

Click-IT

Beschrijving

Enzo Evers

2016

Contents

[Click-IT 3](#_Toc452381638)

[Beschrijving 3](#_Toc452381639)

[Samenvatting 3](#_Toc452381640)

[Profiel 3](#_Toc452381641)

[Profiel updaten 3](#_Toc452381642)

[Mail 3](#_Toc452381643)

[Score 4](#_Toc452381644)

[Screenshots 5](#_Toc452381645)

[Database 8](#_Toc452381646)

[Ontwerp 8](#_Toc452381647)

[Beschrijving 8](#_Toc452381648)

[Player, Country en Gender 8](#_Toc452381649)

[LoginData 8](#_Toc452381650)

[GameData 8](#_Toc452381651)

[Gebruik 8](#_Toc452381652)

[Query’s 9](#_Toc452381653)

[Query (SQL functie (COUNT, AVG), INNER JOIN, GROUP BY, HAVING, ORDER BY) 9](#_Toc452381654)

[Resultaat: 9](#_Toc452381655)

[Uitleg: 9](#_Toc452381656)

[Query (SQL functie (MAX, MIN, AVG), 5x sub-query, INNER JOIN, GROUP BY, ORDER BY) 10](#_Toc452381657)

[Resultaat 11](#_Toc452381658)

[Uitleg 11](#_Toc452381659)

[Query (SQL functie (MIN), INNER JOIN, GROUP BY, ORDER BY) 12](#_Toc452381660)

[Resultaat 12](#_Toc452381661)

[Uitleg 12](#_Toc452381662)

[Querie (SQL funtie (COUNT)) 13](#_Toc452381663)

[Resultaat 13](#_Toc452381664)

[Uitleg 13](#_Toc452381665)

[Querie (SQL funtie (COUNT), INNER JOIN, sub-query, GROUP BY, HAVING) 14](#_Toc452381666)

[Resultaat 14](#_Toc452381667)

[Uitleg 14](#_Toc452381668)

# Click-IT

## Beschrijving

### Samenvatting

Click-IT is een reactie spel. Je maakt een profiel aan en logt vervolgens in. De login en logout worden en een aparte tabel opgeslagen door gebruik te maken van de DateTime klasse. Je kiest hoeveel rondes (aantal keer te klikken per spel) je wilt en klikt op de start knop om het spel te beginnen. Er word door middel van de Random klasse een random button ‘geactiveerd’. Er word eigenlijk een index gekozen voor de List van type Button En de List van type string. Wanneer op een button word geklikt wordt de naam vergeleken met de gekozen index in de lists met alle button van het type Button en het type string. Hieruit blijkt of er een tijd genoteerd moet worden of niet.

Tijdens het spelen van Click-IT wordt er achtergrond muziek afgespeeld. Dit gebeurd door gebruik te maken van de SoundPlayer klasse.

### Profiel

Wanneer men een nieuw profiel aanmaakt wordt er gekeken of het wachtwoord en username al bestaat. Als een van de twee bestaat wordt er een error getoond. Wanneer er een emailadres is ingevoerd en er is op de “Make profile” knop gedrukt wordt er geprobeert een ‘confirmation’ mail te sturen door middel van een “try-catch”. Wanneer het sturen van de email niet is gelukt doordat er een FormatException optreed word er een error getoond met de info dat het email adress niet valide is. In het geval van een ander soort error word er een algemene error getoond. Wanneer een een veld leeg in gelaten wordt er een error getoond die zegd dat alles ingevuld moet zijn. Deze velden krijgen en rode border.

### Profiel updaten

Er is ook de mogelijkheid het profiel te ‘updaten’ wanneer deze optie word gekozen kom je in hetzelfde scherm als wanneer je een nieuw profiel gaat aanmaken, maar zijn alle velden met jouw persoonlijke informatie gevuld. In de code gebeurd er echter iets heel anders. Waar er een nieuwe rij in de database word aangemaakt als je een nieuwe profiel aanmaakt, wordt nu de rij van de ingelogde persoon verandert.

