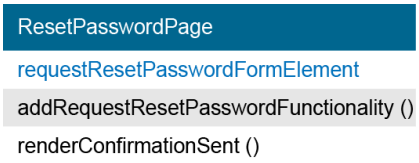
- *Testfixturer – ogiltiga värden: /spec/fixtures/bad_values.json*
- *Testfixturer – giltiga värden: /spec/fixtures/user_good_values.json*

UserPage:

Test Case ID	Förvillkor	Följande sker	Förväntat resultat	Verkligt resultat
1	UserPage objektet är skapat. Ett User objekt är skapat.	UserPage.addUser(user) metoden körs	Ett user objekt finns i UserPage objektet.	Ett user objekt finns i UserPage objektet.
2	UserPage objektet är skapat med en User lagrad i sig.	UserPage.saveUsers() metoden körs	User objekten sparas i localstorage	User objekten sparas i localstorage
3	UserPage objektet är skapat.	UserPage.getUsersUpdate() körs.	UserPage objektet har en uppdaterad array med Users hämtat ifrån localstorage	UserPage objektet har en uppdaterad array med Users hämtat ifrån localstorage
4	UserPage objektet är skapat med en User lagrad i sig.	UserPage.getUserById(UID) metoden körs med samma UID sträng som User objektet som finns lagrad i UserPage objektet.	Rätt User objekt returneras	Ett internt fel uppstår i metoden
5	UserPage objektet är skapat med en User lagrad i sig.	UserPage.remove(User) metoden körs.	User objektet tas bort iUserPage.userObjArray och localstorage	User objektet tas bort iUserPage.userObjArray och localstorage

- *Testfixturer – ogiltiga värden: /spec/fixtures/bad_values.json*
- *Testfixturer – giltiga värden: /spec/fixtures/user_page_values.json*

Klassdiagram



Reflektion:

Testerna tog upp den största tiden i denna iteration, vilket gick över tid. Det var många egenskaper och metoder i User och UserPage objekten som var testbara. 4 fel påträffades i de 19 tester som skapades, dessa åtgärdades inom sin tidsram.

Nya objektet ResetPasswordPage skapades som interagerar användaren genom webgränssnittet. Enhetstester för objektet kunde inte identifieras. Koden berör inte logik utan fångar endast upp uppgifterna användaren anger och ser om dessa hittas i localstorage, därefter skickas e-post eller sms till användaren med återställningslänk (ej implementerat).

Problemen med localstorage fortsatte även här. Några små ändringar i metoder fick göras i User objekten (sendEmail och sendSms) så att dessa metoder går att anropa med värden som inte berör själva objektet. Det bästa hade varit att fixa någon sorts konvertering att localstorage objekten konverteras tillbaka till User objekt vid inläsning, men detta fick jag aldrig till att fungera bra.

Demot hittas här:
http://pesola.se/webbprogrammerare/1DV404/Laboration%204/resetpassword.html

