# Eindwerk Webdevelopment

Syntra 2018-2019

Seppe Meerten

Inhoudstabel

1. Definitie van het project
2. Database
3. Laravel
4. Workflow
5. Budgetering
6. Mockups
7. Toelichting code
8. Conclusie

1 Definitie van het project

Voor mijn eindwerk maak ik een app/website om het leven van gitaristen makkelijker te maken.

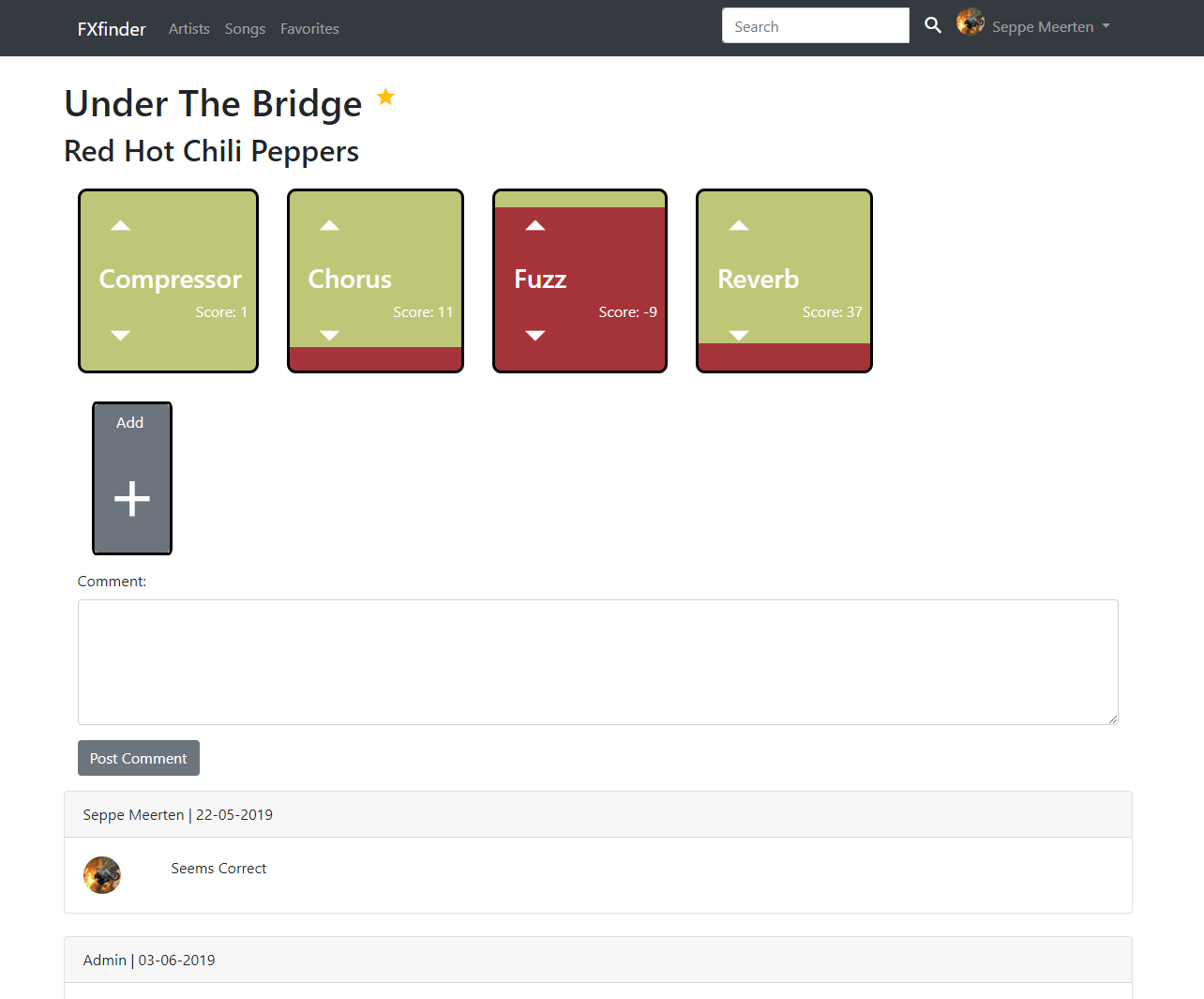
Er bestaan duizenden verschillende soorten gitaareffecten en het is (zeker voor een beginnende gitarist) vaak moeilijk om te herkennen welk soort effect er in een liedje gebruikt wordt.  
 Mijn app zou je simpelweg laten zien welk effect je bij welk liedje moet gebruiken. Hier ga ik er niet te specifiek op in welk merk of welk exact product de artiest gebruikt. Want niemand heeft al die effect pedalen. Wat velen wel hebben, is een kleine collectie goede pedalen en/of een multi-fx (een pedaal die digitaal heel veel verschillende soorten effecten kan nabootsen). Als ze dus weten welk soort effect ze nodig hebben, kunnen ze met behulp van wat ze al bezitten, vaak toch in de buurt komen van de juiste sound.

Ik kan niet voor elk liedje nagaan wat er exact gebruikt wordt, dus dit is iets wat ik (na een hoop dummy data in de development fase) aan de gebruikers overlaat.

Dit is een concept dat veel toegepast wordt in de online guitar community, met onder andere sites zoals Ultimate-Guitar, waar ijverige gitaristen enorm veel tabs (= notatie van hoe je een liedje op gitaar speelt) uploaden.

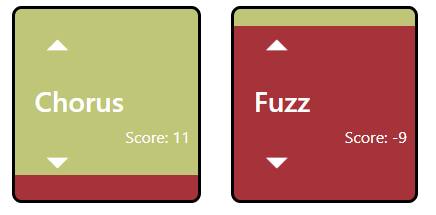
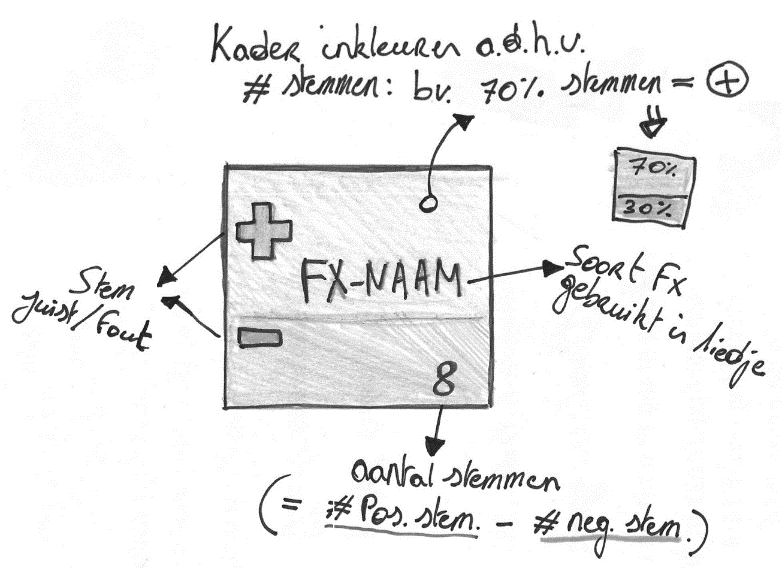
Als ik dus genoeg van deze ijverige gitaristen kan bereiken en kan aansporen om te helpen, geloof ik dat mijn app een waardige contributie aan de online gitaar community kan leveren.

Ik kijk alleszins alvast uit om niet elke keer te gokken, of door tientallen willekeurige forums moet scrollen elke keer als ik een liedje leer.



*- Voorbeeld van de pagina die voor elk liedje aangemaakt wordt.*

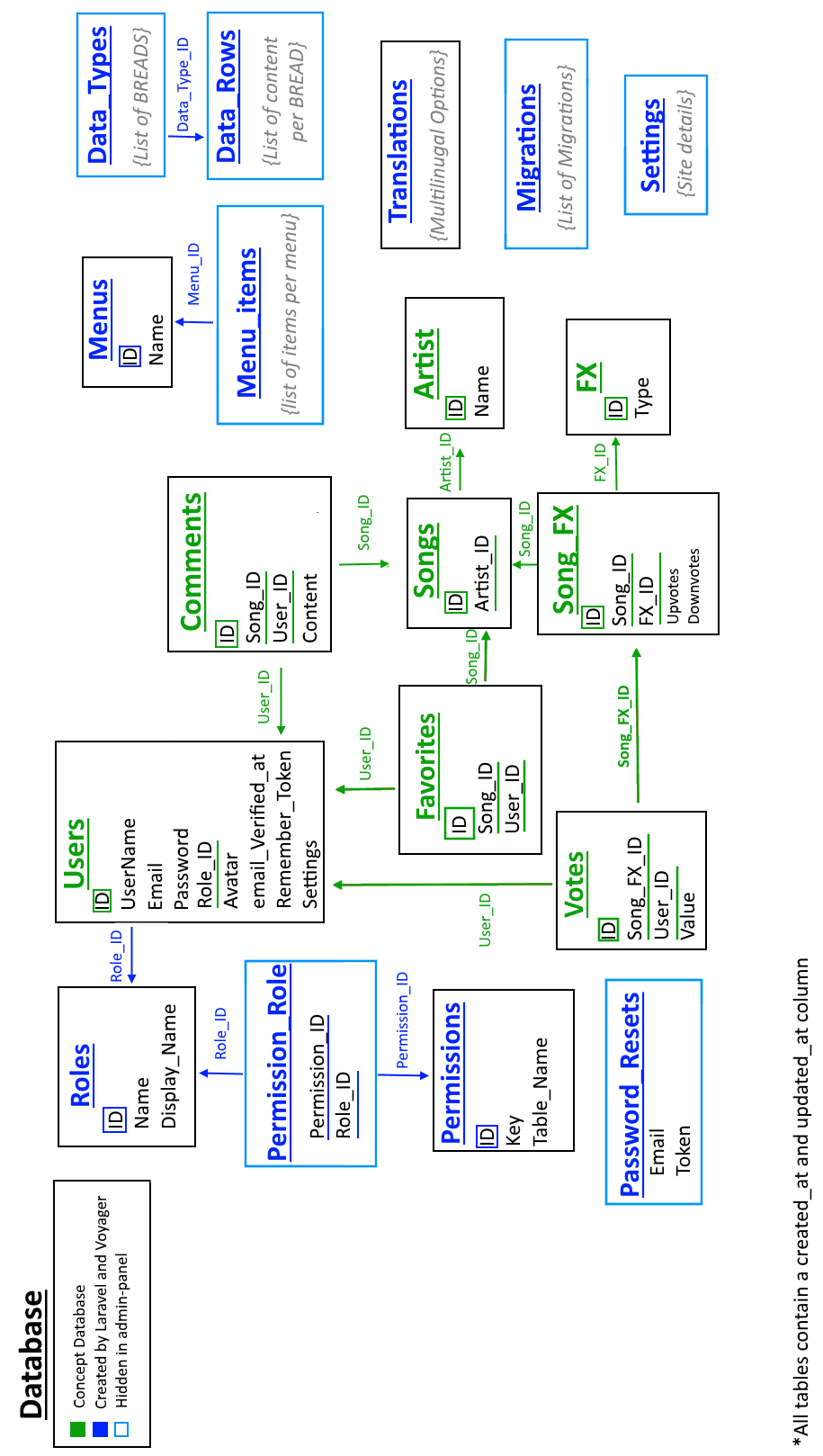
Natuurlijk kan ik niet verwachten dat alles wat geupload wordt correct is. Daarom implementeer ik een stem systeem, waarbij iedereen met een account kan stemmen wanneer een effect juist of fout werd toegevoegd aan een liedje. Hier bovenop kunnen gebruikers ook nog comments plaatsen om verdere discussie aan te gaan.

*- Voorbeeld ‘effecten block’ - Mockup ‘effecten block’*

Samengevat moet mijn app dus de volgende eigenschappen/functionaliteiten bezitten:

1. Een database met liedjes en hun artiesten.
   1. Een manier om hierdoor te navigeren (zoeken/oplijsten en weergeven).
      * Een read only die gebruikers te zien krijgen.
      * Een met CRUD functionaliteiten voor admins.
   2. Een manier om nieuwe liedjes en artiesten toe te voegen.
   3. Een persoonlijke favoriete pagina waar de gebruiker zijn favoriete liedjes aan kan toevoegen.
2. Gebruikers kunnen effecten aan de liedjes toevoegen.
   1. Er moet een lijst bestaan met effecten om uit te kiezen.
   2. Er moeten ook nieuwe effecten kunnen worden toegevoegd in het geval dat deze nog niet in de database staan.
   3. Op deze effecten kan voor juist of fout gestemd worden. (up- of downvotes)
3. Admins krijgen CRUD functionaliteiten (de opgelijste effecten, liedjes, artiesten, users, … aanpassen, toevoegen of verwijderen.)
4. Form acties moeten beschermd worden van cross-site request forgery (CSRF).
5. Gebruikers kunnen in- /uitloggen en registreren.
   1. Nieuwe gebruikers moeten eerst hun account verifiëren via email.
   2. Gebruikers kunnen enkel stemmen, commenten en gebruik maken van favorieten wanneer ze ingelogd zijn.
   3. Gebruikers kunnen hun profielfoto veranderen, en hun wachtwoord aanpassen.
      * Wachtwoord aanpassen gebeurt via email wanneer de gebruiker zijn wachtwoord vergeten is (reset). Anders kan de gebruiker zijn wachtwoord veranderen door eerst hun oud wachtwoord in te geven.
   4. Wachtwoorden moeten gehasht worden.
   5. Gebruikers kunnen gebruik maken van een ‘remember\_me cookie’.