### Mail

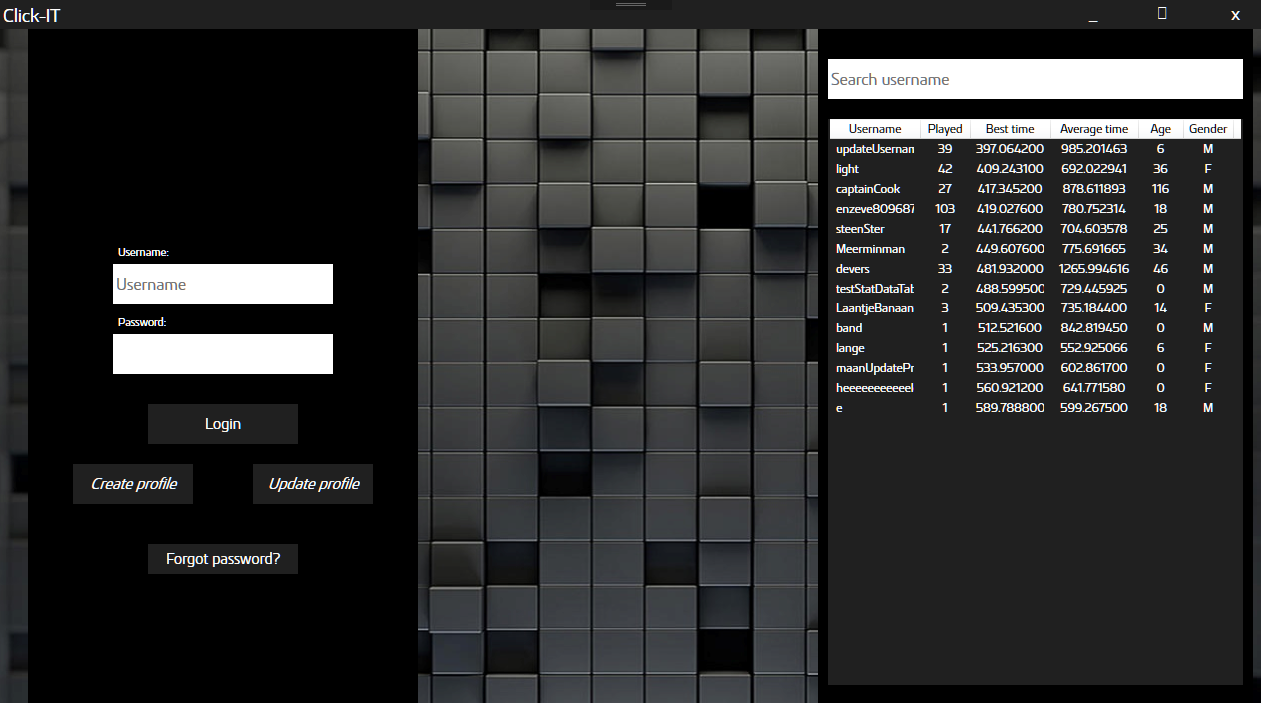
Wanneer iemand zijn wachtwoord vergeten is kan hij zijn username opgeven. Vervolgens wordt er met een query het opgegeven emailadres en wachtwoord opgevraagd uit de database. Het wachtwoord wordt naar het emailadres gestuurd. Voor het kunnen mailen word gebruik gemaakt van de SmtpClient en MailMessage klasse. De SmtpClient klasse zorgt er voor dat je kan zeggen van welke server gebruik gemaakt kan worden, met welk profiel dat gebeurt en het verzenden van de mail. De MailMessage klasse is er om te bepalen welk emailadres de mail stuurt, naar welk emailadres het wordt gestuurd en wat de inhoud van de mail is.

