Achievement Hunter

Achievement Hunter is een library om het mogelijk te maken om achievements/uitdagingen in je applicatie te verwerken. Deze uitdagingen kunnen vooraf geconfigureerd worden, of tijdens run-time worden toegevoegd.

De achievements worden ingeladen via en zijn benaderbaar met de achievement manager klasse. Deze klasse is de klasse die alle functionaliteit uitvoert of in het geval van een individuele achievement een object beschikbaar stelt die de benodigde functionaliteit kan uitvoeren. Dit staat hieronder bij het kopje Logica fatsoenlijk uitgelegd.

Alle klassen in deze library zijn losgekoppeld en kunnen hierbij worden uitgewisseld voor een ander gelijkwaardig component indien gewenst. Dit wordt allemaal geregeld via de configuratie utility klasse zoals hieronder uitgelegd.

# Configuratie

## Data Acces Configuratie Bestanden

Er zijn twee configuratie bestanden die dienen te worden ingevuld voor persoonlijk gebruik.

### Data-acces.config.json

Hieronder is een voorbeeld van de opzet van het json bestand. Alle velden hierin zijn vereist voor de werking van het systeem.

|  |
| --- |
| {  "connection": "INSERT CONNECTION STRING",  "database": {  "linkTable": "Link Table Name",  "achievementTable": "Achievement Table Name",  "userTable": "User Table Name",  "userKey": "User Primary Key"  }  } |

***Connection***: Hierin komt de connection string te staan.

***LinkTable***: Hierin komt te staan hoe de link tabel heet of moet gaan heten.

***AchievementTable***: Hierin komt te staan hoe de achievement tabel heet of moet gaan heten.

***UserTable***: Hierin komt te staan hoe de gebruikerstabel heet. Deze moet dus al vooraf in de database staan.

***UserKey***: Hierin komt te staan wat de primary key is van de gebruikerstabel. Deze is nodig om de individuele gebruiker aan zijn of haar achievement voortgang te koppelen.

### Achievement.config.json

Hieronder is een voorbeeld opgezet van het json bestand. Hier staan twee achievements in een json array. Deze array mag ook leeg gelaten worden indien de achievements in de database tijdens run-time worden geconfigureerd.

|  |
| --- |
| [  {  "id": "ACH1",  "title": "I am an achievement!",  "description": "And you deserve to have achieved me!",  "score": 500,  "goal": 3  },  {  "id": "ACH2",  "title": "I am another achievement!",  "description": "And you trigger me!",  "score": 100,  "goal": -1  }  ] |

***Id***: Hierin komt het te geven id van de achievement te staan.

***Title***: Hierin komt de te geven naam van de achievement te staan.

***Description***: Hierin komt de te geven beschrijving of flavour tekst van de achievement te staan.

***Score***: Hierin komt waarde van het behalen van de achievement te staan.

***Goal***: Hierin komt het doel van de achievement te staan. Deze kan ook 0 of minder zijn. Hierdoor zal de achievement anders werken en kan deze alleen worden behaald door deze te triggeren. (Voor uitleg zie je de het kopje Functies onder User Achievement van de Logica.

## Configuratie Klasse

De configuratie klasse kan worden gebruikt om de library te configureren vóór gebruik binnen een applicatie middels de Autofac dependency injection container oplossing. Hiervoor zijn twee functies beschikbaar.

### Algemeen

De klasse is hoofdzakelijk voor het aanmaken van een DI-container met Autofac, maar wordt tevens ook gebruikt als een oplossing om automatisch de library te configureren indien de container wordt afgebouwd binnen deze klasse.

***Opmerking***: Indien er alleen een builder wordt geconfigureerd en de data laag is nog niet geconfigureerd zal dit handmatig moeten worden verricht! Denk hieraan!

### Functies

***ConfigureBuilder***: Maak een nieuwe builder aan met de juiste depencies of voeg nieuwe dependency configuraties toe aan een al bestaande builder.

***ConfigureContainer***: Maak een container aan met de juiste dependency configuratie. Deze kan ook worden voortgebouwd op een bestaande builder. Wanneer deze container is aangemaakt wordt meteen de rest van de library geprepareerd.

# Logica

## Achievement Manager

De achievement manager heeft 6 verschillende functies beschikbaar en is de hoofdmotor achter de library. De achievements zelf zijn bereikbaar via een indexer binnen de achievement manager.

### Algemeen

De achievement manager is verantwoordelijk voor algemene achievements. Deze worden gemanipuleerd middels de functies hieronder. Teven stelt de achievement manager middels indexers de voortgang van een gebruiker beschikbaar.

***achievementManager***[string *userId*]: Dit is een lijst van voortgangen van de gebruiker met het gespecificeerde userId. Deze kunnen alleen worden bekeken, en niet direct worden gemanipuleerd.

***achievementManager***[string *userId*, string *achievementId*]: Dit stelt de individuele voortgang van een achievement bloot en kan gebruikt worden om een achievement te updaten. Indien je de voortgang van een gebruiker wilt updaten moet je dat via dit object doen. Voor een beschrijving hoe dit object werkt zie het kopje User Achievement hieronder.