2 Database

2.1 Users

Dit onderdeel bespreekt alle tabellen die iets te maken hebben met user authentication.

1. Users



De users tabel bevat alle gegevens van gebruikers.***R*ole\_id** is een *foreign key* en verwijst naar de roles tabel.

De tabel bevat de basisgegevens van de gebruiker: **name, email, avatar** (profielfoto) en **password.** Waarbij het email-veld uniek is om duplicaat gebruikers te voorkomen.

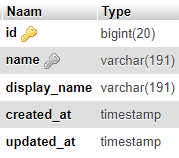
Daarnaast zijn er nog enkele velden nodig voor functionaliteiten in verband met de gebruiker:

**Email\_verified\_at** is nodig om te bevestigen dat de gebruiker zijn email geverifieerd heeft.

De **remember\_token** bevat een token aangemaakt door Laravel om ‘cookie hijacking’ te voorkomen. Deze token wordt steeds vernieuwd bij het in- en uitloggen zodat hackers geen cookies kunnen gebruiken om accounts over te nemen.

Hetveld **settings** werd voorzien door Voyager en bevat een object met een “locale”-instantie waarin de taal-voorkeur bewaard wordt. Ik heb echter wel geen andere talen dan Engels voorzien op mijn website.

1. Roles



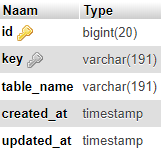
Roles worden toegekend aan gebruikers en bepalen hun functie op de website. Een gebruiker kan ofwel een normale user zijn, ofwel een admin (**name**). Deze rollen krijgen via de pivot tabel *Permission\_role* bevoegdheden toegekend in de *permissions-*tabel.

1. Permission\_Role



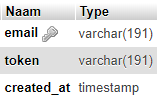
Permission\_role dient enkel als pivot tabel tussen *roles* en *permissions.*

1. Permissions



Deze tabel houdt alle bevoegdheden bij, waarin **key** de permission bijhoudt en **table\_name** de tabel waarin ze deze permission krijgen.

1. Password\_Resets

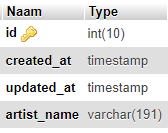


Wanneer een gebruiker zijn wachtwoord reset ontvangt hij een link om zijn wachtwoord te resetten via zijn **email**. Er wordt een random hashed string opgeslagen in het **token** veld en dus niet in de password kolom van de user. Hierdoor kan de gebruiker toch nog inloggen met zijn oud wachtwoord totdat het gereset wordt.

2.2 Content

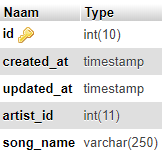
Dit onderdeel bespreekt de tabellen die de echte inhoud van de website bevatten.

1. Artists



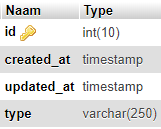
Deze tabel houdt de namen van alle artiesten bij.

1. Songs



Deze tabel houdt alle liedjes en hun artiesten bij. **Artist\_id** is een *foreign key* die verwijst naar de artiesten tabel.

1. Effects



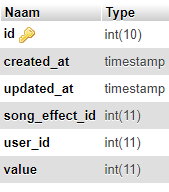
Effects bevat alle soorten (**type**)effecten.

1. Song\_Effects



Song\_Effects is een pivot tabel tussen *songs* en *effects*. De tabel vertegenwoordigd ook een effectenblock in de database. Elke effectenblock heeft een score die berekend wordt aan de hand van de **up**- en **downvotes**.

1. Votes



De votes tabel houdt bij welke gebruiker op welk effect gestemd heeft, en welke waarde (**value** = 1 staat voor upvote en = -1 aan downvote). Hiermee kan een gebruiker niet 2 keer hetzelfde stemmen op eenzelfde effect (maar wel aanpassen).

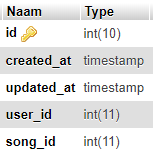
**User\_id** en **song\_effect\_id** zijn een *foreign key’s.*

1. Comments



Deze tabel bewaart alle comments (**content**)**,** wie (**user\_id**:*foreign key*) ze geplaats heeft, en waar (**song\_id**: *foreign key*).

1. Favorites

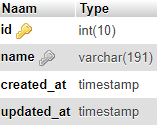


Deze tabel houdt per gebruiker een lijst met favoriete liedjes bij. **User\_id** en **song\_id** zijn *foreign key’s.*

2.3 Admin Panel

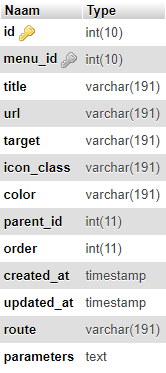
Dit onderdeel beschrijft de tabellen die Voyager gebruikt om het admin panel aan te maken en te beheren. Omdat deze automatisch gegenereerd worden ga ik hier niet te diep op ingaan.

1. Menus



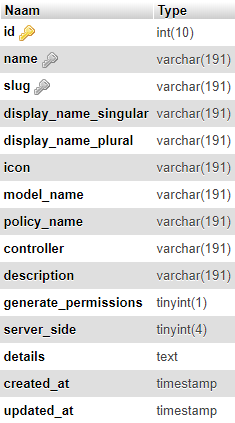
De menus tabel werd voorzien door Voyager om makkelijk menu’s te maken maar wordt enkel gebruikt voor het menu in het admin panel.

1. Menu\_Items

.

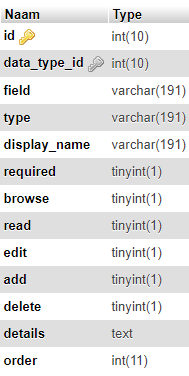
Deze tabel bevat de inhoud van de menu’s voorzien in de menus tabel. **Menu\_id** is de *foreign\_key* die de inhoud linkt aan de menus.

1. Data\_Types



De data\_types tabel beschrijft alle BREAD’s geassocieerd met de tabellen in de database.

1. Data\_Rows

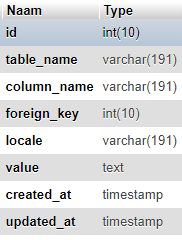


Data\_rows beschrijft wat de BREAD’s moeten weergeven. Deze worden gelinkt aan de juiste BREAD via de *foreign key* **data\_type\_id**.

2.4 Overige Tabellen

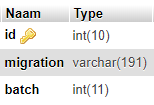
Ook de volgende tabellen werden allemaal aangemaakt door Voyager maar passen niet in een van de vorige catagoriën.

1. Translations



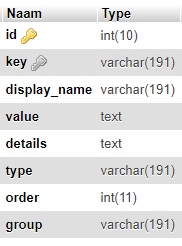
Deze tabel biedt de gebruiker de mogelijkheid om zijn taalvoorkeur aan te passen op de website. Dit is een mogelijkheid die Voyager biedt maar waar ik geen gebruik van maak.

1. Migrations



In deze tabel worden de namen van alle migration bestanden bijgehouden.

1. Settings



Deze tabel bevat de details van de website (zoals bv. title, description... en de Google Analytics Tracking ID).

3 Laravel

In dit puntje zou ik kort willen toelichten waarom ik Laravel gekozen heb en hoe ik bij afloop van mijn project terugkijk op die keuze.

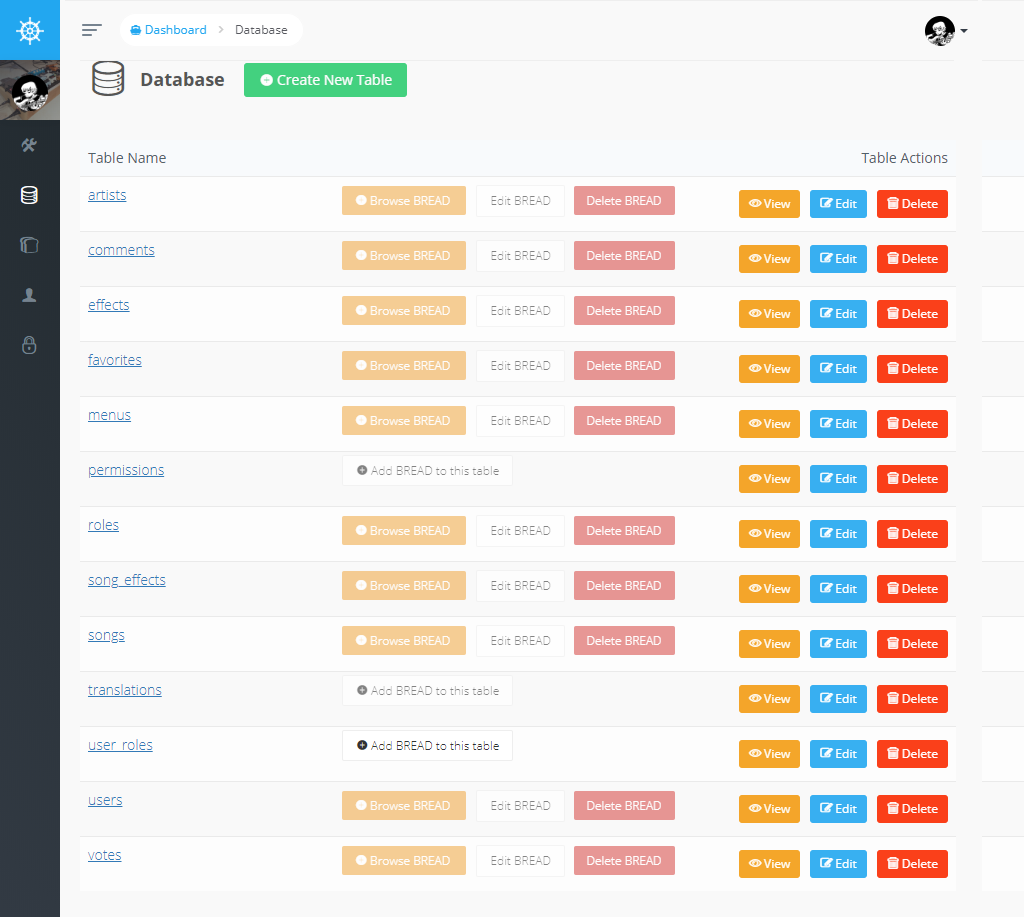
3.1 User Authentication

Voor mij was user authentication een van de grootste argumenten om voor Laravel te kiezen. Met slechts een simpel artisan command: php artisan make:auth maakt Laravel een volledig login/registratie systeem met (bijna) alles erop en eraan. (email verificatie, basis beveiliging, remember me, password reset, …)

3.2 Voyager

Voyager is een Laravel admin package dat een volledig aanpasbaar database management system aanmaakt voor CRUD operaties, media management en nog veel meer.

Mijn admin panel is volledig met Voyager gemaakt. Het database menu vervangt bijna volledig phpMyadmin. De BREAD’s (Browse, Read, Edit, Add en Delete) kunnen aangepast worden om te kiezen welke kolommen waar zichtbaar moeten zijn en relaties te leggen tussen verschillende tabellen.



3.4 Security

Laravel biedt standaard al de volgende security features:

* Laravel Authentication System
* Password encryption using bcrypt()
* Reduce Laravel Vulnerabilities From CSRF (Cross Site Request Forgery)
* Protection against XSS (Cross Site Scripting)
* SQL Injection
* Improve Laravel Application Security
* Laravel Security Packages

Daar bovenop staat het team achter Laravel ervoor bekend om snel loopholes uit hun framework te halen.

3.5 Evaluatie Keuze

Over het algemeen ben ik zeer tevreden over mijn keuze om met Laravel te werken. Hoewel het werken met een framework, waar ik in het verleden bijna niet in gewerkt heb, ook zijn nadelen had. Ik had verwacht de features Laravel sneller onder de knie te krijgen. Dit is te merken in het verschil tussen de code die ik eerst geschreven heb t.o.v. mijn latere code. Veel functies die ik later geleerd heb waren zeker handig geweest in die oudere code.

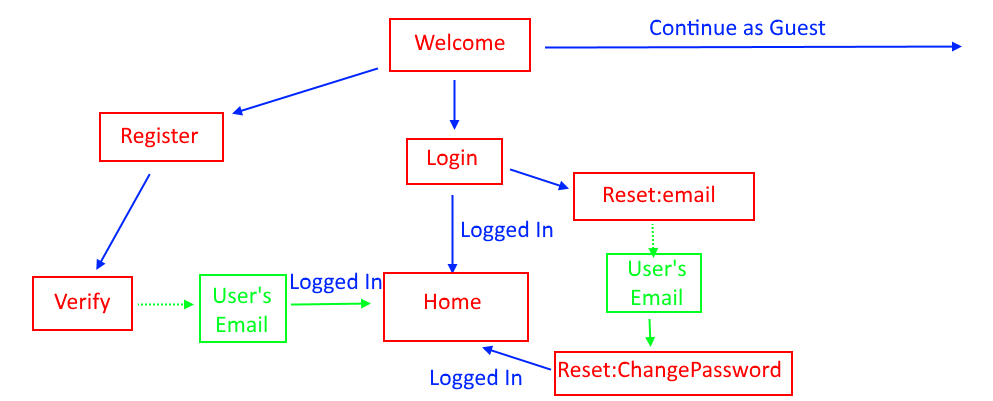
4 Workflow

Dit onderdeel beschrijft hoe de gebruiker door de website navigeert.

4.1 De gebruiker surft de website binnen.

Gebruikers hebben 3 mogelijkheden als ze binnen komen:

1. Registreren
2. Inloggen
3. Doorgaan zonder in te loggen



1. De gebruiker surft door naar de registratie-pagina en geeft een gebruikersnaam, email en wachtwoord op.

Hij wordt vervolgens doorgestuurd naar de email verificatie-pagina. Hij kan nu via email zijn account verifiëren en doorgestuurd worden naar de home-pagina. De gebruiker is nu ingelogd.