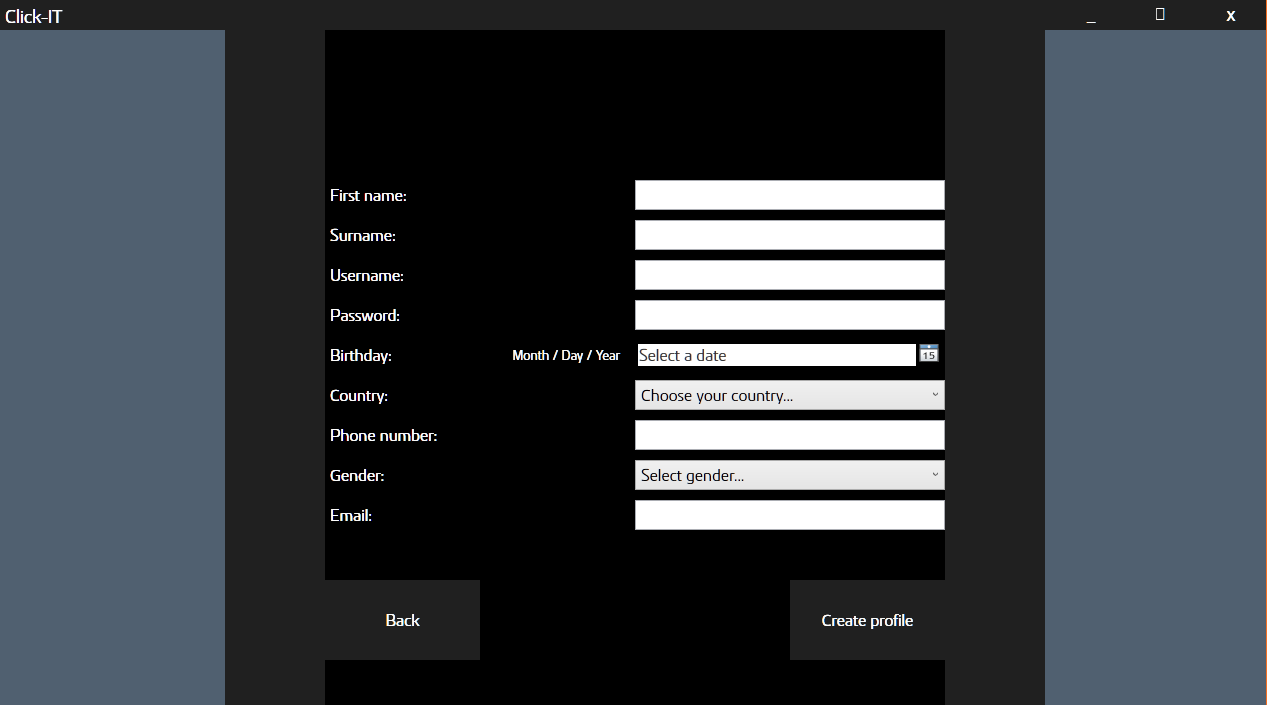
### Score

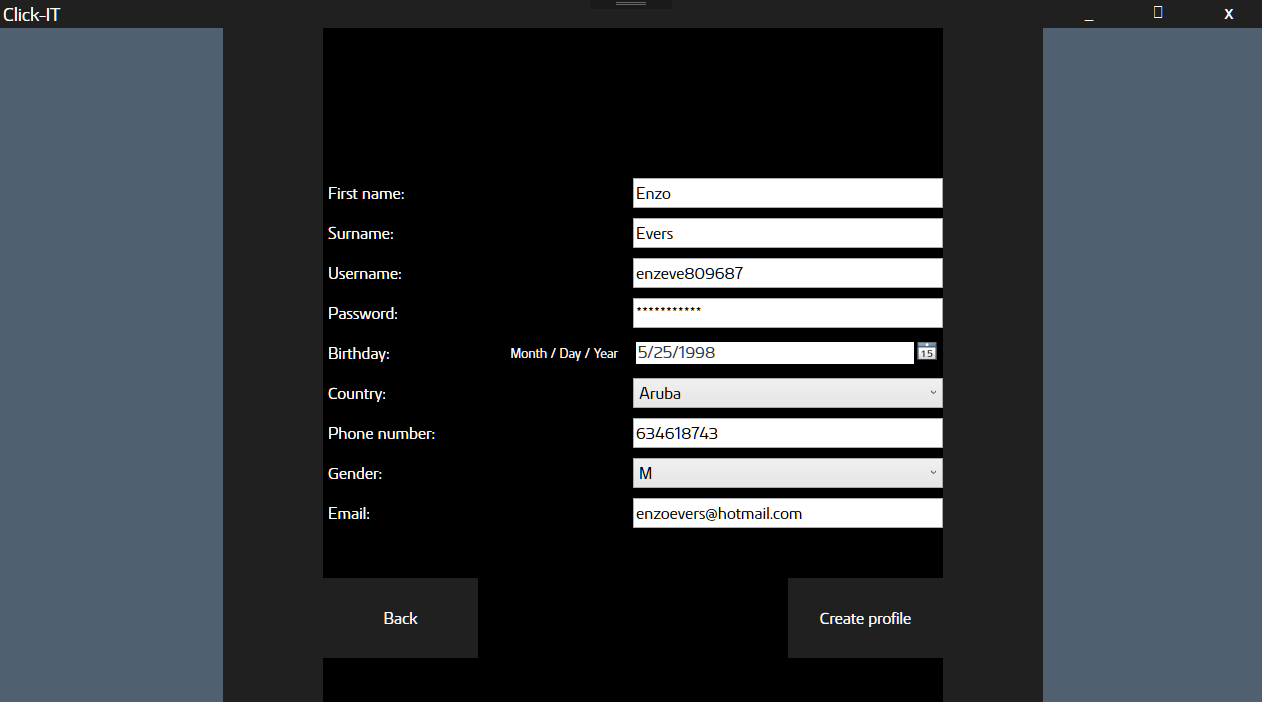
In het beginscherm is een scorebord te zien. Hier de statestieken van de spelers te vinden. Deze maakt gebruik van een ListView die een Binding heeft aan de properties van de PlayerStats klasse. Hierdoor worden de kolommen automatisch met de juiste informatie gevuld. De ListView haalt de informatie uit een eerder gemaakte List van het type PlayerStats. Deze wordt bij het opstarten van de gevuld met alle spelers in de database die minimaal één keer hebben gespeeld. Het is ook mogelijk om te zoeken