## Tidslogg

|  |  |  |
| --- | --- | --- |
| **Datum** | **Timmar** | **Arbetsbeskrivning** |
| 2014-12-16 | 2.5 | Läste och strukturerade uppgiften |
| 2014-12-17 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Uppgift 1- Planera

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Steg | Plan.  tid | Beskrivning | Verkl.  tid | Problem på vägen |
| 1 | 40 m | **Uppgift 1 - Planera**  Förstå alla uppgifter och planera dem. | 35 m | Svårt att förstå exakt vad det är som ska göras i uppgifterna. |
| 2 | 15 m | **Skapa reposition på github** | 5 m | Inga problem |
| 3 | 1 t | **Uppgift 2 – Testplan**   * Läs på om testplan. | 1 t | Svårt att skiva tester och plan innan man skriver klassen?!  Mycket att information att själv hitta. |
| 4 | 1 t | * Skriv en testplan som beskriver mitt testprojekt. | 45 m | Skriven, omständigt, kommer att behövas redigeras senare. |
| 5 | 15 m | **Uppgift 3 – Design och implementation**   * Sätt upp kodningsmiljön | 30 m | Konfigurering av en webbutvecklingsmiljö med jasmine |
| 6 | 2 t | * Designa och implementera ett användningsfall, OBS. minst två klasser. | 2 t 35 m | Gick ganska smärtfritt, hoppas att jag gjort rätt. |
| 7 | 30 m | * Dokumentera klasserna | 45 m | Klasserna dokumenterade på ett logiskt sätt. Med diagram, hade svårigheter att hitta verktyg för diagram. |
| 7 | 15 m | * Sammanställ användningsfallmodell, användningsfallsbeskrivningar, klasser med beskrivningar samt kod för inlämning. |  |  |
| 8 | 30 m | **Uppgift 4 – Enhetstestning**   * Läs på om xUnit, testsviter, testfixturer och testfall. |  |  |
| 9 | 2 t | * Gör analys av två klasser och specificera testsvit som innehåller testfixturer och testfall. * Beskriv testdata och test mot förväntat resultat, och hur jag kommit fram till detta. |  |  |
| 10 | 30 m | **Uppgift 5 Implementera testsviten och kör**   * Läs på om hur man bygger testklasser för min kod. |  |  |
| 11 | 2 t | * Bygg testklasser för mina två klasser. Exekvera testklasserna och samla in testdata. |  |  |
| 12 | 30 m | * Analysera och föreslå förbättringar |  |  |
| 13 | 30 m | **Uppgift 6 – Integrationstestning**   * Läs på om integrationstestning |  |  |
| 14 | 3 t | * Ta fram testbeskrivningar för integrationsfall |  |  |
| 15 | 30 m | **Uppgift 7 – Reflektion**   * Reflektera kring svårigheter i att planera och genomföra uppgiften. |  |  |

Total tidsåtgång ???.

# Uppgift 2- Testplan

**Produkt**

Gymnastiktävlingssystem för gymnastikligan. Version 1.

**Referenser**

1. Gymnasietävlingssystemet; Användningsfall 2 ”Skapa tävlingstillfälle för säsong”, version 1.
2. Gymnasietävlingssystemet; Supplementary specification, version 2.

**Introduktion och Bakgrund**

Den här testplanen är skapad för gymnastiktävlingssystemet ämnad att användas av gymnastikligan och alla inblandade såsom; gymnaster, domare, administratörer och administratörer.

För att kunna bygga ett bättre system och att slippa att ett högt antal fel i systemet når slutanvändaren så skriver vi här en plan för hur vi skapar tester för systemets programmerade klasser. Detta minskar risken för fel i det slutgiltiga systemet och ger en bättre användarupplevelse.

Den här testplanen berör användningsfall 2 ”Skapa tävlingstillfälle för säsong” (se referens 1) där användaren skapar ett tävlingstillfälle för en säsong.