1. De gebruiker surft door naar de login-pagina en geeft zijn email + wachtwoord op.

Hij wordt vervolgens doorgestuurd naar de home-pagina en is ingelogd.

Indien de gebruiker zijn wachtwoord niet meer weet, kan hij op de *forgot your password* link klikken waarop doorgestuurd wordt naar de reset:email pagina. Hier moet hij zijn email opgeven. Vervolgens kan hij via zijn email doorgestuurd worden naar de reset:changepassword-pagina en zijn wachtwoord aanpassen. Van daar wordt de gebruiker doorgestuurd naar de home-pagina. Hij is nu ingelogd.

* Gebruikers die ingelogd zijn krijgen toegang tot 2 nieuwe pagina’s: My account en Favorites

1. De gebruikers kunnen ook door de pagina navigeren zonder in te loggen. Hierdoor kunnen ze wel een aantal dingen niet doen: Comments plaatsen, liedjes/artiesten/effecten toevoegen, effecten up- /downvoten en liedjes markeren als favorieten.

Vanaf hier ga ik ervan uit dat gebruikers ingelogd zijn. Dit maakt geen verschil in workflow behalve de hierboven vermelde functionaliteiten. Deze gebruikers kunnen altijd nog inloggen of registreren vanuit de andere pagina’s.

*-Header wanneer de user uitgelogd is.*



*-Header wanneer de user ingelogd is.*

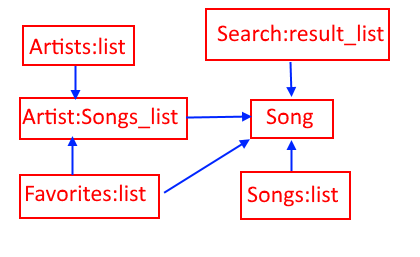


4.2 De gebruiker surft naar zijn account-pagina.

Op deze pagina kan de gebruiker zijn gegevens bekijken, zijn wachtwoord aanpassen en zijn profielfoto bewerken.

4.3 De gebruiker wil naar een liedje surfen.

Gebruikers kunnen op verschillende manieren bij liedjes terecht komen:

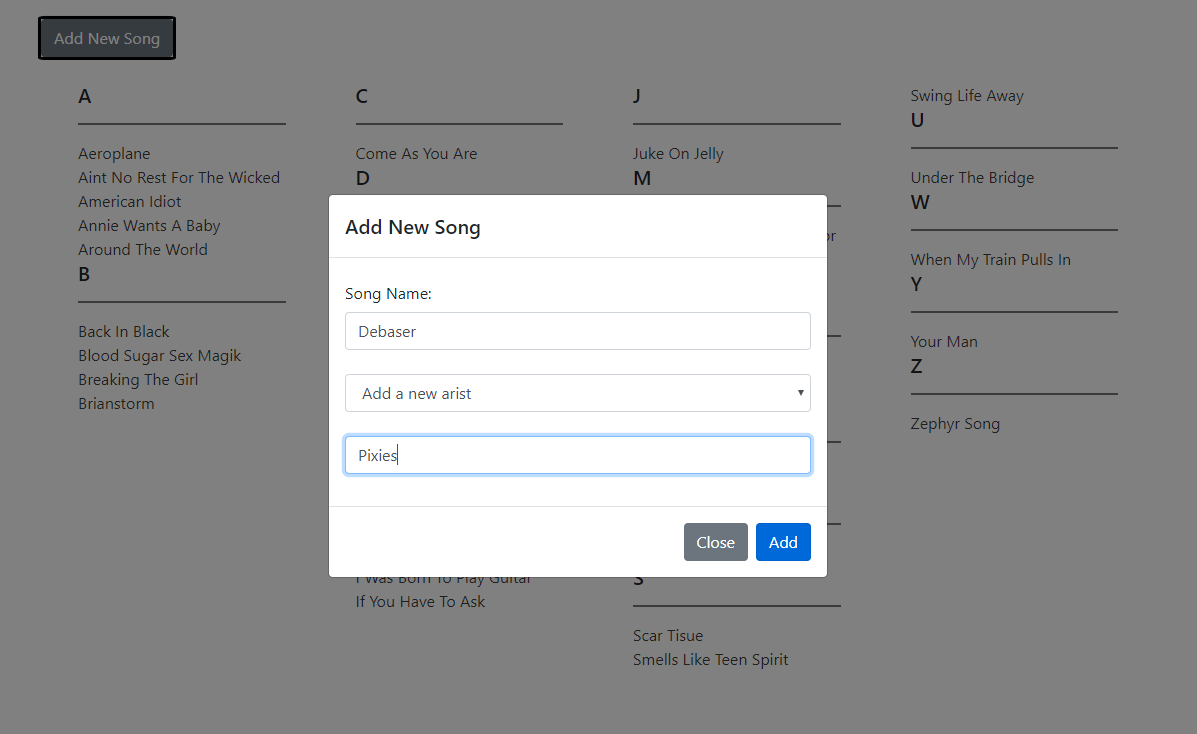


1. De gebruiker surft naar de artiesten-pagina en selecteert de artiest van het liedje dat hij zoekt.

Hij wordt door gestuurd naar een pagina met alle liedjes van die artiest. Hier selecteer hij een liedje en wordt naar de pagina van dat liedje doorgestuurd.

1. De gebruiker surft naar de liedjes-pagina en selecteert een liedje uit de lijst. Vervolgens wordt hij doorgestuurd naar het liedje dat hij zoekt.

Hier heeft de gebruiker ook de mogelijkheid om een nieuw liedje toe te voegen door op de ‘add song’ knop te klikken. Hij wordt gevraagd de naam van het liedje in te geven en een artiest te selecteren. Indien de artiest nog niet in de database bestaat kiest hij ‘add a new artist’. Er verschijnt een nieuw tekstveld waarin hij de artiestennaam kan invullen. Hij klikt op de add knop om het liedje toe te voegen.

*-Liedjeslijst met het ‘Add a new song’ venster op de voorgrond.*

1. De gebruiker surft naar zijn favorieten-pagina, hier krijgt hij een lijst met alle liedjes (en hun artiesten) die hij als favoriet heeft aangeduid. Hij kan weer navigeren naar een liedje door er een te selecteren. Hij kan ook de artiest selecteren om door te gaan naar een de lijst van alle liedjes van die artiest.

De gebruiker heeft hier ook de optie om liedjes te verwijderen uit de lijst.

1. De gebruiker zoekt een liedje op in de zoekbalk, hij ontvangt een lijst met alle liedjes waar de zoekterm in de naam van het liedje voorkomt. Hij navigeert naar het liedje door er een te selecteren.

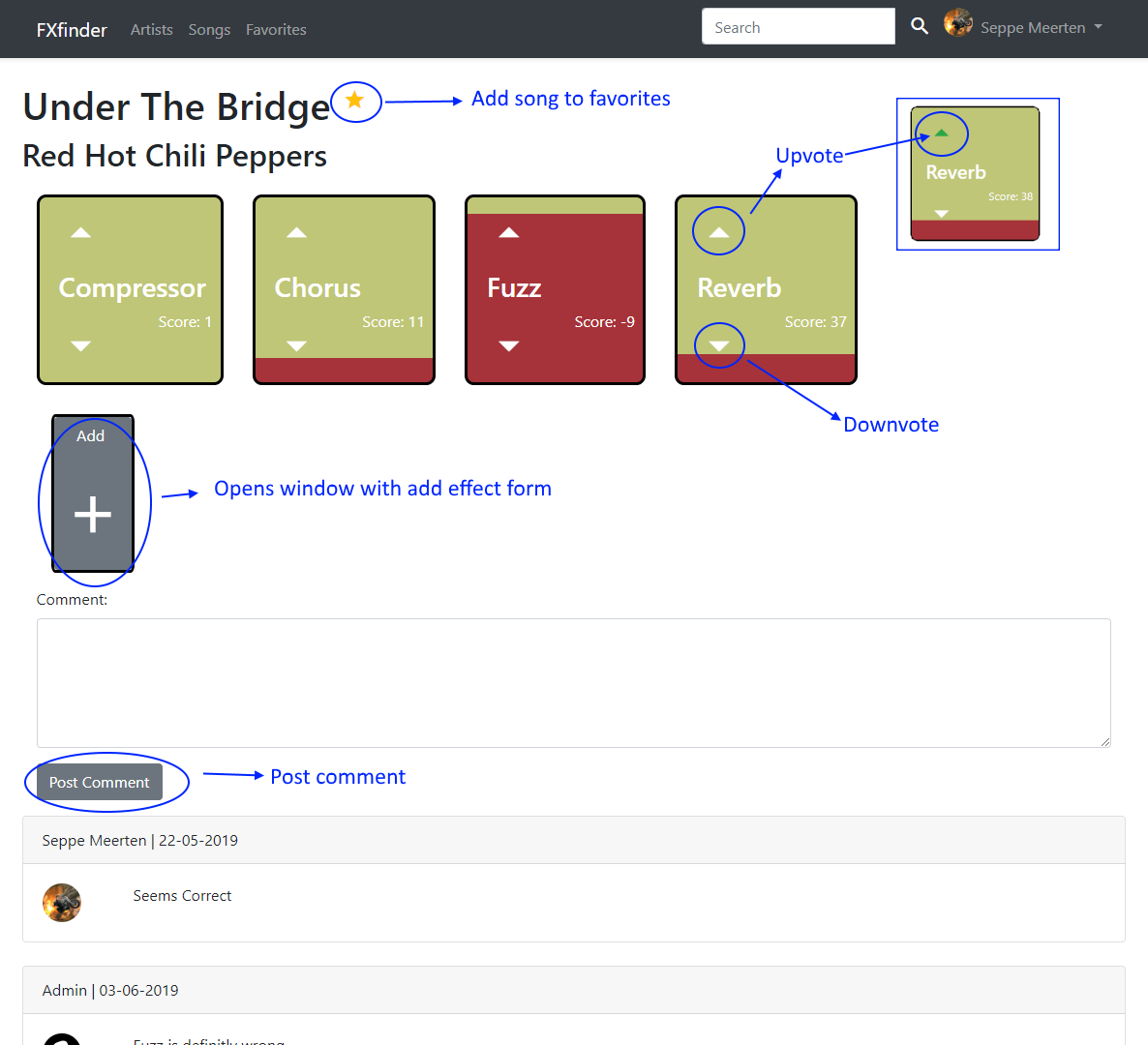
4.3 De gebruiker surft naar een liedje.

Dit is de pagina waar alles om draait. Hier kan de gebruiker eindelijk zien welke effecten hij nu moet gebruiken om dat liedje te spelen.

Hier kan hij andere effecten toevoegen als hij denkt dat nog iets ontbreekt. Het liedje toevoegen aan favorieten. Comments plaatsen als hij er vragen bij heeft en stemmen welke effecten hij denkt juist of fout zijn.

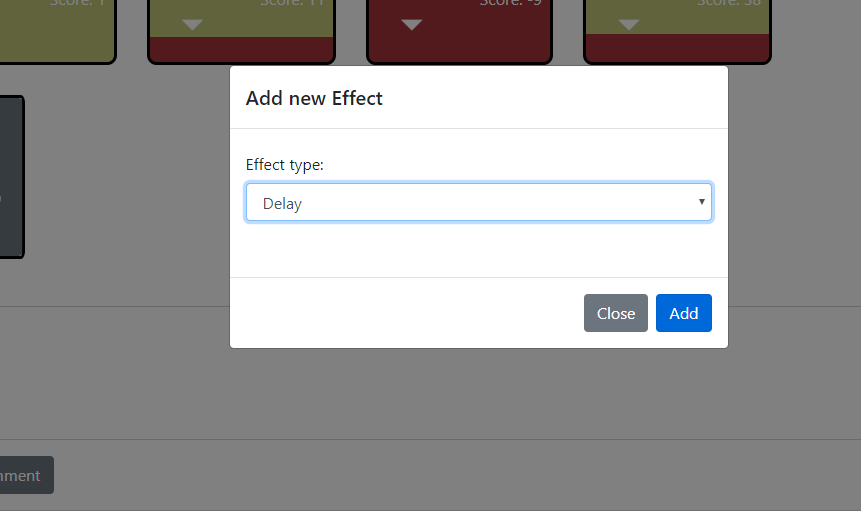
Eenzelfde gebruiker kan natuurlijk geen 2x dezelfde stem uitbrengen op een effect. Hij kan echter wel zijn stem aanpassen of annuleren door nogmaals op hetzelfde te klikken.

*-Pagina van een liedje*



De gebruiker kiest een effect uit een lijst met bestaande effecten die al in de database staan. Mocht het gewenste effect er niet tussen staan, dan kan hij ‘add a new effect’ selecteren waarbij een nieuw tekst-veld tevoorschijn komt en hij een nieuw effect aan de database kan toevoegen.

-*Window dat geopend wordt wanneer de gebruiker een effect toevoegt.*



5 Budgettering

Aangezien mijn website geen producten verkoopt of diensten verleent, moet ik op een andere manier aan geld geraken. Mijn Businessplan is advertenties hosten op mijn website. Omdat mijn website zo een specifiek publiek aanspreekt, zal het niet moeilijk zijn om goede marketingpartners te vinden. (Grote online muziekshops zoals bax-shop.be bijvoorbeeld)

Dit wil zeggen dat ik in het begin altijd verlies ga maken. En enkel als de website populair wordt, er geld te verdienen is.

