Technisch Ontwerp Codeniacs platform

Timo Strating - medewerker bij UU-games

Februari 14, 2017, Groningen

Table of Content

Chapter 1

Inleiding

Het Codeniacs platform is een interactief platform waar kinderen kunnen leren programmeren of de basis begrippen van het programmeren kunnen vinden. Het Codeniacs platform zal een centraal systeem zijn die meerdere projecten onder het Codeniacs domein versterkt en verbind. Deze opdracht is vanuit een zeer grote opdracht verkleind door de project grenzen te verleggen. De versie die opgeleverd zal gaan worden zal minder gefocust zijn op leraren en ouders en zal zich voornamelijk gaan focussen op de gebruikers / kinderen.

De belangrijkste technische aspecten van de opdracht zijn:

- 2.0 versie proof
  - Er moet van te voren goed zijn nagedacht over de mogelijkheid van 2.0 features.

- Groot project krappe deadline
  - Het project is relatief groot daarom worden er cutting edge technieken gebruikt om zo veel mogelijk te kunnen ontwikkelen in de ontwikkeltijd die gegeven is. Dit betekend wel dat dingen zoals security laag op het vaandel staan. Hier dient daarom de juiste balans in gevonden te worden.

- Platform voor pc, tablet en telefoon
  - Het platform zal uiteindelijk zowel op een pc als een tablet of een telefoon geoptimaliseerd te worden zodat de gebruikers (kinderen)

het in zoveel mogelijk settings zouden kunnen gebruiken.

- Centraal project binnen het Codeniacs domein
  - Het platform zal als een centraal punt fungeren voor de andere projecten. Dit betekend dat de wensen van de andere project goed in kaart gebracht dienen te worden. Zodra alle wensen van alle verbonden partijen volledig in kaart zijn gebracht dienen deze ook verwerkt te worden in het design van het platform.

- AI gids
  - Het platform zal voorzien zijn van een Artificial intelligence systeem. Dit systeem zal zich voor doen als virtuele gids en zal de gebruikers door het platform heen helpen. Deze gids zal juist gebalanceerd moeten gaan worden zodat hij niet te robotachtig lijkt maar ook niet te veel tijd inneemt in het ontwikkel proces.

Chapter 2

Klassendiagram

2.1 Inleiding

Het project zal gebouwd worden in een programmeer taal die niet OOP first is ontworpen daarom zal er geen klassen diagram gemaakt worden vooraf de ontwikkeling.

2.2 Components

Wel zal de website worden ontwikkeld door het gebruik van components. Components zijn kleine onderdelen van de website. Hierbij moet je denken aan grote onderdelen zoals de header of de footer maar ook aan weer kleinere onderdelen zoals de profile knop in de header. Dit zijn allemaal components.

De programmeer style is voor deze opdracht specifiek afgestemd. De volgende onderdelen zijn in overweging gebracht.

- Rapid development / toekomst uitbreidbaarheid
- Rapid prototyping / toekomst bestendigheid
- Beginnen met een boilerplate / of vanaf scratch
- NoSQL / SQL
- True isomorphic code / language specifiek domein talen voor de Database, Server en Client

Uit deze overwegingen is de keuze gemaakt om voornamelijk de linker kan van de bovenstaande tabel aan te houden in verband met de omvang van het project dat in verhouding staat tot de deadline's.

Daarom is er gekozen om MeteorJS te gebruiken in een zeer specifieke manier. Dit wil zeggen dat het systeem zo gebouwd zal gaan worden dat code in cellen geplaatst gaan worden. Dit wordt binnen MeteorJS een component genoemd. Wordt een cel te groot of heeft het te veel verantwoordelijkheden dan zal deze worden opgesplitst in 2 of meerdere cellen. Op deze manier kan de code gemakkelijk dynamisch groter groeien.

Dit heeft wel als nadeel dat er op dit moment geen geranties zijn voor de uiteindelijke klassen structuur van het Platform. Het is daarom niet mogelijk om op dit tijdstip een overzicht te gaan geven voor de Klassen of Components die uiteindelijk gebruikt gaan worden.

Chapter 3

Gegevensverzameling

3.1 Database

Alle gegevens zullen worden opgeslagen in een MongoDB database, Deze database zal op de server gaan draaien. Zodra een client de website download zal hij een kleine versie van de database downloaden en uiteindelijk gaan draaien. Dit systeem wordt een miniMongo genoemd.