op gebruikersnaam. Waneer er tekst in de zoekbalk word getypt wordt het scorebord gefilterd. Een soortgelijk scorebord is te vinden in het scherm van het spel zelf. Deze is echter niet te filteren en laat alleen de username en beste tijd zien.

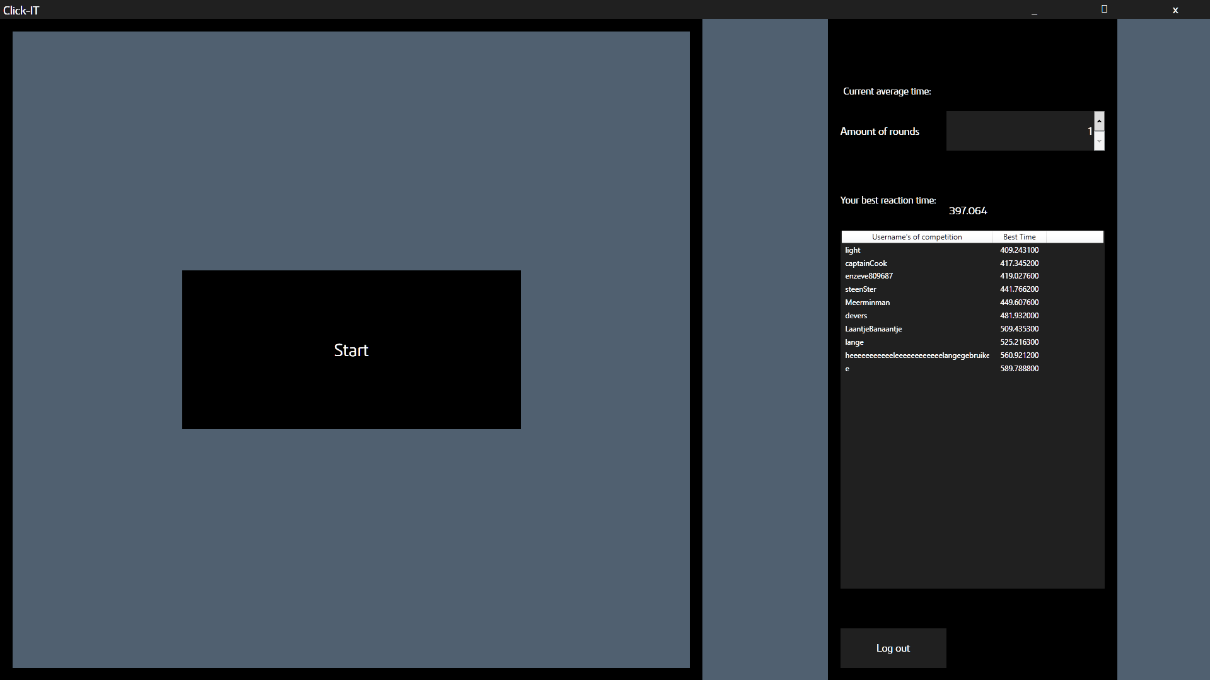
Voor het meten van de tijd worden de Stopwatch en TimeSpan klasse gebruikt. De Stopwatch meet de tijd en wanneer op de juiste button wordt gedrukt tijdens het spel word de tijd opgevraagd door de TimeSpan klasse. Deze wordt vervolgens in de GameData tabel gestop nadat het naar het type decimal is geconvert.