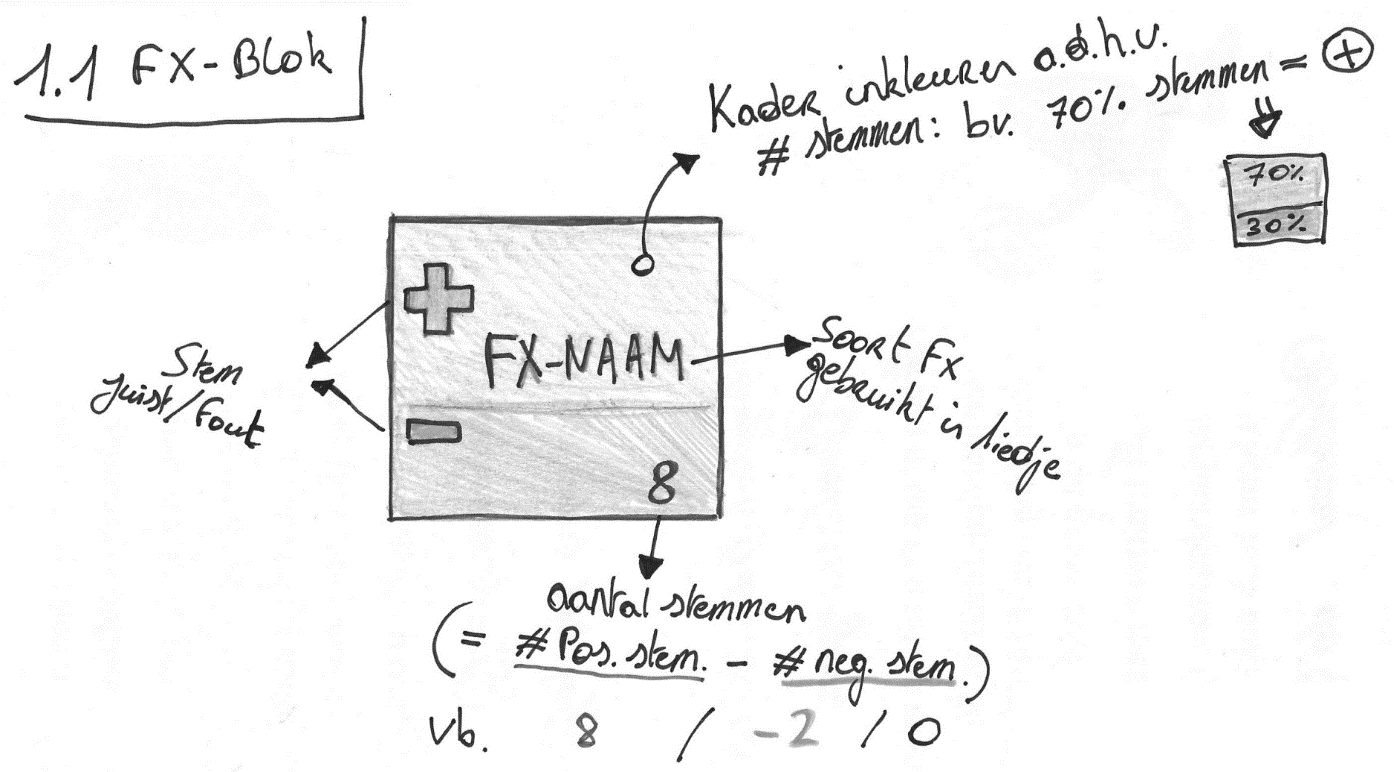
Kosten:

1. Serverkosten (€9.67 per maand via ovh)
2. Werkuren (72 uur aan €46 per uur = €3,312)

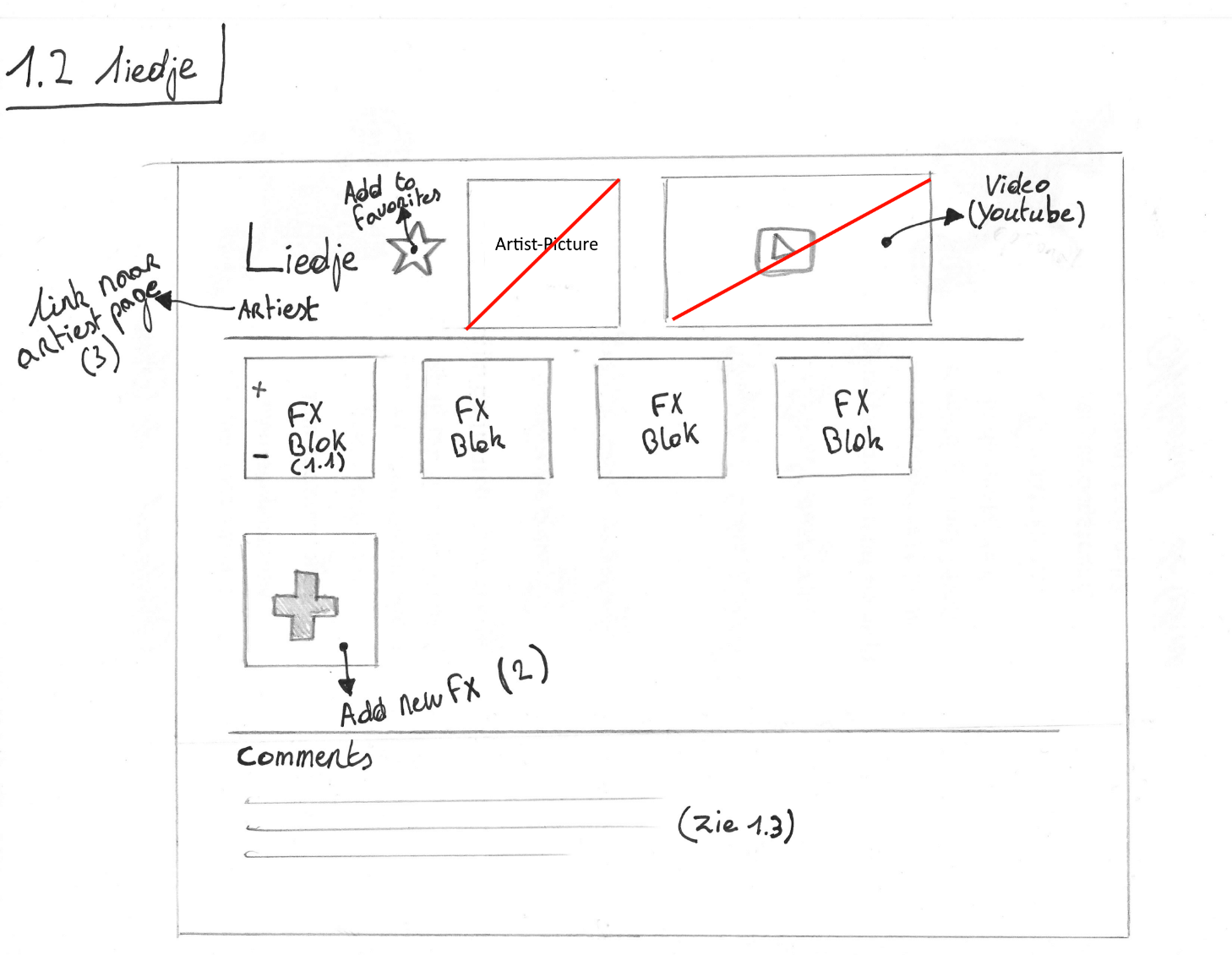
Uit mijn kosten volgt dat ik pas echt winst maak wanneer ik meer dan €3,312 + €9.67 per verlopen maand verdien. Deze €3,312 heb ik er echter niet werkelijk moeten investeren. Dus zolang ik meer dan €9.67 per maand verdien, kan ik de website online laten staan. Deze drempel zou ik binnen het half jaar moeten overschrijden, anders leg ik me neer bij mijn verlies en stop ik met de website.

6 Mockups

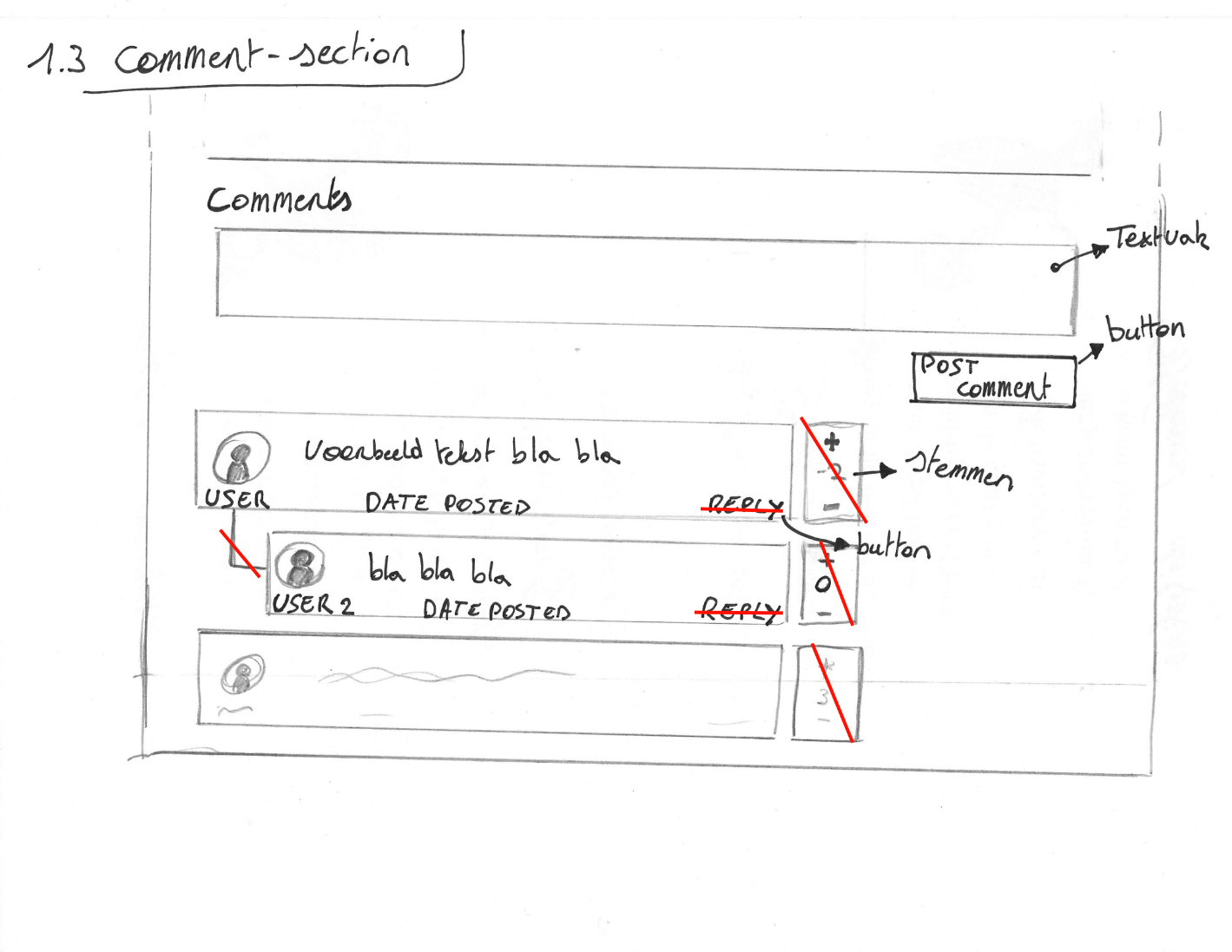
De volgende afbeeldingen zijn de originele schetsen van hoe ik me mijn website voorstelde. Ik heb hier en daar in het rood een aantal features/functionaliteiten met rood doorstreept die het niet tot de uiteindelijke versie van de website gehaald hebben (Wegens tijdsgebrek, of omdat ik het simpelweg geen goed idee meer vond.)