De MongoDB database die als het ware uiteindelijk bij de client draait zal voor de client net doen alsof hij de enigste database is. De client kan ook alleen maar strikt met zijn eigen database communiceren. daarna gaat deze zogenoemde miniMongo database de volgende eigenschappen vertonen:

- De client voert een Mongo commando uit op de database maar de database veranderd niet. Voorbeeld commando "get" oftewel haalop.
  - Minimongo voert het commando uit en doet verder niks.
  - Kan hij de data niet vinden dan controleert hij ook ff bij de server of hij de data daar wel zou mogen vergaren.

- De client voert een Mongo commando uit op de database die een aanpassing maakt op de data. Voorbeelden put, update, remove oftewel voeg iets toe, update iets of verwijder iets.
  - Minimongo voert het commando uit en comminuceerd daarna terug naar de MongoDB database op de server wat veranderd is.

- De Server krijgt van een minimongo een aanpassing door voor zijn MongoDB database. Dit heeft meerder stappen en gebeurt enkel en alleen in volgende volgorde:

1. MongoDB kijkt of de het een geldig mongo commando is.
2. MongoDB voerd het commando uit op zijn database als het commando geldig is.
3. De MonoDb database kijkt daarna of er miniMongos zijn die hebben aangegeven dat zij de meest actuele versie willen hebben. Dit wordt gedaan door een zogenaamde subscriptie oftewel abonnement op de data.
4. Iedereen die een zogenaamt abonnement heeft op de data zal daarna een berichtje van de server ontvangen met daarin uitgelegd wat er is aangepast.
5. Nadat een Client een berichtje ontvangt over zijn abonnement geeft hij dit door aan zijn eigen database oftewel zijn miniMongo.
6. MiniMongo kijkt of de het een geldig mongo commando is.
7. MiniMongo voerd het commando uit op zijn database als het commando geldig is.
8. Zodra er iets in de miniMongo aangepast wordt worden de anpassingen van de database doorgegeven aan de viruele DOM.
9. De virtuele DOM verwekt de input en veranderd de DOM zonder hem te hoeven herladen.

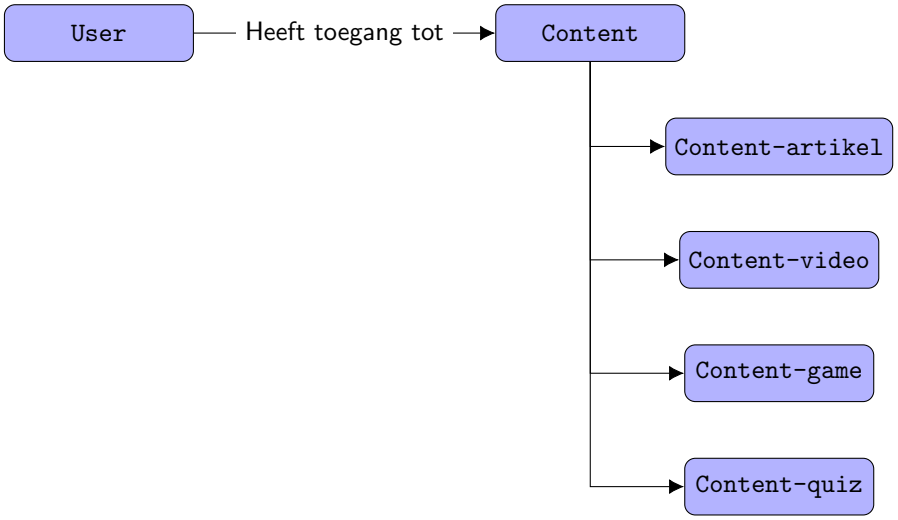
MongoDB is een nosql database die zijn content opslaat in json achtige files. MongoDB wordt daarna door het Meteor framwork uitgebreid met een miniMongo op de client en meerdere connecties tussen de virtuele DOM, de miniMongo en de MongoDB database op de server.

3.2 Gegevens opslag

De client zal niet alleen zijn eigen database hebben maar zal ook voorzien zijn van seccion variabelen. Hierin kan een cleint gegevens opslaan die uit principe nooit gedeeld zouden moeten worden of opgeslagen dienen te worden. Denk hierbij aan staat een menu open ja of te nee of moeten de zoek resultaten gefilterd worden op aantal likes of op aantal weergaven.

Chapter 4

ERD



5