**Testprocess Testmål (objectives)**

* Enhetstester i systemet
  + Systemet kräver att användaren är inloggad som administratör
    - Ansvarig: Johnny
    - Testtyp: Whitebox testning (Villkor)
    - Stoppregler:
      * Användaren inloggad och administratör.
  + Tävlingstillfälle är skapad innan deltävling skapas
    - Ansvarig: Johnny
    - Testtyp: Whitebox testning (Villkor)
    - Stoppregler:
      * Deltävling kan inte skapas om tävlingstillfälle inte är skapad och angiven.
  + Deltävling kan inte skapas för ett tävlingstillfälle som inte finns.
  + Alla uppgifter måste finnas för att en deltävling ska kunna skapas
    - Ansvarig: Johnny
    - Testtyp: Whitebox testning (Villkor)
    - Stoppregler:
      * Deltävling kan inte skapas om inte alla uppgifter är specificerade.
        + Starttid, sluttid, gymnastikgren, junior eller seniortävling, män eller kvinnor, allround eller individuell och domare.
  + Startpunkt för deltävlingen kan inte vara efter slutpunkten för deltävlingen.
    - Ansvarig: Johnny
    - Testtyp: Whitebox testning
    - Stoppregler:
      * Deltävling kan inte skapas om startpunkten för deltävlingen är efter slutpunkten.

**Testade artefakter**

Bokningssystemet i gymnastiktävlingssystemet.

**Rutiner för dokumentation av testresultat**

Varje aktuell testkörning loggas i en textfil information om:

* När testet exekverades
* Testets utfall
* Vem som exekverat testet

**Testomgivning**

Testsystemets omgivning består av följande hårdvara och mjukvara. En webbserver krävs. Klasserna är byggda med javascript men kan inte köras utan webbserver på grund av att testfixturer i form av JSON objekt hämtas med hjälp av ajax.

Tänk på att funktionalitet i mjukvaran kan ändras i och med nya (eller äldre) versioner av mjukvaran.

**Server**

System: Virtuell maskin i VMware ESXi   
Minne: 1 GB  
Hårddiskutrymme: 17 GB  
Operativsystem: Ubuntu 14.04 Server  
Webbserver mjukvara: nginx 1.4.6

Javascript testmjukvara: Jasmine 2.1.3  
pluginet jasmine-jquery 2.0.5  
jquery 2.1.1

**Klient**

Operativsystem: Windows 7  
Webbläsare: Chrome

# Uppgift 3- Design och implementation

## ”Klassen” Competition

Det här objektet rymmer själva tävlingstillfället som användaren skapar. Detta objekt rymmer sedan i sin tur deltävlingar i en arrayen ”eventsArray” som finns som en egenskap i objektet.

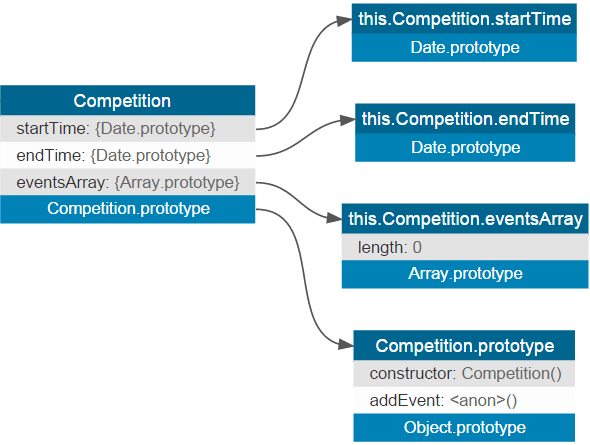
**Egenskaper med getters och setters:**

* startTime Starttiden för tävlingen i form av ett Date Objekt.
* endTime Sluttiden för tävlingen i form av ett Date Objekt.
* eventsArray Array med Event objekt (deltävlings objekt).

**Metoder:**

Metoderna ligger som en prototype, för att spara resurser ifall det någon gång i framtiden skapas många Competition objekt.

* addEvent(eventObj) Lägger till Event objekt i eventsArray:en.



## ”Klassen” Event

Det här objektet rymmer själva deltävlingen som, när den är fylld med tävlingens information i sina egenskaper ska läggas in i Competition objektets array.