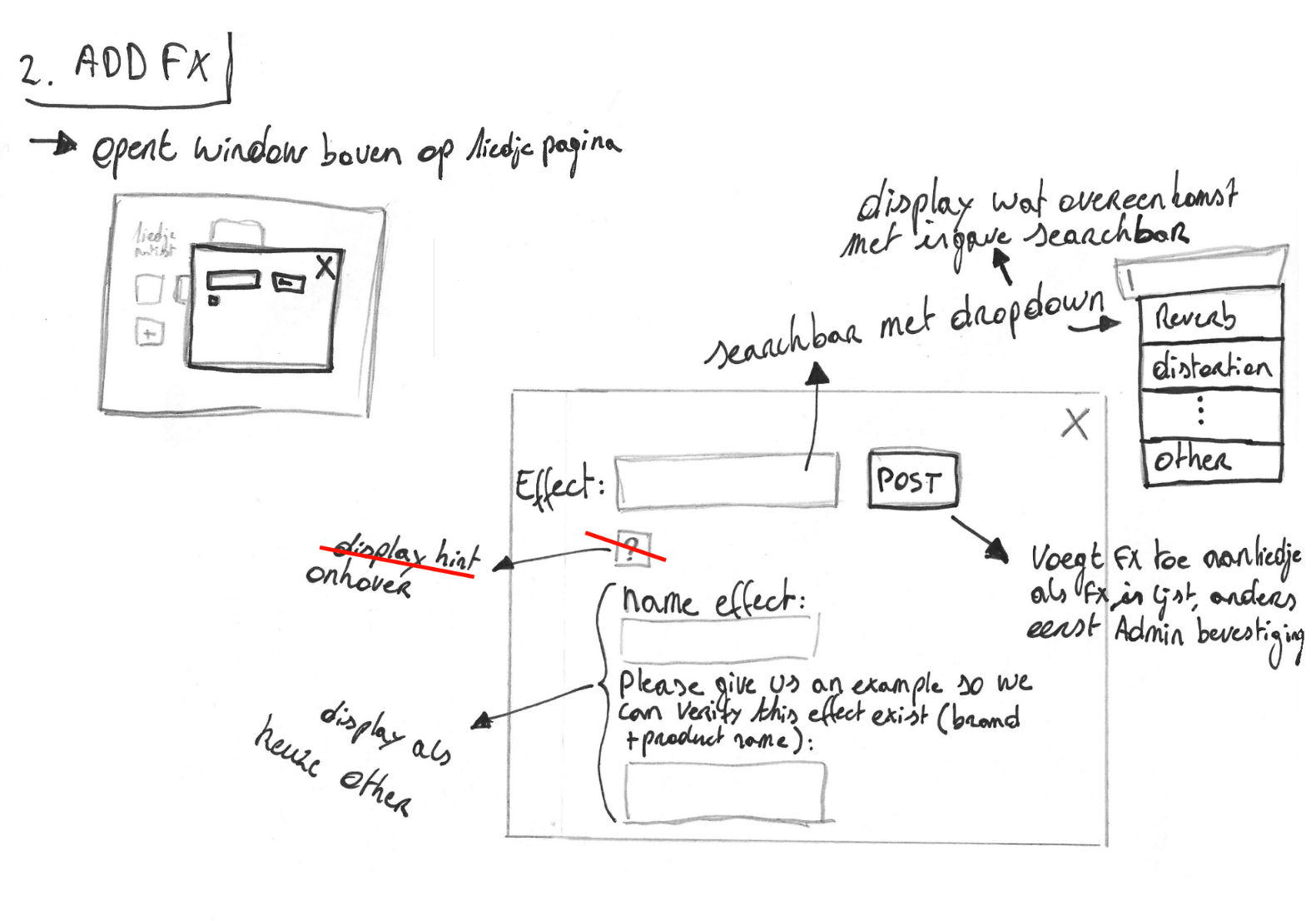
*- Het volledige concept van mijn website draait rond deze effect-block. Het idee is snel te kunnen zien wat juist of fout is.*



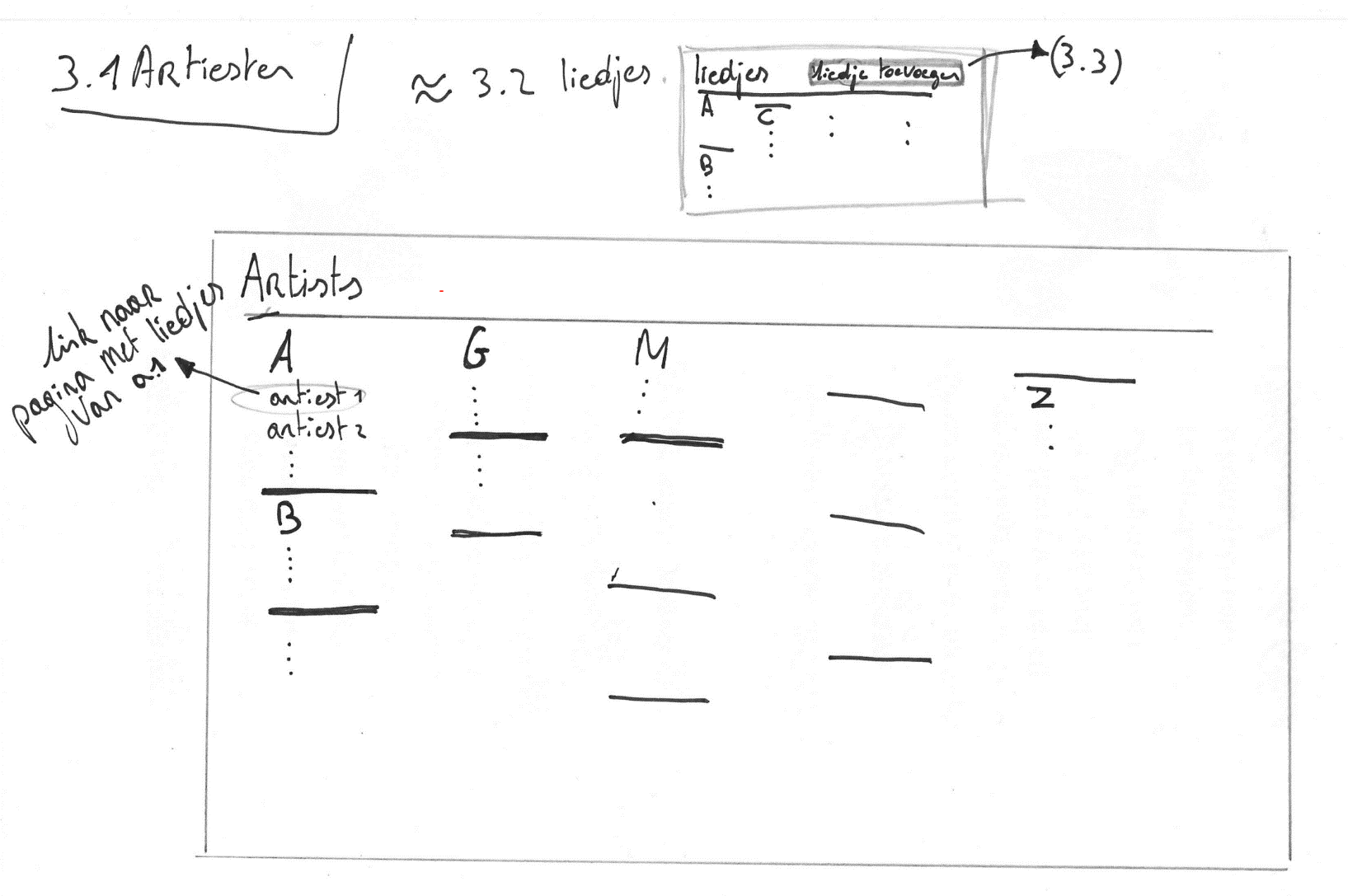
*-Hier heb ik heb de afbeelding en video laten vallen omdat ik geen manier vond om dit te automatiseren, en het ook geen feature is om aan de gebruiker toe te vertrouwen.*

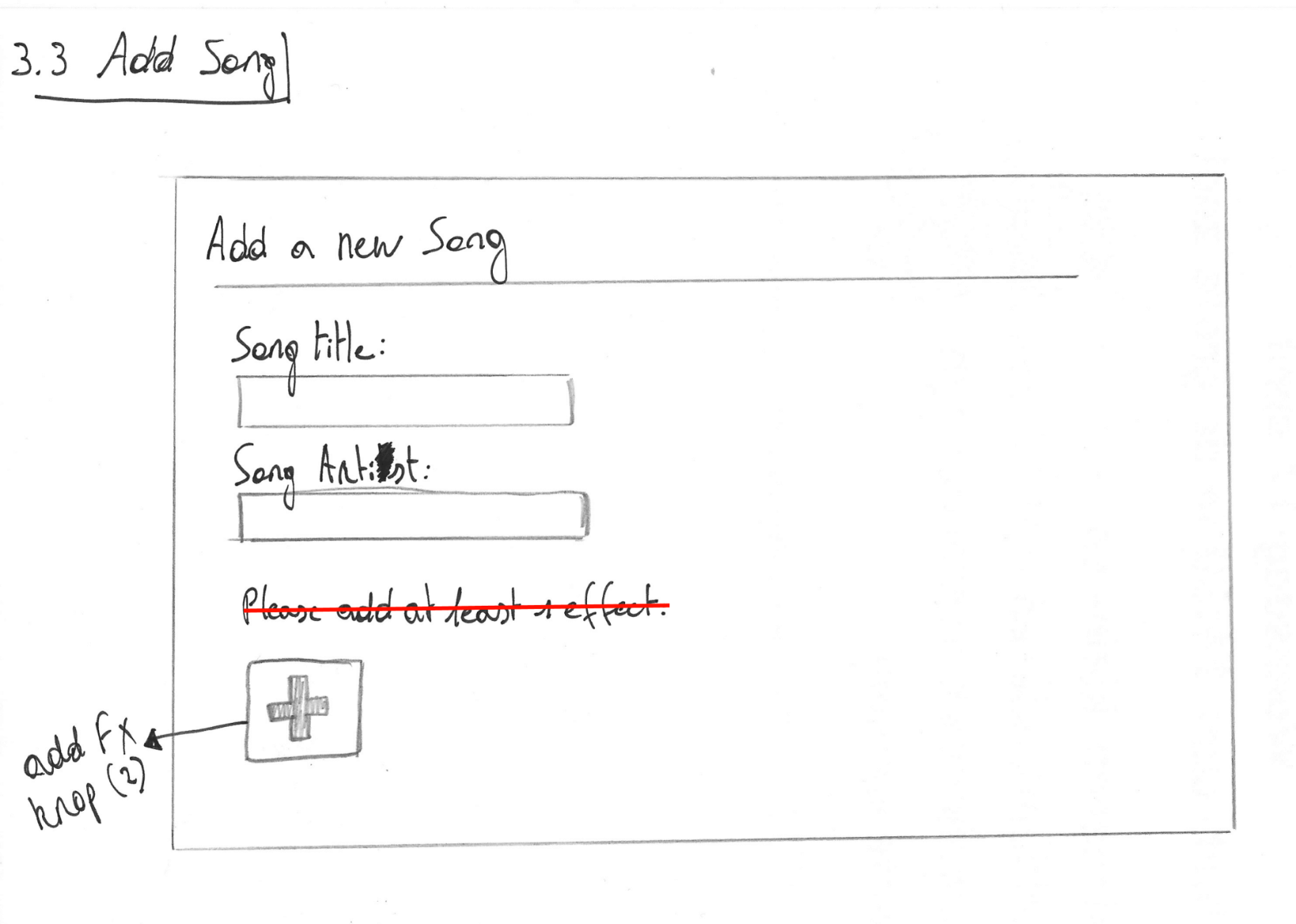


*-Een voting en reply systeem voor mijn comments kunnen later nog toegevoegd worden.*



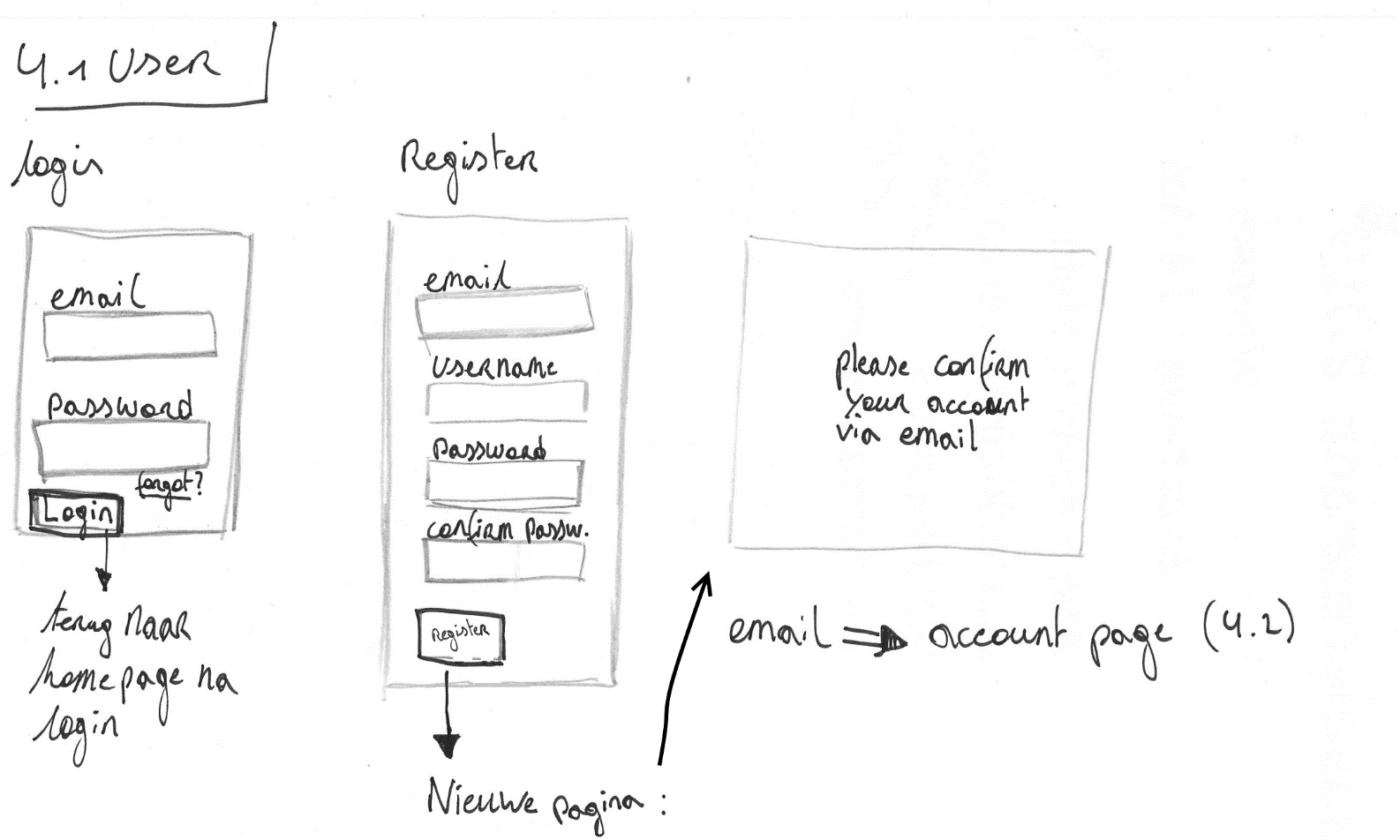
*-Toen ik bij het toevoegen van effecten aankwam heb ik besloten om de gebruiker te laten kiezen uit een lijst met effecten en niet elke keer zelf een in te geven. Hierdoor vermijd ik dat gebruikers een effect toevoegen dat al in de database onder een andere naam staat.*