Uppgift 3 – Reflektion

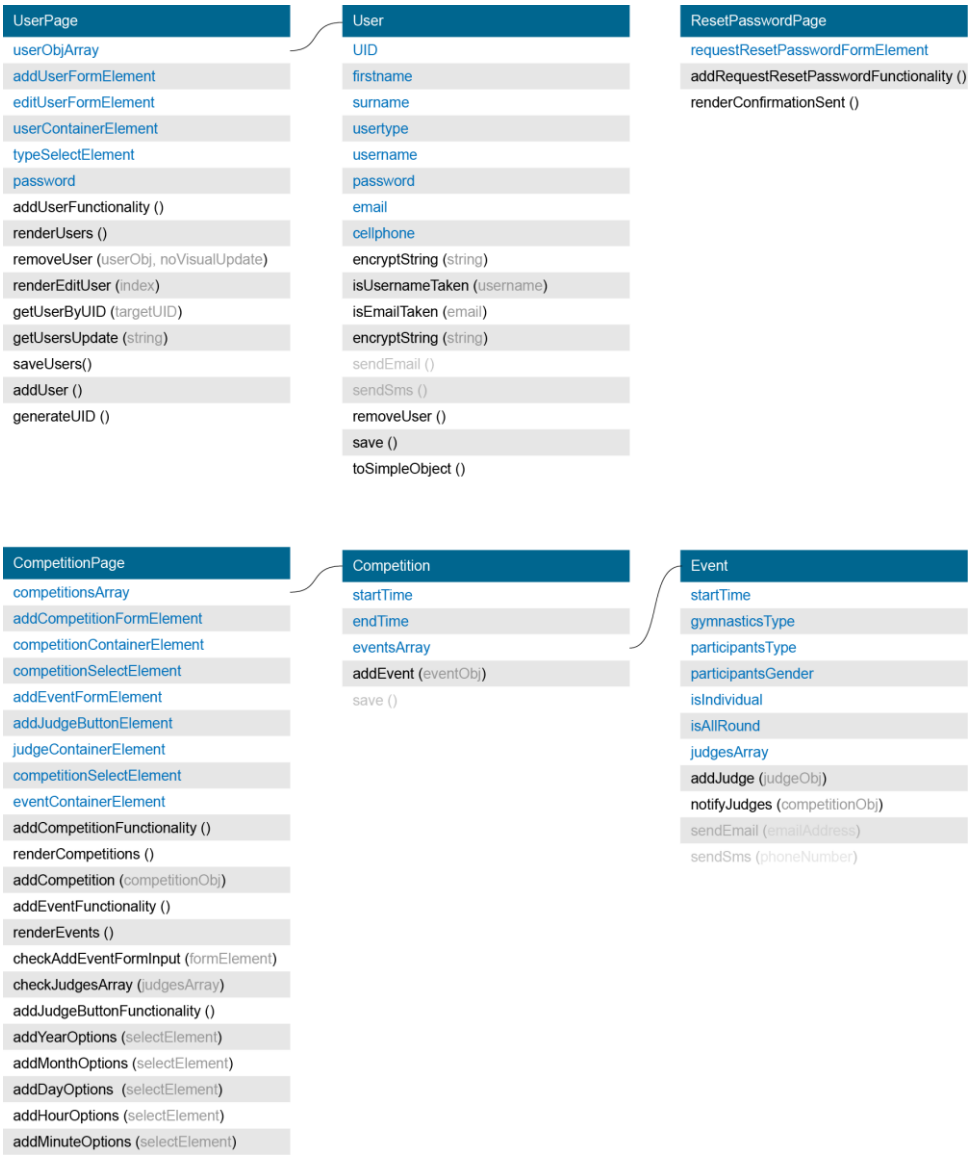
Från början till slut så hade jag ett kundfokuserad syn på alla iterationer, jag ville likt SCRUM ha en fungerande demo att visa upp för kund efter varje iteration. Då jag arbetade hårt för att få denna demo klar till slutet av iterationerna så fick enhetstestningen lida för detta och i båda fallen förflyttades den till nästa iteration. Mycket av en sorts testning skedde under utveckling av gränssnittet, denna var svår att dokumentera, men det kom det ju ganska snabbt fram ifall någon kod inte fungerade som den skulle. Men många osynliga fel upptäcktes genom enhetstestningen.

Det är nu när man arbetar iterativt som man ser styrkan av testning. Enhetstestningen känns som en väldigt bra grund att ha till koden. Det fungerar som en trygghet och det är där man verkligen ser om ändringar som man blir tvungen att göra vid senare iterationer fungerar, eller om de gör skada.

Fast man i iterationerna gjorde detaljerade planer på hur själva iterationen skulle läggas upp så upptäckte man så fort man började koda att planeringen man gjort inte fungerade i verkligheten. I alla iterationer, utom den sista var det någon viktigt grund jag glömt; Page klassen i den första iterationen, Hur många tester det blev i den andra iterationen. Det är nog bara genom erfarenhet att jobba på det här sättet som man bättre lär sig uppskatta hur lång tid saker verkligen tar och hur stor buffert av tid man realistiskt ska planera för att få alla iterationer att gå ihop i slutändan.