## Screenshots

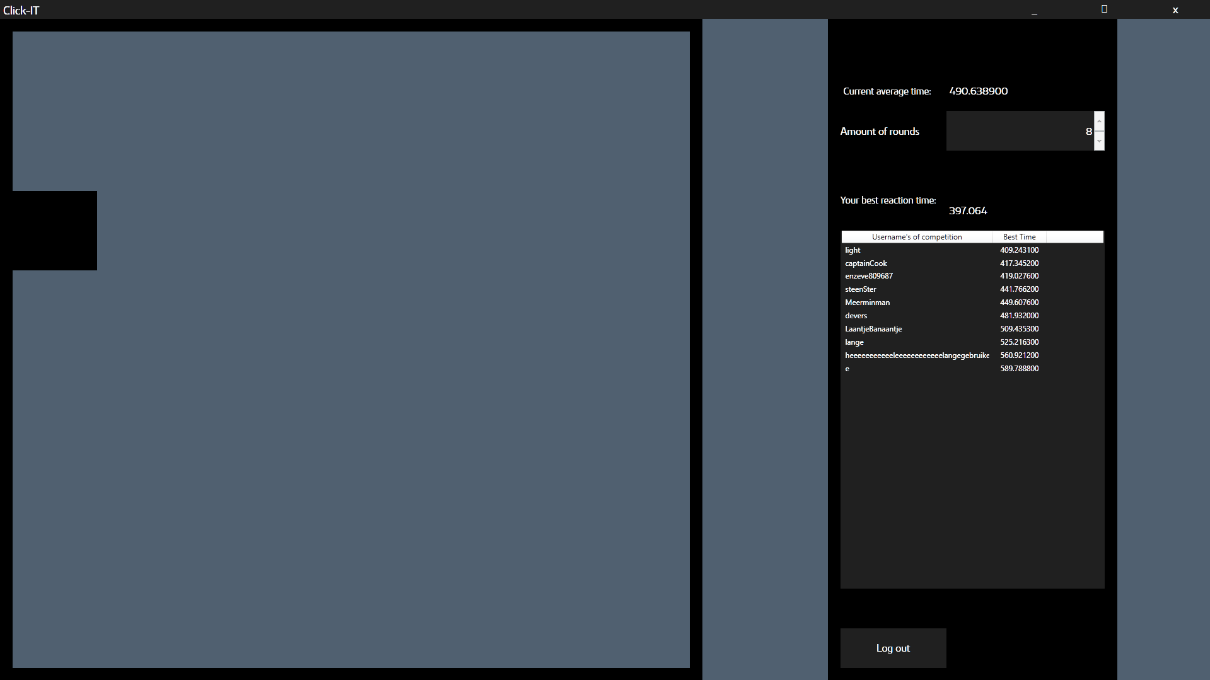
Het inlog scherm met een scorebord.

