**Projektdokumentation**

**Syfte**

Anledningen till att konstruera den här tjänsten är att möjliggöra en snabb och smidig webbapplikation där fokus ligger på att: Öppna upp sidan->Göra sökningen->Välj resultat->Få upp relevant information och se trailer för aktuell film. Det finns redan liknande tjänster men många utav dem tillhandahåller flera alternativ vilket tar bort fokus från att snabbt hitta en film att se, ex. www.traileraddict.com. Ett ytterligare syfte är att många sökmotorer låter användaren endast söka på titel. Förutsatt att användaren ger ett väldigt generellt sökord i sin sökning kan det vara väldigt svårt att filtrera sina resultat, därför ska tjänsten även leverera sökfilter för Genre, År och Imdb-rating. Kort beskrivet en tjänst för att hitta kvällens film genom sökning som returnerar relevant information och trailer.

**Användarmanual-** Slutrapporten innehåller en kort användarmanual.

Infoga bild över sökmenyn ed beskrivningar ?

**Körinstruktioner**

1. Dubbelklicka på filen ”Webserver.jar” – Servern startas.
2. Högerklicka på filen ”Home.html” och välj ”Öppna med”-> En webbläsare av egen preferens.

* Webbläsaren öppnas och Applikationens startsida visas.

1. Utför någon av funktionerna som finns beskrivna under rubriken ”Användarmanual”

**Teknisk beskrivning**

**Klienten:**

För klientsidan använder vi oss utav Html, CSS, Bootstrap och JavaScript. Vi har byggt vår Html med tre klasser: Home-, Search- och Info-.html. I varje klass har vi valt att låta en header-div med vår sökfunktion alltid ligga överst i fönstret oavsett hur långt ner på sidan användaren är, detta val har vi gjort då applikationens huvudfokus är att snabbt söka nya filmer. I vår CSS fil använder vi oss av diverse BootStrap komponenter då vi ansåg att de var både smidiga och stilrena komponenter som även besparade oss tid. Vid vår server kommunikation använder vi JavaScript för att bland annat hämta sökresultat och lägga upp dem dynamiskt i applikationen. JavaScriptklassen tar emot JSON data och representerar det i HTML.

**Servern:**

Kärnan i vår server är Spark. Då vi som Systemutvecklare är bekväma i Java beslutade vi att köra med Sparks-ramverk som både var bra dokumenterat och lättarbetat. Vår server tar emot olika http anrop som i vårt fall endast blir ”GET” då vi inte har någon funktionalitet som kräver uppdatering eller dylikt. För alla våra server-svar använder vi JSON där vi använder oss av Googles konverter- Gson. När vi kontaktar TrailerAddict Api:t så returnerar det endast XML, därför fick vi även konvertera detta. Konverteringen av både JSON och XML görs till Java-Object som vi hanterar och sedan genererar tillbaka som svar i form av nytt JSON dokument.

Under tiden vi arbetade så fann vi skillnader i OMDB API och TrailerAddicts API referens till filmtitlar. Därför gjorde vi en klass som vi kallar för ”QueryConverter” som rensar ut ord och tecken som inte önskas, dock är inte APIerna konsekventa och därför lyckas inte alltid genereringen av exempelvis trailers. QueryConverter tar även och sorterar ut information från våra URLer. I vår sökfunktion skickar vi med både titel, genre, år och rating där QueryConverter plockar ut denna information och gör den användbar.

Nästa steg var att utöka sökfunktionerna då Omdb endast låter användaren söka på titel och år. Därför skapade vi ”ResultSorter” som sorterar efter genre och rating utöver Omdb APIets titel och år sortering.

Utöver dessa klasser har vi en klass som kommunicerar med film och trailer APIet samt klasser som tar emot Json och Xml datan.

**Exempel Serverstruktur:**

1. Ta emot http-request.
2. Meddela kontroll-klassen och sortera ut URL informationen.
3. Hämta information från API.
4. Konvertera från Json/Xml till Java-klass
5. Sortera resultat.
6. Konvertera resultat till Json.
7. Returnera till klient.