Object-Oriented Programming 4 – Spotify

Versie 1.0.1

Onderdeel van traject: Software  
Vervolg na dit project: Full-stack 1

# Samenvatting casus

Je gaat in tweetallen een CLI (Command Line Interface) varient ontwikkelen van het populaire muziekplatform “Spotify”.

Het project dient gedocumenteerd te worden via GitHub. Maak een scrumboard aan en vul de sprintbacklog met de eisen en wensen. Alle user stories in de backlog dienen te voldoen aan de criteria: Wie, Wat, Waarom en “Realistisch”. Iedere user story dient een duidelijke Definition of Done te bevatten.

Vanuit de eisen en wensen dienen ontwerpen gemaakt te worden. Je verifieert de functionaliteit van het vastgesteld klassendiagram doormiddel van activiteitendiagrammen en use-case diagrammen.

Tijdens het realiseren koppel je regelmatig terug naar de product-owner. Dit doe je door gebruik te maken van sprint reviews en retrospectives.

Na het realiseren van een functionaliteit, dient deze grondig getest te worden. Alle testcases dienen aan te sluiten op de user stories en bevatten alle scenario’s. Alle stappen, het gewenste resultaat en testdata zijn benoemd. Het is belangrijk dat niet alleen het hoofdscenario, maar ook alternatieve scenario’s worden getest.

Na de oplevering wordt een verbetervoorstel gedaan naar de product-owner. De verbetervoorstellen dienen gedaan te worden vanuit het testen, de oplevering en de retrospectives. Het is belangrijk dat je alle opvallende details documenteert en advies uitbrengt hoe het project succesvol kan worden overgedragen.

Tot slot moet de product-owner worden overtuigd dat het werk authentiek is. Toon op zoveel mogelijke manieren aan dat je begrijpt wat je hebt gemaakt.

## Versiebeheer

|  |  |  |
| --- | --- | --- |
| 1.0.0 | Robert Nieuwkoop, Erik Seldenthuis | Opzet leeswijzer – Gebruik van bestaande rubrics, schrijven nieuw casus. |
| 1.0.1. | Robert Nieuwkoop, Erik Seldenthuis | Beoordeling van werkprocessen toegevoegd |

Inhoud

[1. Samenvatting casus 2](#_Toc164840887)

[Versiebeheer 2](#_Toc164840888)

[2. Leerdoelen 3](#_Toc164840889)

[3. Beschrijving van de casus 4](#_Toc164840890)

[4. Beoordeling van de casus 5](#_Toc164840891)

[4.1. Leerdoelen 5](#_Toc164840892)

[4.2. Werkproces 6](#_Toc164840893)

# Leerdoelen

Deze casus biedt de student de kans om de vaardigheden in Object Oriented Programming toe te passen en te ontwikkelen. Het helpt de student bij het verbeteren van de vaardigheden in het ontwikkelen van een interactieve toepassing. Het toepassen van een grafische interface is gewenst, maar is niet verplicht.

* De student oefent het projectmatig werken in iteraties
* De student vertaalt eisen en wensen van de product-owner in concrete user-stories
* De student past schematechnieken toe om functionaliteiten van het te bouwen systeem inzichtelijk te maken
* De student leest en realiseert (onderdelen van) klassendiagrammen om de structuren van het te bouwen systeem in beeld te krijgen
* De student realiseert in de voorgeschreven omgeving vanuit het ontwerp de applicatie
* De student is in staat het ontwikkelproces te bewaken en zorgt voor terugkoppeling richting product-owner
* De student test de applicatie grondig volgens van te voren opgezet testplan en legt de resultaten vast in een testrapport
* De student levert het ontwikkelde systeem op en presenteert dit aan de product-owner

# Beschrijving van de casus

Hendrik Kavel, een entrepreneur, wil graag een concurrent maken voor Spotify. Hij heeft opgemerkt dat Spotify, ondanks veel goede kwaliteiten, toch wat specifieke functionaliteit mist. Hij heeft ervoor gekozen om jullie in te huren om zijn eigen muziekclient na te maken. Hendrik wil uiteindelijk dat zijn muziekclient echt muziek kan afspelen, maar voor nu vind hij het prima als zijn muziekclient alleen via tekst de functionaliteit laat zien.

De applicatie hoeft nog geen verbinding te maken met een database, het dient voor nu alleen als een platform om nieuwe functionaliteit te testen. Alle nummers, albums en gebruikers mogen dus *hard-coded* aangemaakt te worden. Binnen de applicatie dient één gebruiker als **hoofdgebruiker** gekozen te worden. Alle acties die worden ondernomen, worden dan ook vanuit deze hoofdgebruiker uitgevoerd. Dus bijvoorbeeld: als een nieuwe speellijst wordt aangemaakt, dan is deze automatisch van de hoofdgebruiker.

**Client**:

Alle CLI interacties dienen via de Client class afgehandeld te worden. Bijvoorbeeld: Wanneer de gebruiker een nummer wil selecteren, dan dient de bijbehorende methode van Client aangeroepen te worden. Alle nummers, albums.

Alle CLI interactie dient via de Client class afgehandeld te worden. Dus bijvoorbeeld: Wanneer de gebruiker een nummer wil selecteren, dan dient de daarbij horende methode van Client aangeroepen te worden.

**Nummers**:

Nummers hebben een titel, een verzameling aan deelnemende artiesten en een genre. Wanneer een nummer wordt afgespeeld dienen de titel, de deelnemende artiesten en het genre geschreven te worden naar de console.