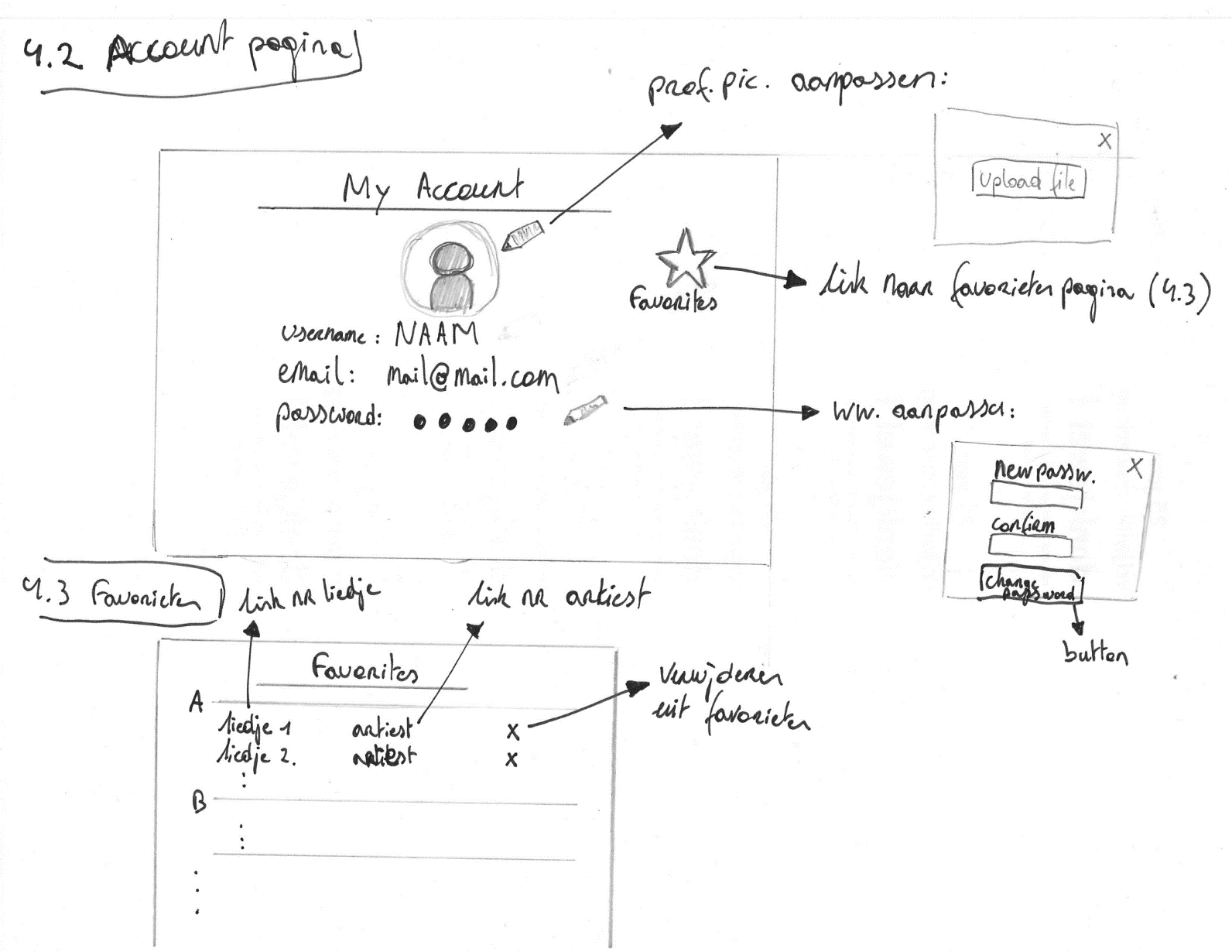




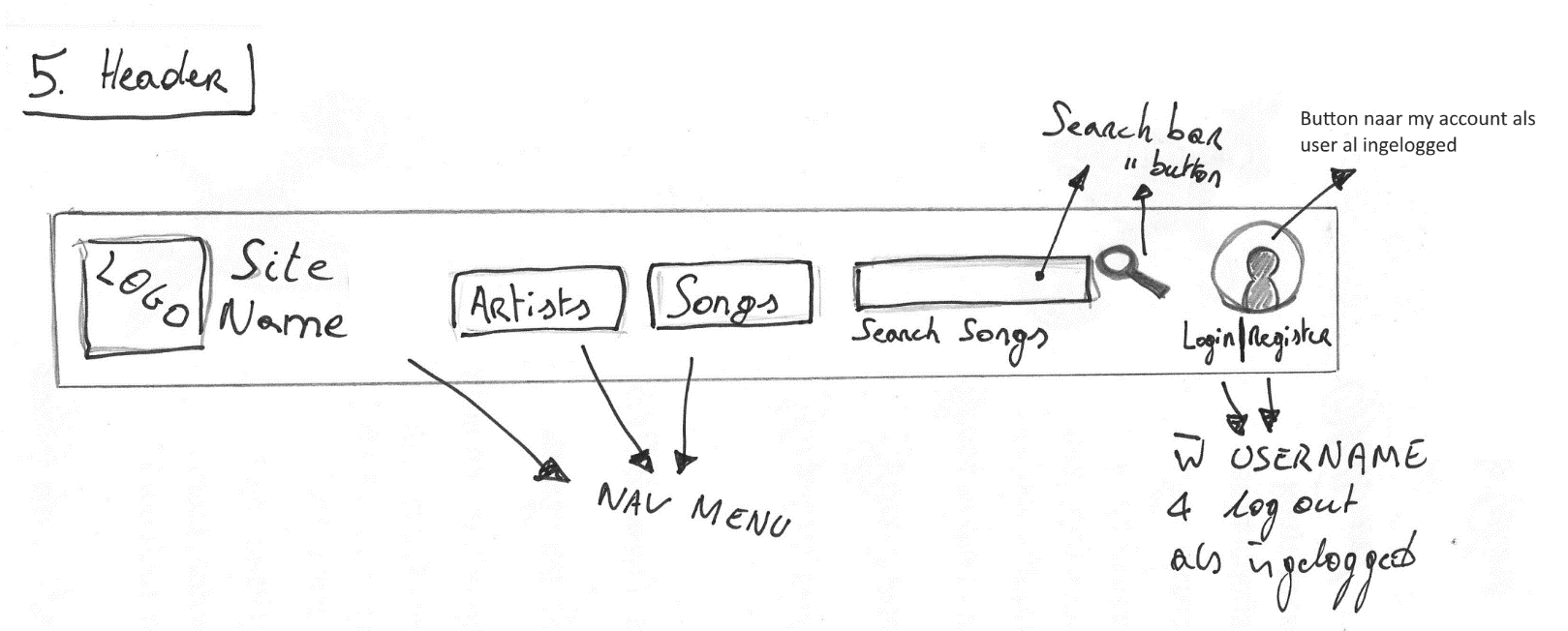
*-De restrictie dat gebruikers verplicht zijn om een effect op te geven wanneer ze een nieuw liedje toevoegen heb ik laten vallen.*

*Ik heb ook het input-veld voor de artiest veranderd naar een dropdown met een lijst van artiesten die reeds in de database bestaan. Dit heb ik gedaan voor het gemakzucht van de gebruiker.*





*-De link naar de gebruiker zijn favorieten heb ik verplaatst naar de header (wanneer de user ingelogd is).*



7 Toelichting Code

Hier licht ik een aantal belangrijke onderdelen in mijn code toe. De source files van mijn website zijn allemaal verkrijgbaar op: <https://github.com/meertenseppe/EindwerkApplicatie>;

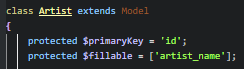
7.1 Models

In deze bestanden staan alle functies die de database *query’en*. Ze *query’en* elk hun overeenkomstige tabel in de database. Deze functies worden later gebruikt in de controllers.

Hier worden alle models opgelijst met hun doel. Er worden ook een aantal functies kort uitgelegd.

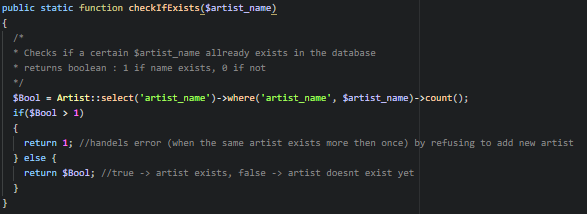
In het begin van elk model wordt een *primary key* en *fillable* gedefinieerd. Fillables zijn mass assignable.

1. Artist.php



Deze functies worden voornamelijk gebruikt om uiteindelijk een artiestenaam te kunnen weergeven.

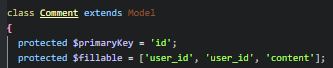
Het bevat ook een functie **checkIfExists($artist\_name);**



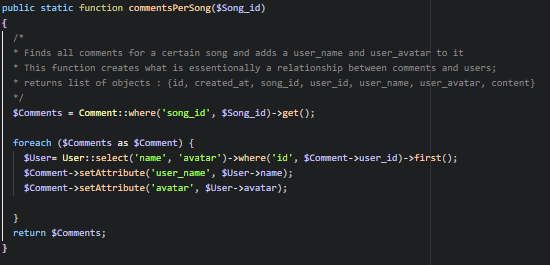
Deze functie zoekt of een bepaalde artiest al in de database zit.

Veel andere models bezitten een gelijkaardige functie.

1. Comment.php



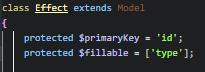
Dit model bevat maar één functie: **commentsPerSong($Song\_id);**



Het zoekt alle comments die bij een bepaald liedje horen en geeft deze terug samen met de user\_name die deze comment geplaatst heeft en zijn user\_avatar.

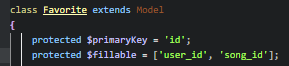
Dit is het enige model dat een andere tabel dan zichzelf oproept.

1. Effect.php

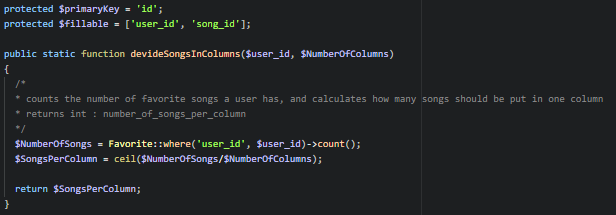


Dit model haalt voornamelijk effecten uit de database om weer te geven om de Song pagina.

1. Favorite.php



Dit model bevat maar één functie: **devideSongsInColumns($user\_id, $NumberOfColumns);**



Het berekent het aantal liedjes dat er in een kolom moeten staan, opdat elke kolom evenveel liedjes bevat (voor de oplijsting van favorieten).

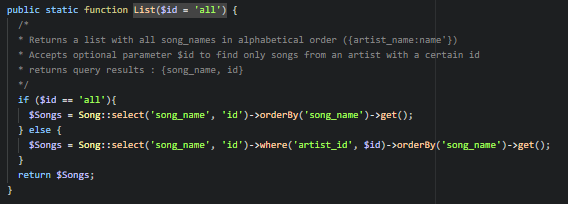
Ook deze functie heeft veel gelijkaardige functies in andere models.

1. Song.php



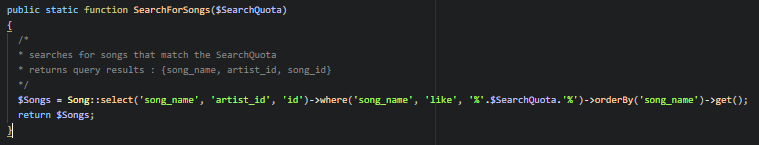
Dit is een belangrijk model aangezien een groot deel van de website liedjes bevat. Het heeft dan ook een heel aantal verschillende functies.

De functie **List($id = 'all');** is ook weer zo een functie die regelmatig gelijkaardig terugkomt in andere models.

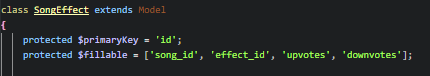


Deze functie zal ofwel naar àlle liedjes in de database ofwel naar slechst één. Dit is afhankelijk van of er een $song\_id parameter meegegeven wordt aan de functie.

De functie **SearchForSongs($SearchQuota);** werkt gelijkaardig, maar dan enkel voor alle liedjes waar de naam een bepaalde zoekterm (**$SearchQuota**) bevat.

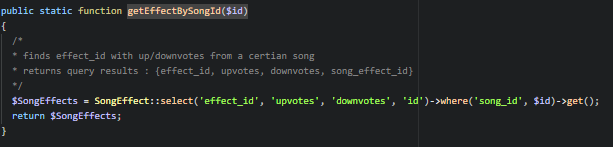


1. SongEffect.php

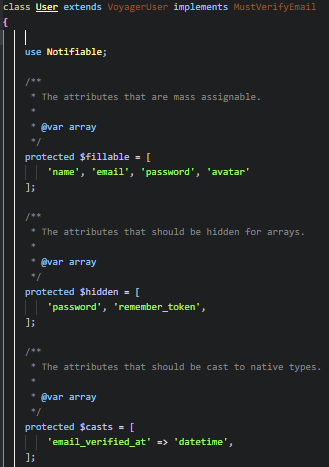


Dit model bevat enkel een functie om de juiste effecten aan een liedje te binden.