Het scherm dat je ziet wanneer je een nieuw profiel aanmaakt.

Als een gebruiker zijn profiel wilt updaten ziet hij zijn huidige gegevens al ingevuld.

Wanneer het spel nog niet is begonnen ziet je dit.

Er staat een lijst met de beste tijden van de competitie.

Wanneer op de start button is geklikt zie je een blokje waar je op moet klikken.

# Database

## Ontwerp



## Beschrijving

### Player, Country en Gender

Wanneer er een nieuw profiel wordt aangemaakt komt er een niewe rij in de **Player** tabel. Tijdens het aanmaken van het moet er door middel van een ComboBox een land en geslacht gekozen worden. De index van de ComboBox is gelijk aan de index van in de tabellen van **Gender** en **Country**. Hierdoor kan de index van het gekozen land en geslacht in de **Player** tabel worden gezet.

Wanneer een profiel geupdate wordt gebeurd er in princiepe het zelfde. Het verschil is dat er in plaats van een nieuwe rij wordt aangemaakt, er een bestaande rij (profiel) veranderd word.

### LoginData

In de **LoginData** wordt een nieuwe rij toegevoegd wanneer een profiel uitlogt. Het moment van inloggen is al opgeslagen toen de speler was ingelogt, maar word pas in de tabel geschreven wanneer er ook wordt uitgelogt.

### GameData

Wanneer er een spel bezig is worden alle reactie tijden in een List opgeslagen. Wanneer het spel is afgelopen worden alle tijden uit de List, samen met het eerder opgevraagde huidige datum, GameID en UserID, in de tabel gezet.

### Gebruik

Om nuttige informate uit de database te halen maak ik gebruik van een grote query (p.9). Hierdoor bespaar ik opslag omdat ik geen extra tabel hoef the maken die alle deze informatie opslaat.

# Query’s

## Query (SQL functie (COUNT, AVG), INNER JOIN, GROUP BY, HAVING, ORDER BY)

SELECT

p.Username,

COUNT(Reaction\_time) AS Times\_played,

AVG(Reaction\_time) AS Average\_time

FROM

GameData gb

INNER JOIN

PLayers p ON gb.UserID = p.PlayerID

GROUP BY

Username

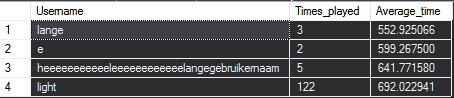
HAVING

AVG(Reaction\_time) <= 700

ORDER BY

AVG(Reaction\_time) ASC

### Resultaat:

(1)

### Uitleg:

De querie geeft een tabel terug die de kolommen Username, Times\_played en Average\_time heeft en is geordend op basis van de Average\_time. Deze loopt op. Er worden echter alleen de spelers getoond die een gemiddelde tijd lager dan 700 miliseconden hebben.

De kolommen Times\_played en Average\_time krijgen hun naam door het AS statement.

## Query (SQL functie (MAX, MIN, AVG), 5x sub-query, INNER JOIN, GROUP BY, ORDER BY)

SELECT

p.Username,

p.birthday,

g.Gender,

(SELECT

MAX(gd.GameID)

FROM

GameData gd

WHERE

p.PlayerID = gd.UserID

GROUP BY

gd.UserID

HAVING

MAX(gd.GameID) > 0) AS Times\_played,

(SELECT

MIN(gd.Reaction\_time)

FROM

GameData gd

WHERE

p.PlayerID = gd.UserID

HAVING

MIN(gd.Reaction\_time) > 0) AS Best,

(SELECT

AVG(gd.Reaction\_time)

FROM

GameData gd

WHERE

p.PlayerID = gd.UserID

HAVING

AVG(gd.Reaction\_time) > 0) AS Average

FROM

Player p

INNER JOIN

Gender g ON g.Id = p.Gender\_id

WHERE

(SELECT

MAX(gd.GameID)

FROM

GameData gd

WHERE

p.PlayerID = gd.UserID) > 0

ORDER BY

(SELECT

MIN(gd.Reaction\_time)