Wanneer één nummer wordt overgeslagen, dan begint hetzelfde nummer te spelen. Wanneer een nummer wordt gestopt, dan zal de console tonen dat het huidige nummer is gestopt met spelen. Het pauzeren van een nummer hoeft voor het prototype nog niets te doen.

**Speellijsten**:

Een gebruiker kan meerdere speellijsten hebben en moet nummers, albums en andere speellijsten kunnen toevoegen aan een speellijst. De gebruiker moet alle nummers van een specifieke speellijst kunnen inzien. Ook moet de gebruiker een nummer uit de speellijst kunnen verwijderen. Tot slot moet de gebruiker een speellijst moeten kunnen afspelen. Bij het afspelen van een speellijst, begint het eerste nummer in de lijst te spelen. Wanneer het laatste nummer van een speellijst is gespeeld, dan stopt het spelen automatisch. Wanneer een nummer uit een speellijst wordt overgeslagen, dan moet het volgende nummer beginnen met spelen.

**Gebruikers**:

Een hoofdgebruiker moet kunnen inzien welke andere gebruikers deel zijn van de nieuwe spotify client. -Van deze andere gebruikers moet de hoofgebruiker iemand kunnen toevoegen als vriend. Ook moet de hoofdgebruiker een vriend kunnen verwijderen. De hoofdgebruiker moet alle vrienden kunnen inzien. De hoofdgebruiker moet speellijsten van vrienden kunnen inzien. Dit betekend dat er een lijst van speellijsten naar voren moet komen. De hoofdgebruiker moet vervolgens alle nummers van een specifieke speellijst kunnen inzien. Tot slot, de hoofdgebruiker moet zo’n geselecteerde speellijst kunnen overnemen of toevoegen aan een eigen speellijst.

**Albums**:

Albums hebben een titel, een verzameling van artiesten en een verzameling van nummers. Aan albums kunnen geen nieuwe nummers worden toegevoegd. Een hoofdgebruiker moet alle nummers of albums kunnen inzien. Dit betekend dat er een lijst van alle nummers of albums naar voren moet komen. Uit deze nummers of albums moet de gebruiker nummers kunnen selecteren. De gebruiker kan op deze wijze een nummer of album toevoegen aan een eigen speellijst, of kunnen afspelen. Het afspelen van een album gebeurt op dezelfde manier als een speellijst.

**Artiesten**:

Artiesten hebben een naam en verzamelingen van albums en nummers waar zij aan hebben gewerkt. De hoofdgebruiker moet een alle artiesten kunnen inzien. Dit betekend dat alle albums en nummers (singles) waar een artiest aan heeft gewerkt naar voren moet komen.

# Beoordeling van de casus

Op de volgende pagina’s staan de rubrics waar je aan dient te voldoen, je kunt met gebruik van deze rubrics jezelf een indicatie geven van hoe je ervoor staat en waar je nog aan moet werken tijdens de projecturen voor dit project.z

## Leerdoelen

|  |  |  |
| --- | --- | --- |
| Wat kan beter? | Wat zijn de eisen? | Wat ging goed? |
|  | (W1) De eisen en wensen zijn verwerkt in een product backlog |  |
|  | (W1) De user stories in de backlog voldoen aan de criteria (wie, wat, waarom en realistisch) en zijn voorzien van een Definition of Done |  |
|  | (W2) Er is gebruik gemaakt van schematechnieken. De diagrammen zijn conform de UML standaard |  |
|  | (W3) Er is inhoud van de user stories gerealiseerd binnen de gestelde/geplande tijd |  |
|  | (W3) De opgeleverde functionaliteiten voldoen aan de eisen en wensen zoals omschreven in de betreffende user story. |  |
|  | (W3) De kwaliteit van de code is voldoende. Dit uit zich onder andere in: OOP, objectstructuur, MVC, validatie, efficiëntie, foutafhandeling en terugkoppeling, security (veilig programmeren). |  |
|  | (W3) De code is verzorgd, leesbaar, en gestructureerd |  |
|  | (W1) De voortgang is bewaakt en de juiste keuzes/afwegingen zijn gemaakt op basis van prioriteiten |  |
|  | (W5) Er zijn verbetervoorstellen gedaan |  |
|  | **De beoordelaar is overtuigd van de authenticiteit van het gemaakte werk van de kandidaat.** |  |

## Werkproces

|  |  |  |
| --- | --- | --- |
| B1-K2-W1 Overlegt | | |
| De student is actief aanwezig in overlegmomenten; brengt zelf onderwerpen in en stelt vragen. | | |
| Niet of nauwelijks | **Enigszins** | **Volledig** |
| De student is niet aanwezig. | De student is aanwezig, brengt soms zelf onderwerpen in en stelt soms vragen. | De student neemt actief deel aan overleggen. |
| De student kan de eigen voortgang helder communiceren en daarin problemen en knelpunten benoemen. | | |
| Niet of nauwelijks | **Enigszins** | **Volledig** |
| De student communiceert niet. | De student communiceert deels over de eigen voortgang en benoemt deels de problemen en knelpunten. | De student communiceert helder over de voortgang en kan de problemen en knelpunten volledig benoemen. |
| De student legt de gemaakte afspraken vast. | | |
| Niet of nauwelijks | **Enigszins** | **Volledig** |
| De student legt niks vast. | De student legt de gemaakte afspraken deels vast. | De student kan de functionaliteit uitleggen en de gemaakte keuzes voldoende onderbouwen. |