**getEffectBySongId($id);**



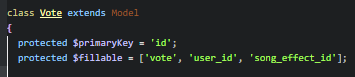
1. User.php



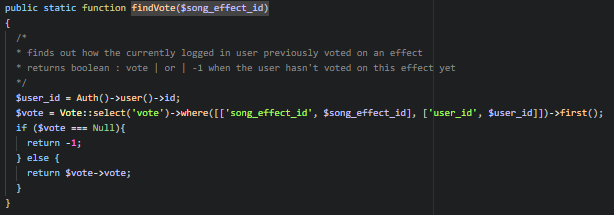
Dit model breidt uit op de VoyagerUser class (aangemaakt door Voyager) en gebruikt nog enkele built in Laravel classes voor Authentication.

De meeste User methods bestaan al in door Laravel aangemaakte bestanden die niet besproken worden in deze paper.

1. Vote.php



Het Vote model wordt gebruikt om te controleren of een gebruiker op een effect gestemd heeft, en zo ja hoe. Dit doet het aan de hand van de functie: **findVote($song\_effect\_id);**

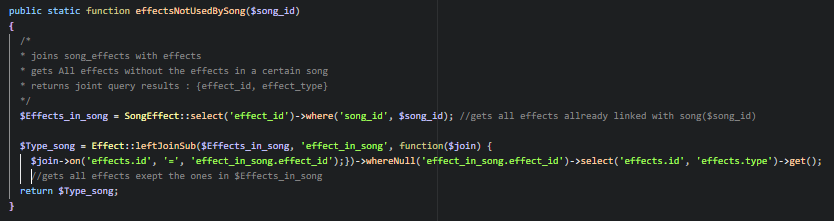


In de volgende models oefen ik joins uit op de verschillende tabellen die ze aanroepen.

1. SongEffect\_Effect.php

Hier wordt de tabel song\_effects met de tabel effects gejoind.

De functie **effectNotUsedBySong($song\_id);** is nodig om enkel de effecten, die nog niet aan een bepaald liedje toegekend zijn, op te lijsten wanneer de gebruiker een effect toevoegt. Zo kan hij geen effect selecteren dat al toegekend is.



1. Favorite\_Song\_Artist.php

Hier worden de tabellen: favorites, songs en artists gejoind.

Het model bevat maar één functie: **getFavoritesByUser($id);** Deze functie haalt alle favoriete liedjes van een bepaalde user op en verbindt deze met de naam en artiest van dat liedje.



7.2 Controllers

Controller bestanden doen een aantal dingen. De belangrijkste hieruit zijn:

* Views laden en variabelen meegeven:



* De user redirecten naar een ander pagina:



* Gegevens opslaan, aanpassen en verwijderen uit de database:



* Requests valideren en verwerken.



Hier worden een aantal controllers opgelijst met hun doel. Er worden ook een aantal functies besproken.

1. Auth\*controller*

De controllers in de Auth folder werden aangemaakt door Laravel met het php artisan make:auth command. Ze handelen over alles wat met login / registratie te maken heeft.

1. ArtistsController.php

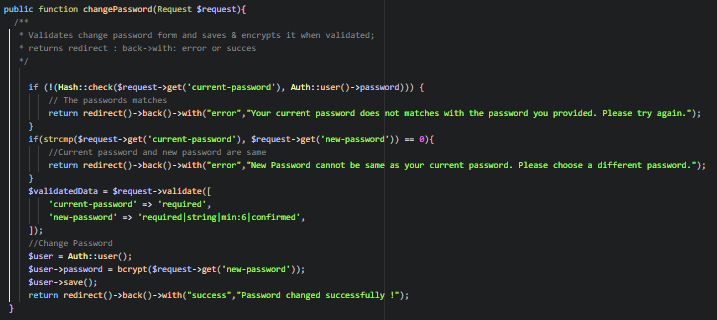
Deze controller laadt enkele views in. Er worden 2 views geassocieerd met Artists:

Artists:List en Artist:Song\_List

1. ChangePasswordController.php

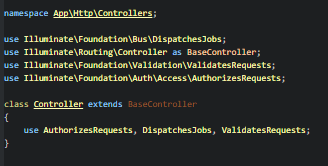
Deze controller laadt en verwerkt het *change password* form. De input wordt gevalideerd, encrypt en opgeslagen. Wanneer de input de validatie niet slaagt, ontvangt de gebruiker een error.

De verwerking gebeurt a.d.h.v. de functie **changPassword(Request $request);**



1. Controller.php

Deze controller verbindt mijn controllers met Laravel zijn baseController en een aantal andere Laravel functionaliteiten.



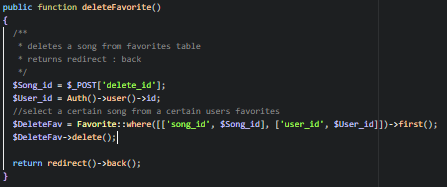
1. CommentsController.php

Hier worden comments opgeslagen in de database. Gebruikers die niet ingelogd zijn, krijgen een foutmelding dat ze zich eerst moeten inloggen.

1. FavoritesController.php

Deze controller laadt het favorites:list view. Hier voegt het ook nieuwe favorieten toe of verwijdert ze uit de tabel.

Verwijderen gebeurt met de **deleteFavorite()** functie:

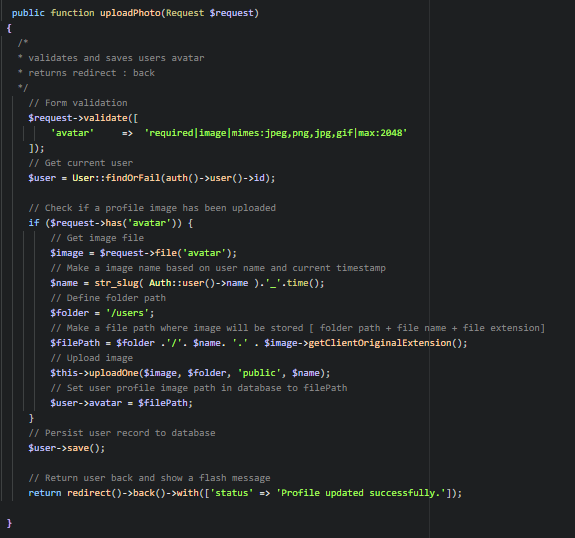


1. HomeController.php

Deze controller laadt de homepagina. Als de gebruiker niet ingelogd of geverifieerd is, wordt hij doorgestuurd naar de loginpagina of verificatiepagina.

1. MyAcountController.php

Hier wordt het view myAccount opgeladen. Deze class gebruikt het uploadTrait, zodat de gebruiker een afbeelding kan uploaden. Dit gebeurt met de functie **uploadPhoto(Request $request);**



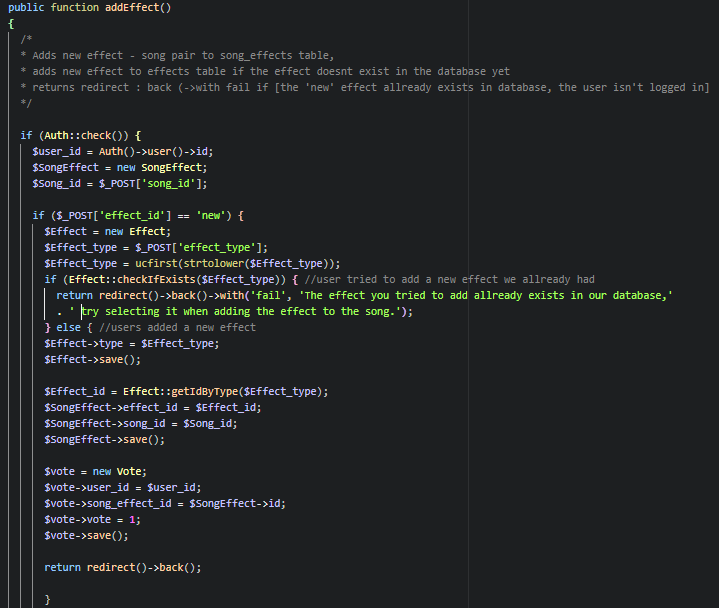
De functie valideert eerst of het gegeven bestand het juiste formaat heeft, slaat de afbeelding op in storage\users en het path naar de afbeelding in de database.

1. SearchController.php

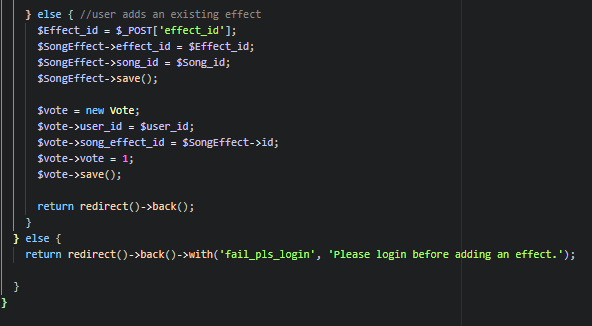
Deze controller laadt het view Search:Song\_List en gebruikt de model functie **SearchForSongs($SearchQuota);** om de juiste liedjes mee te geven.

1. SongEffectController.php

De SongEffectController bevat slechts één functie: **addEffect();** Deze wordt opgeroepen wanneer de gebruiker een effect aan een liedje probeert toe te voegen.



Als de gebruiker ‘add a new effect’ selecteert, geeft dat een effect\_id = ‘new’ mee. In dit geval maakt de controller een nieuw record in de *effects* tabel aan. Er wordt een nieuw record met een effect-liedje paar toegevoegd aan de *song\_effects* tabel. Een *song\_effect* wordt standaard aangemaakt met één upvote. Deze stelt de upvote van de gebruiker die hem aanmaakt voor. Daarom moet er ook een nieuw record toegevoegd worden aan de *vote* tabel om bij te houden dat de gebruiker al op dat effect gestemd heeft.



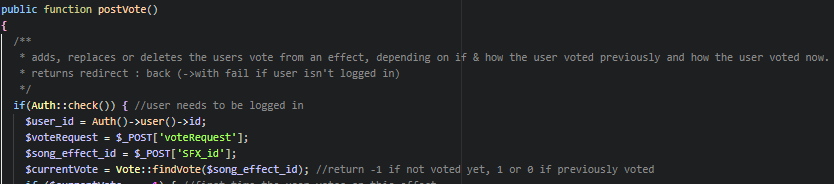
Als de gebruiker een bestaand effect selecteert, geeft dat een effect\_id = ‘Id van dat bestaand effect’ mee. Hier is het niet meer nodig om een nieuw record aan te maken in de *effects* tabel.

1. SongsController.php

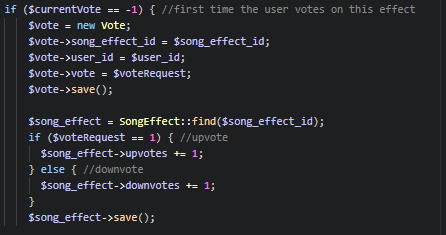
De SongsController laadt 2 views: Songs:List en Song. Deze controller gebruikt heel veel models omdat de Songpagina zoveel gegevens nodig heeft.

Op de Songs:List pagina moet de gebruiker een nieuw liedje kunnen toevoegen. Dit gebeurt met de **addNewSong();** functie.

Op de Song-pagina kan de gebruiker stemmen op een effect. Hiervoor bevat de controller een functie: **postVote();**

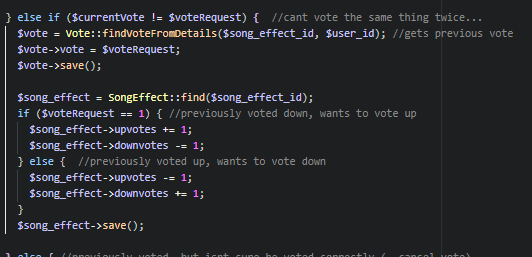


De ingelogde gebruiker ($user\_id) stemt ($voteRequest, up=1, down=0) op een bepaald effect ($song\_effect\_id). Als hij voor de eerste keer op dit effect stemt ($currentVote = -1) dan wordt er een nieuw record toegevoegd aan de tabel *vote.* Het effect krijgt een extra up- /downvote (afhankelijk van $voteRequest).

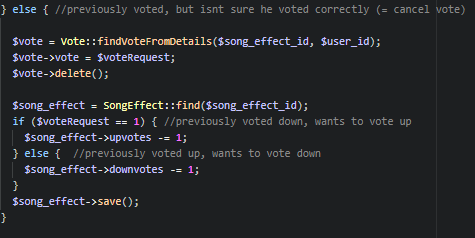


Als hij eerder al op dit effect stemde mag deze nieuwe stem er niet gewoon bij optellen.

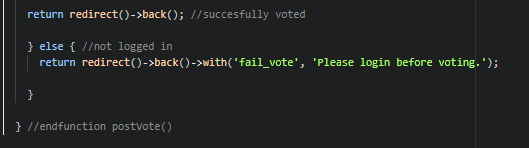
Dus als hij anders stemt dan vorige keer ($currentVote != $voteRequest), dan wordt zijn stem aangepast in de *vote* tabel. Het effect waar hij op stemde krijgt een up- /downvote bij en een down- /upvote minder.



Als hij hetzelfde stemt als vorige keer ($currentVote == $voteRequest) dan wordt zijn vorige stem geanulleert. Zijn stem wordt uit de *vote* tabel verwijderd en het effect verliest een up- /downvote.



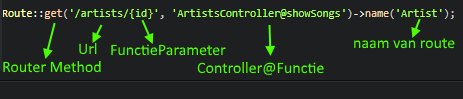
Op dit punt heeft de controller de nieuwe stem opgeslagen of ondervonden dat de gebruiker niet ingelogd was. De gebruiker wordt teruggestuurd naar de liedjes pagina waar hij vandaan kwam.