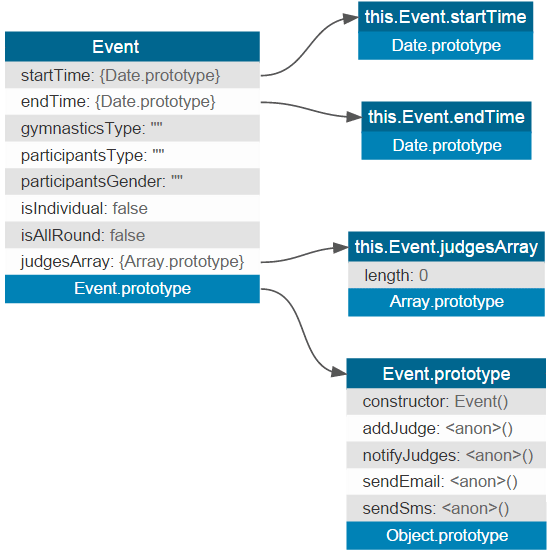
**Egenskaper med getters och setters:**

* startTime Starttiden för tävlingen i form av ett Date Objekt.
* endTime Sluttiden för tävlingen i form av ett Date Objekt.
* gymnasticsType Sträng som representerar tävlingsgren.
* participantsType Sträng som representerar om deltävlingen är för juniorer eller seniorer.
* paticipantsGender Sträng som representerar om deltävlingen är för män eller kvinnor.
* isIndividual Boolean som representerar om deltävlingen är individuell.
* isAllRound Boolean som representerar om deltävlingen är allround.
* judgesArray Array med domare i objektform som ska döma i deltävlingen.

**Metoder:**

Metoderna ligger som en prototype, för att spara resurser ifall det någon gång i framtiden skapas många Event objekt.

* addJudge(judgeObj) Lägger till ett Domar objekt i judgesArray:en.
* notifyJudges() Informerar domarna i judgesArray:en via e-post eller sms om att de ska döma i deltävlingen.
* sendEmail(emailAdress) Skickar e-post till e-postadress (ej implementerad)
* sendSms(phoneNumber) Skickar sms till telefonnummer (ej implementerad)

****

# Uppgift 4 - Enhetstestning

Testsvit för skapande av tävlingar

**Beskrivning av testfixturer**

* Testfixturer för Competition ”klassen”.
  + Json-fil formaterad fixtur-fil med korrekta värden för skapande och manipulation av ett Competition objekt .
  + Json-fil formaterad fixtur-fil med felaktiga värden för skapande och manipulation av ett Competition objekt.
* Testfixturer för Event ”klassen”.
  + Json-fil formaterad fixtur-fil med korrekta värden för skapande och manipulation av ett Event objekt .
  + Json-fil formaterad fixtur-fil med felaktiga värden för skapande och manipulation av ett Event objekt.

## Enhetstest för ”klassen” Competition.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Scenario | Skapa Tävlingstillfälle Kommando | Starttid | Sluttid | Förväntat resultat |
| 1 | Scenario 1 | Ja | Giltig | Giltig | Tävlingstillfället skapas. |
| 2 | Scenario 2 | Ja | Giltig | Ogiltig | Tävlingstillfället skapas inte. |
| 3 | Scenario 3 | Ja | Ogiltig | Ogiltig | Tävlingstillfället skapas inte. |

|  |  |
| --- | --- |
| **ID** | **1** |
| Rubrik | Publicera artikel |
| Förberedelser | Inloggad i systemet |
| Teststeg | 1. En författare skriver en artikel 2. Författaren skickar artikeln till redaktionen |
| Förväntat resultat | Ett meddelande visas som säger att artikeln är publicerad |

## Enhetstest för ”klassen” Event.

# Uppgift 7 – Reflektion

Varför ska testerna skrivas efter att klasserna skrivits, enligt boken ska man skriva testerna före det att klasserna skrivits enligt Test-Driven Development? För att vi ska skriva klasserna med fel ifrån början kanske?

Klasserna får inte exakt samma arbetsflöde som användarfallet. Blir mer detaljerad i klasserna. Exempelvis notifyJudges metoden i event objektet anropas när eventet läggs till i competition objektet.

# Temporärt

## Uppgift 3, borttagna rubriker:

**Modultester**

**Integrationstester**

**Systemtester**

**specialfall: Acceptanstest**

**Resurser**

**Planering**

**Rapportmallar**

Statiska tester? Dynamiska tester?

Vad testar vi?

Testa säkerheten?

Hur testar vi?

Testsviter

Med fokus på olika delar i systemet (säkerhet?)

**– Funktionstestning**

**– Kvalitetskrav**