Samlad dokumentation för inlämning

Klassdiagram efter 3 iterationer



Testsviten efter 3 iterationer

42 specs, 0 failures

- Competition
  - should be created with correct properties from good constructor values
  - should throw an error when created with bad constructor endTime argument
  - should throw an error when created with bad constructor startTime argument
  - should throw an error when created with an endTime that is before startTime

- Event
  - should be created with correct properties from good constructor values
  - should throw an error when created with bad constructor startTime argument
  - should throw an error when created with bad constructor endTime argument
  - should throw an error when created with bad constructor gymnasticsType argument
  - should throw an error when created with bad constructor participantsType argument
  - should throw an error when created with bad constructor participantsGender argument
  - should throw an error when created with bad constructor isIndividual argument
  - should throw an error when created with bad constructor isAllRound argument
  - should throw an error when created with bad constructor judgesArray argument
  - should throw an error when created with an endTime that is before startTime
  - should have correct judgesArray after using 'addJudge' method
  - should throw an error after using 'addJudge' with faulty argument.
  - should should run 'sendEmail' method or 'sendSms' method correct after using 'notifyJudges' method

- Integration
  - should throw an error when Competition.addEvent(Event) and Event.startTime < Competition.startTime
  - should throw an error when Competition.addEvent(Event) and Event.endTime > Competition.endTime
  - should throw an error when Competition.addEvent(Event) and Event has missing properties
  - should throw an error when Competition.addEvent(Event) and Event is allready present in the Competition object.
  - should throw an error when Competition.eventsArray has an Event object with a judge assigned to it and a new event is added over the same period of time with the same judge.

- CompetitionPage
  - should throw an error when trying to add a Competition with the same startTime and endTime as another competition that already exists.

- User
  - should be created with correct properties from good constructor values
  - should throw an error when created with a bad constructor UID argument
  - should throw an error when and trying to change UID after User has been constructed.
  - should throw an error when created with a bad constructor firstname
  - should throw an error when created with a bad constructor surname
  - should throw an error when created with a bad constructor usertype
  - should throw an error when created with a bad constructor username
  - should throw an error when created with a bad constructor password
  - should throw an error when created with a bad constructor email
  - should throw an error when created with a bad constructor cellphone
  - should throw an error when created with a username that is already in use.
  - should throw an error when created with a email that is already in use.
  - should not throw an error when trying add a user that was deleted.
  - should have user stored i localstorage after using user.save() method.

- UserPage
  - should store user object after UserPage.addUser() method.
  - should store array of user objects after UserPage.saveUsers() method.
  - should fetch array of user objects after UserPage.getUsersUpdate() method.
  - should return the right user object after UserPage.getUserById(UID) method.
  - should remove the user object from both localstorage and userPage.userObjArray after running UserPage.removeUser(userObj, true) method.

Tidslogg

Datum	Timmar	Arbetsbeskrivning
2015-12-21	4 t 30 m	Planering
2015-12-22	8 t	Iteration 1
2015-12-23	6 t	Iteration 1, Iteration 2
2015-01-24	8 t	Iteration 2, Iteration 3
2015-01-25	6 t	Iteration 3

Krav efter analys och prioritetsordning från iterationerna

Krav Iteration 1

1. Klasserna Competition och Event ska gå igenom enhetstesterna.
2. Ett nytt webbgränssnitt ska designas som uppfattas som professionellt av kund.
3. Webbgränssnittet kopplas ihop med Competition- och Eventklasserna och fungera felfritt, ny interaktionskod måste skapas.
4. En körbar demo ska genereras som kan visas upp för kund.
5. Webbgränssnittet ska gå fort att arbeta i.
6. Enhetstesterna för integrationstestet ska gå igenom.
7. Nya tester för ny funktionalitet ska utföras.

Krav Iteration 2

1. Skapa enhetstester och få dem att gå igenom för Page objektet/klassen från föregående iteration.
2. Skapa ny User klass/objekt med egenskaperna: type, username, password, email och cellphone
3. User klassens egenskaper ska varna ifall man sätter ett ogiltigt värde.
4. Webbgränssnittet kopplas ihop med User klassen och fungera felfritt, ny interaktionskod måste skapas i ett Page objekt.
5. En körbar demo ska genereras som kan visas upp för kund.
6. Enhetstester skapas för User klassen/objektet och dessa ska gå igenom.

Krav Iteration 3

1. Skapa enhetstester och få dem att gå igenom för UserPage och User objekten från föregående iteration.
2. Skapa ny ResetPasswordPage klass/objekt med interaktionskod så användaren genom webbgränssnittet kan återställa sitt lösenord.
3. ResetPasswordPage klassen ska varna ifall användaren anger felaktiga uppgifter.
4. En körbar demo ska genereras som kan visas upp för kund.
5. Enhetstester skapas för ResetPasswordPage där det är möjligt. Dessa ska gå igenom.