7.3 Routes

Dit onderdeel bespreekt kort het bestand web.php.

Hier worden de web routes van de applicatie geregistreerd. Deze worden geladen door de RouteServiceProvider binnen een groep die de “web” middleware groep bevat.



Vb. <http://localhost/mijnFiles/Eindwerk/Eindwerk/public/artists/6> gebruikt dus decontroller ArtistsController en roept de functie showSongs(6) op.

7.4 Views

Deze bestanden bevatten de front-end. Deze code wordt volledig verwerkt naar HTML, CSS en Javascript.

Laravel gebruikt het templating engine: Blade. Dit maakt het makkelijk om verschillende view bestanden op een pagina te gebruiken. Hier worden de view.blade bestanden kort besproken.

1. App.blade.php

Dit bestand bevat de volgende structuur:

|  |
| --- |
| <html>  <head>  <!--CSRF TOKEN-->  <!-- Scripts -->  <!-- Fonts -->  <!-- Styles -->  @yield(‘head’)  </head>  <body>  <!--header navbar-->  @yield(‘content’)  </body>  </html> |

Alle andere view.blade bestanden breiden deze structuur uit.

In het <head> element zit standaard een *CSRF Token*, een link naar het *app.js script* van Laravel, een aantal fonts, links naar de css bestanden app.css en *main.css, bootstrap* en *Material icons.*

@yield(‘head’) laadt de inhoud van de section (‘head’) uit het uitbreidende view. Dit bevat altijd een <title> en soms een css of javascript bestand.

De navbar geeft verschillende content weer wanneer de gebruiker wel of niet ingelogd is.

@yield(‘content’) laadt de inhoud van de section(‘content’) uit het uitbreidende view.

De volgende bestanden gebruiken allemaal @extends(‘layouts.app’)

1. \Auth\

De map Auth bevat een aantal views die te maken hebben met authentication (login, registration, changePassword, …)

1. \Artists\
   1. Artist.blade.php

Dit blade bevat een lijst met alle liedjes van één bepaalde artiest. Deze worden opgedeeld in 3 kolommen en alfabetisch gerangschikt weergegeven. Dit soort view komt vaak voor op mijn website en wordt besproken in *5. Favorites.blade.php.*

* 1. ArtistList.blade.php

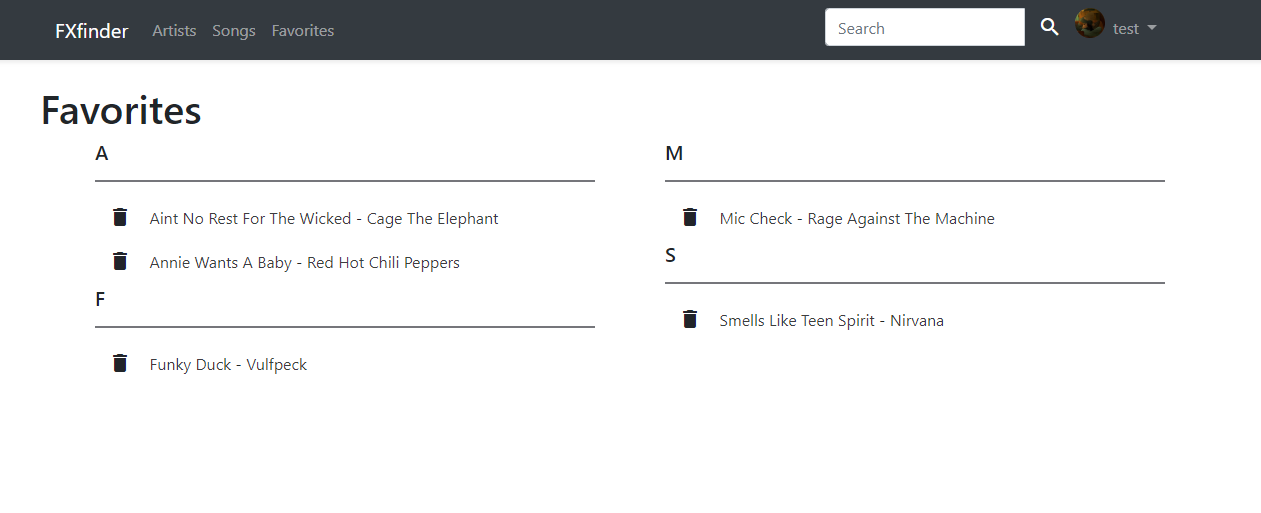
Dit blade bevat ook een gerangschikte lijst, opgedeeld in kolommen. Maar nu een lijst van alle artiesten in de database.

1. Search.blade.php

Dit blade bevat ook een gerangschikte lijst, opgedeeld in kolommen. Maar nu een lijst van alle liedjes die overeenkomen met de zoekterm.

1. Favorites.blade.php

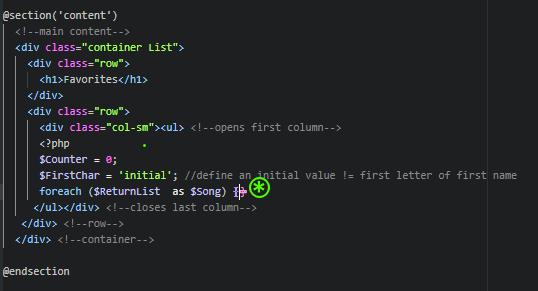
Dit blade bevat ook een gerangschikte lijst, opgedeeld in kolommen. Maar nu een lijst van alle liedjes (en hun artiest) die tot de gebruiker zijn favorieten behoren. Hier kan de gebruiker ook liedjes uit de lijst verwijderen.



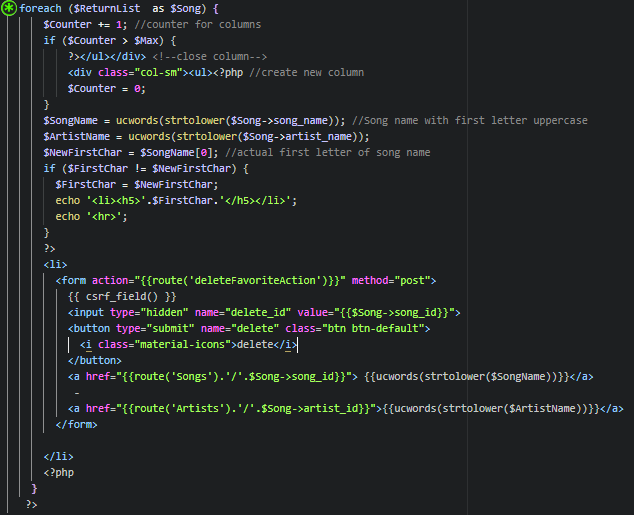
De lijst is een reeks van bootstrap kolommen met een <ul> in. Via de controller ontvangt het view de variabele *$max.* Deze bevat het aantal liedjes dat in een kolom mag staan (met de overschot in de laatste kolom).

De view doorloopt alle liedjes, en houdt door middel van een counter bij hoeveel liedjes er al in de kolom zitten. Wanneer deze het $*max* aantal liedjes bevat start het een nieuwe kolom.

De eerste letter van het huidige liedje wordt bijgehouden in *$FirstChar* en steeds vergeleken met de eerste letter van het volgende. Wanneer deze verschilt van het vorige wordt er een hoofding gemaakt met de nieuwe letter.



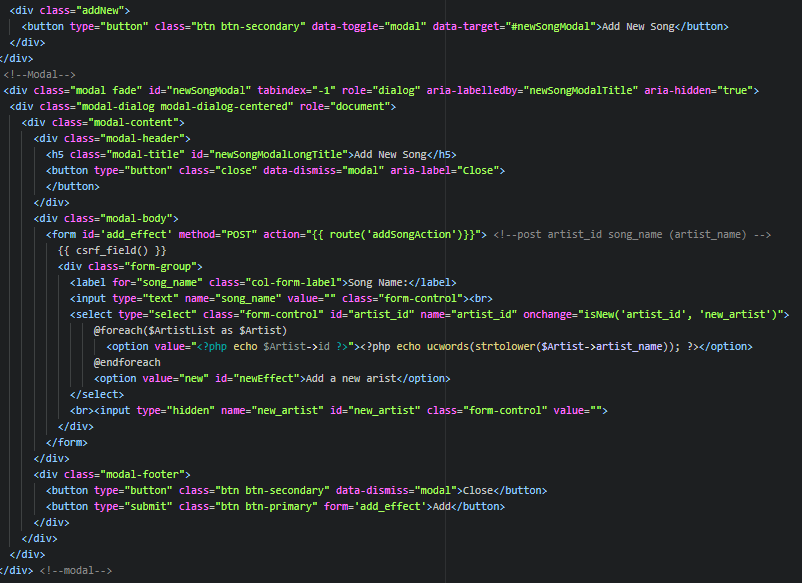
Voor elk liedje wordt een delete-knop, naam en artiestennaam gemaakt.



1. \Songs\
   1. SongList.blade.php

Dit blade bevat ook een gerangschikte lijst, opgedeeld in kolommen. Maar nu een lijst van alle liedjes in de database.

Deze pagina heeft ook een ‘add Song’ knop. Wanneer hierop geklikt wordt, komt een bootstrap modal met een form tevoorschijn.



Dit form bevat 3 inputs. Een tekst input voor de naam van het liedje. Een select input, met alle artiesten als keuze. En een verborgen tekst input om een nieuw effect toe te kunnen voegen. Deze komt tevoorschijn wanneer de gebruiker bij de select input ‘add a new artist’ selecteert.

* 1. EffectBlock.blade.php

Dit blade genereert alle effectblokken die op de liedjespagina terecht komen.

Een effectblok wordt, afhankelijk van het percentage van upvotes t.o.v. downvotes, groen-rood ingekleurd via css (met lineair-gradient).

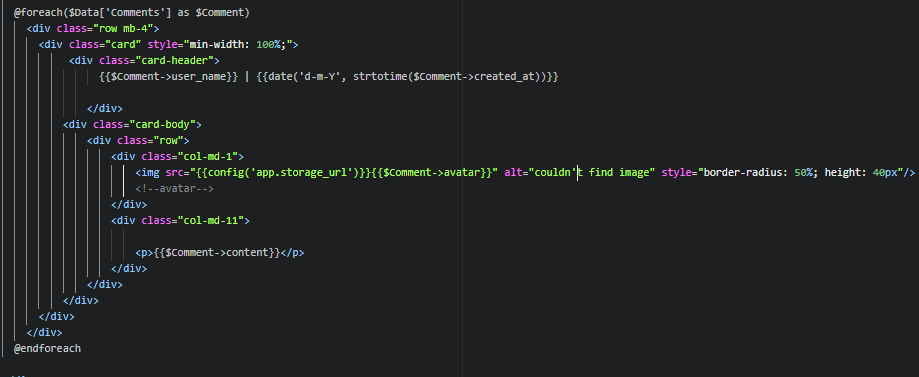
Elke blok bevat daarbovenop nog de naam van het effect en een score (= upvotes – downvotes). De up- /downvote knoppen zijn 2 icons die 1 of –1 kunnen posten, ze worden ingekleurd afhankelijk van wat de gebruiker gestemd heeft.

* 1. Song.blade.php

Elk liedje heeft een Song pagina. Hier worden al zijn effectblokken en comments weergegeven.

Het bevat, gelijkaardig aan de songList pagina een ‘add effect’ knop die een bootstrap modal weergeeft. Maar hier moet de gebruiker een effect selecteren in plaats van een artiest.

Elke comment is een bootstrap card, met gebruikersnaam en datum als titel. De body van de card bevat de profielfoto van de gebruiker en de content van de comment.



7.5 Bootstrap en CSS

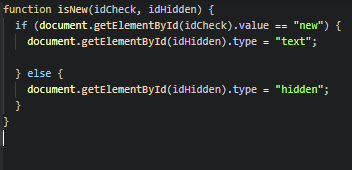
De opmaak van de website maakt voornamelijk gebruik van bootstrap. Er zijn echter toch nog 2 CSS-bestanden gelinkt: main.css en Laravel’s app.css.

7.6 Javascript

Ik heb zelf maar één javascript functie moeten schrijven. Deze staat in het Main.js bestand dat ik link in de nodige views.

De functie **isNew(idCheck, idHidden);** dient om het extra inputveld bij de 'add song’ en ‘add effect’ forms weer te geven.

Wanneer de gebruiker een ander effect / artiest selecteert, controleert hij of die waarde ‘new’ terug geeft. In dit geval komt het extra inputveld tevoorschijn.



8 Conclusie

Ik heb met plezier aan dit eindwerk gewerkt. Het was een uitdaging al doende Laravel aan te leren. Ik heb ook geleerd om in het vervolg eerst op het gemak de features van een framework te bekijken voordat ik zomaar code begin te schrijven. Door meer built in Laravel methodes en classes te gebruiken zou mijn leven deze afgelopen maand een stuk gemakkelijker geweest zijn.

Uitbreiden?

Mijn website kan ook nog zeker wat extra features en opmaak gebruiken. Ik denk dan aan search filters voor mijn zoekbalk, een voting system voor de comment section, comments kunnen aanpassen, verwijderen en op reageren, ... Maar niet alleen technische features, ook een uitbreiding op het concept (wat voor gitaar/pickups, versterker, …) zijn mogelijk.

Ik ga ook zeker eens polsen op online forums (reddit, ultimate-guitar, …) om te kijken of er werkelijk interesse is in dit concept. Afhankelijk van de reacties zou ik mijn project nog verder kunnen uitwerken.

Afrondend kan ik stellen dat ik tevreden ben met mijn project, maar besluit wel ook dat de details nog wat meer uitgewerkt kunnen.