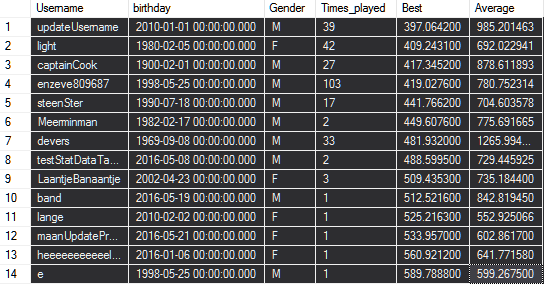
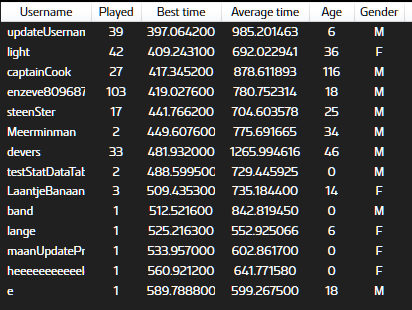
FROM

GameData gd

WHERE

p.PlayerID = gd.UserID) ASC

### Resultaat

(2.1) (2.2)

### Uitleg

Op 2.1 is te zien wat er werkelijk door de query wordt opgevraagd. Dit is alle data die in op het scorebord komt te spalen is het hoofdscherm. De sub-queries zijn er voor om uit een hele grote lijst met alle losse reactie tijden nuttige informatie op te halen.

De INNER JOIN heeft als functie de “M” of “F” uit de Gender tabel te halen.

De opgevraagde informatie word in de klasse “PlayerStats” gezet. Deze klasse maakt van de data in de kolom “birthday” een leeftijd. Het uiteindelijke resultaat is de zien op 2.2.

## Query (SQL functie (MIN), INNER JOIN, GROUP BY, ORDER BY)

SELECT

p.Username,

min(gd.Reaction\_time) AS Best\_time

FROM

GameData gd

INNER JOIN

Players p ON gd.UserID = p.PlayerID

GROUP BY

p.Username

ORDER BY

min(gd.Reaction\_time) ASC

### Resultaat

(3)

### Uitleg

De querie vervangt de kolom UserID uit de tabel GameData met de Username kolom uit de tabel Players. Dit gebeurd op basis van koppeling met de foreign keys die in de UserID kolom staan. Deze worden gekoppeld aan de juiste rij in de Players tabel en vervangen door wat er in de Username kolom staat. Als laatst wordt er gesorteerd zodat de beste tijd boven aan komt te staan.

De kolom Best\_time krijgt zijn naam door het AS statement.

## Querie (SQL funtie (COUNT))

SELECT

COUNT(\*) AS Usernames\_like\_given\_username

FROM

Players

WHERE

Username LIKE @Username

### Resultaat

Met “@Username” = 'enzeve809687'

(5)

### Uitleg

De querie zoekt in de tabel Players ik de kolom Username naar een username hetzelfde als de ingevoerde username bij @Username.

De kolom krijgt zijn naam door het AS statement.

## Querie (SQL funtie (COUNT), INNER JOIN, sub-query, GROUP BY, HAVING)

SELECT

p.Username,

COUNT(ld.LoginID) AS Times\_logged\_in

FROM

LoginData ld

INNER JOIN

Players p ON ld.UserID = p.PlayerID

WHERE Login\_date IN (SELECT Login\_date

FROM LoginData

WHERE Login\_date <= @Date)

GROUP BY

p.Username

HAVING

COUNT(ld.LoginID) >= 10

### Resultaat

Met “@Date” = '2016-05-15T00:00:00.000'

(6)

### Uitleg

De query laat zien wie hoevaak heeft ingelogd voor of gelijk aan een bepaalde datum. Dit wordt gedaan door gebruik van een sub-query. Daarnaast moet er minimaal 10 keer zijn ingelogd om in de tabel te komen staan. Gebeurd doormiddel van de HAVING statement. De username word door middel van de INNER JOIN opgehaald. De uiteindelijke tabel bevat een kolom Username en Times\_logged\_in. De laatste krijgt zijn naam door het AS statement.