Droeftoeters-ICT – Software ontwerp documentatie

Inlever datum: 04/04/2025

# 1 – Informatie

## 1.1 - Studenten

Christian Scott Horler - 2227807

Lilah Sanchez - 2233692

Martijn van Houwelingen - 2225486

Sophie Iersel - 2232301

Thomas Koman – 2226974

## 2.2 – Project

Groepsnaam: Droeftoeters

Projectnaam: Droeftoeters-ICT

# 2 - Inhoud

Contents

[1 – Informatie 1](#_Toc194699978)

[1.1 - Studenten 1](#_Toc194699979)

[2.2 – Project 1](#_Toc194699980)

[2 - Inhoud 1](#_Toc194699981)

[3 – Inleiding 2](#_Toc194699982)

[4 – Diagrammen 3](#_Toc194699983)

[4.1 – Use Case 3](#_Toc194699984)

[4.2 – API 3](#_Toc194699985)

[4.3 Unity 4](#_Toc194699986)

# 3 – Inleiding

Bij het maken van software heb je goede architectuur nodig. Dit vormt als een soort van fundament om op te bouwen, een soort blauwdruk, maar ook als een archief of samenvatting om naar terug te kijken om het te begrijpen en te kunnen uitbereiden. Dit document laat vooral met diagrammen een overzicht zien.

# 4 – Diagrammen

## 4.1 – Use Case

Dit diagram laat vooral zien wat voor users wat moeten gebruiken. Een diagram zoals deze helpt bij het bedenken van de implementatie aan wat je allemaal moet denken, bijvoorbeeld: kan iedereen dit gebruiken? Of maar 1 type persoon.

A diagram of a person's diagram

AI-generated content may be incorrect.

Een diagram zoals dit is een mega grote versimpeling van het product, hierna ga je ook pas echt meer nadenken in de diepgang van het product.

## 4.2 – Backend

### 4.2.1 - API

We hebben gekozen om de applicatie in 2 lagen te stoppen, 1 api en een unity project. Hiervan is in een ander document meer uitleg gegeven over de backend, en hoe het allemaal zit. Maar hier ga ik wel de Klasse Diagram tonen:

[insert]

Een klasse diagram laat echt zien hoe alles werkt, qua klasses en interacties er tussen. Als ik naar iemand zou lopen met mijn klasse diagram zou in theory iemand de zelfde werkende api kunnen verzinnen als dat ik al heb gemaakt. Aan het begin was deze diagram veel simpeler, en als het ware met de code bereidt je ook het klasse diagram uit. Het klasse diagram maakt het simpeler om over toevoegingen te denken, maar je moet ook wel een stukje werkende software realisatie hebben om te beseffen dat bepaalde keuzes uit het klasse diagram aangepast moeten worden.

### 4.2.2 - Database

Verder hebben we ook nog een design gemaakt voor de database, dit heet een ERD.

A computer screen shot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

De eerste diagram is de users die ASP.net standaard toegevoegd heeft. En de 2e foto is de tabellen die we zelf hebben toegevoegd. Achteraf zie ik nu dat we vergeten zijn om de parent ID en ChildID aan te geven als foreign Keys, en die aan Foto 1 te linken. Maar daar hebben we nu geen tijd meer voor :(

## 4.3 Unity

In de frontend hebben we gekozen geen klasse diagram te maken, in plaats hiervan heb ik een flowchart gemaakt over hoe de API connectie werkt bij het inloggen van de unity applicatie. Ik merk aan mezelf dat wanneer ik complexe functionaliteit maak dat ik het beter eerst kan uitdenken, en dat heb ik dus gedaan bij dit stuk. Want wanneer ik hierover na moet denken tijdens het programmeren is het moeilijker. Hij is zo groot dat ik het in 2 plaatjes moet laten zien:

A screenshot of a computer

AI-generated content may be incorrect.

In dit plaatje zie je het automatische login systeem, waarbij we checken voor de sessie token, en het verversen ervan. En links zien we de login van de ouder en kind, en het registreren van een ouder account. Verder wordt het automatisch inloggen flow ook gerunned wanneer 1 query een error terug geeft omdat de user niet is ingelogd. Dan wordt hij automatisch weer ingelogd met deze flow.

A screenshot of a computer

AI-generated content may be incorrect.

In het 2e gedeelte wat je hierboven kunt zien zien we hoe het aanmaken van een kind account werkt. Het kinder account aanmaken is heel scuffed, we doen eerst 1 query om het aan te maken, en daarna doen we er nog 3 om het te koppelen, als 1 van die 3 faalt hebben we een ghost kinder account die niet gekoppeld is aan een ouder maar wel bestaat. We weten dat dit niet zo hoort, en dat we dit nooit in productie zouden doen, maar we hadden geen tijd om onze eigen registratie endpoint te maken. Vandaar dat we hiervoor hebben gekozen.

Als we meer tijd hadden gehad hadden we een api endpoint gemaakt waarbij je met de ingelogde ouder gegevens een call doet met de nieuwe email en wachtwoord, en dat hetzelfde onderwater gebeurt. En het aanmaken van het kind account pas gebeurt als we zeker weten dat de rest ook lukt.

Deze flowchart heeft me geholpen, vooral bij het maken van het registreren van een kind. Dit was zo onoverzichtelijk dat zonder flowchart het veel meer tijd had gekost.