### Functies

***AddAchievement***: Voegt een nieuwe achievement toe aan het systeem. Deze wordt niet opgeslagen in de configuratie file, tenzij gewenst. Ingeladen gebruikers in de achievement manager krijgen de achievement (leeg/onvoltooid) toegevoegd aan hun achievement lijst.

***ResetAchievements***: Laad de achievements van de configuratie file opnieuw in. (Met de in-memory database zullen de voortgangen worden gereset)

***DeleteAchievement***: Verwijder een achievement uit het systeem. Deze wordt niet verwijderd uit de configuratie file, tenzij gewenst. Van ingeladen gebruikers in de achievement manager zal deze achievement uit hun achievement lijst verwijderd worden.

***LoadUserProgress***: Laadt de achievement voortgang van een gebruiker in. Deze voortgang is beschikbaar vanuit de indexer van de achievement manager.

***ClearUserData***: Laadt de voortgang van een gebruiker uit bij de achievement manager. Deze is nog steeds beschikbaar in de database en kan zodoende opnieuw worden ingeladen met *LoadUserProgress*.

***DeleteUserData***: Verwijder de voortgang van een gebruiker permanent uit het systeem. Deze is zal niet meer terug te krijgen zijn wanneer de gebruiker opnieuw wordt ingeladen in de achievement manager.

## User Achievement

Het user achievement object heeft drie functies beschikbaar om een achievement te updaten.

### Algemeen

Het object heeft een paar tal properties om voortgang bij te houden.

***UserId***: Hierin staat de gebruikers id van de gebruiker wiens voortgang dit object betreft.

***Id***: Hierin staat het id van de achievement waarover deze voortgang gaat.

***Title***: Hierin staat de titel van de achievement.

***Description***: Hierin staat de beschrijving van de achievement. Deze kan ook worden gebruikt als flavour tekst voor een leuk pop-up wanneer je een achievement hebt behaald.

***Score***: Hierin staat hoeveel experience of punten het behalen van deze achievement de gebruiker opleverd.

***Goal***: Hierin is het doel van de counter property hieronder beschreven. Indien deze negatief wordt gezet zal een achievement enkel een trigger nodig hebben om behaald te worden.

***Counter***: Hierin wordt staat de werkelijke voortgang van de gebruiker. Indien deze gelijk is aan de goal zal deze niet meer kunnen worden verhoogd. En is de achievement behaald. (Tenzij de goal minder dan 0 is. Dit zal hieronder verder worden uitgelegd)

***IsCompleted***: Hierin staat of de gebruiker de achievement al eens heeft behaald of niet.

***AchievementType***: Hierin staat welk type achievement de voortgang om gaat.

### Functies

***Trigger***: Indien de goal van een achievement 0 of lager is, wordt deze beschouwd als een triggerbare achievement. Als deze functie wordt gebruikt heeft de gebruiker deze soort achievement dan behaald. Als de gebruiker deze achievement nog niet eerder had behaald wordt true teruggegeven om te indiceren dat de achievement voor het eerst is behaald.

***SetCounter***: Indien de goal van een achievement meer dan 0 is, wordt deze beschouwd als een score achievement. Als deze functie wordt gebruikt wordt de waarde van counter verandert naar de ingevoerde waarde. Indien de waarde meer dan of gelijk aan de goal is wordt deze op 0 gezet en wordt teruggegeven of de gebruiker de achievement voor de eerste keer is behaald middels true terug te geven. Indien de waarde lager is dan 0, wordt de waarde op 0 gezet.

***IncreaseCounter***: Indien de goal van een achievement meer dan 0 is, wordt deze beschouwd als een score achievement. Als deze functie wordt gebruikt wordt de waarde van counter opgeteld met de ingevoerde waarde (of 1 indien geen waarde is ingevoerd). Indien de waarde meer dan of gelijk aan de goal is wordt deze op 0 gezet en wordt teruggegeven of de gebruiker de achievement voor de eerste keer is behaald middels true terug te geven. Indien de waarde lager is dan 0, wordt de waarde op 0 gezet.

# Data Access

## In-Memory

De in-memory data access is gemaakt voor systemen zonder een gekoppelde database. Deze is niet ontworpen om te worden gebruikt bij uiteindelijke release van een systeem aangezien een overvloed aan data een stackoverflow error zou kunnen veroorzaken. Daarnaast zal wanneer het programma is afgesloten de data worden gereset. Deze data access is dus voornamelijk ontworpen voor pre-release demonstratie van een systeem.

## MySQL

De MySQL data access moet de implementatie voor het gebruik van een MySQL database ondersteunen. Aangezien er geen MySQL database beschikbaar was is deze niet volledig geïmplementeerd. Wel doende is het inladen van de configuratie data voor de database al verwerkt in de data access laag van de MySQL. De rest van implementatie kan doende worden ingevuld wanneer deze daadwerkelijk implementeerbaar is middels een beschikbare MySQL database.