Documentatie
GuitaristGear

Roel Kemp (500781)

Klas:
EHI1V.Sb

Vak:
Webapplicaties

Academisch Jaar:
2020 / 2021

Table of Contents

1. Introductie

“GuitaristGear” is een *full stack applicatie* waarmee informatie over gitaristen, hun apparatuur en de producenten van die apparatuur kan worden bijgehouden.

De applicatie is als *proof-of-concept* uitgewerkt. Een uitgebreidere versie die meer informatie opslaat zou bijvoorbeeld dienst kunnen doen als informatiesysteem voor een bedrijf dat de logistiek van muziektournees verzorgd.

In dit document wordt de functionaliteit van de applicatie omschreven en wordt er uitgelegd hoe de applicatie achter de schermen werkt, welke ontwerpkeuzen er zijn gemaakt en waarom deze keuzen zijn gemaakt. Tot slot wordt de volledige functionaliteit van de backend API omschreven in de REST specificatie.

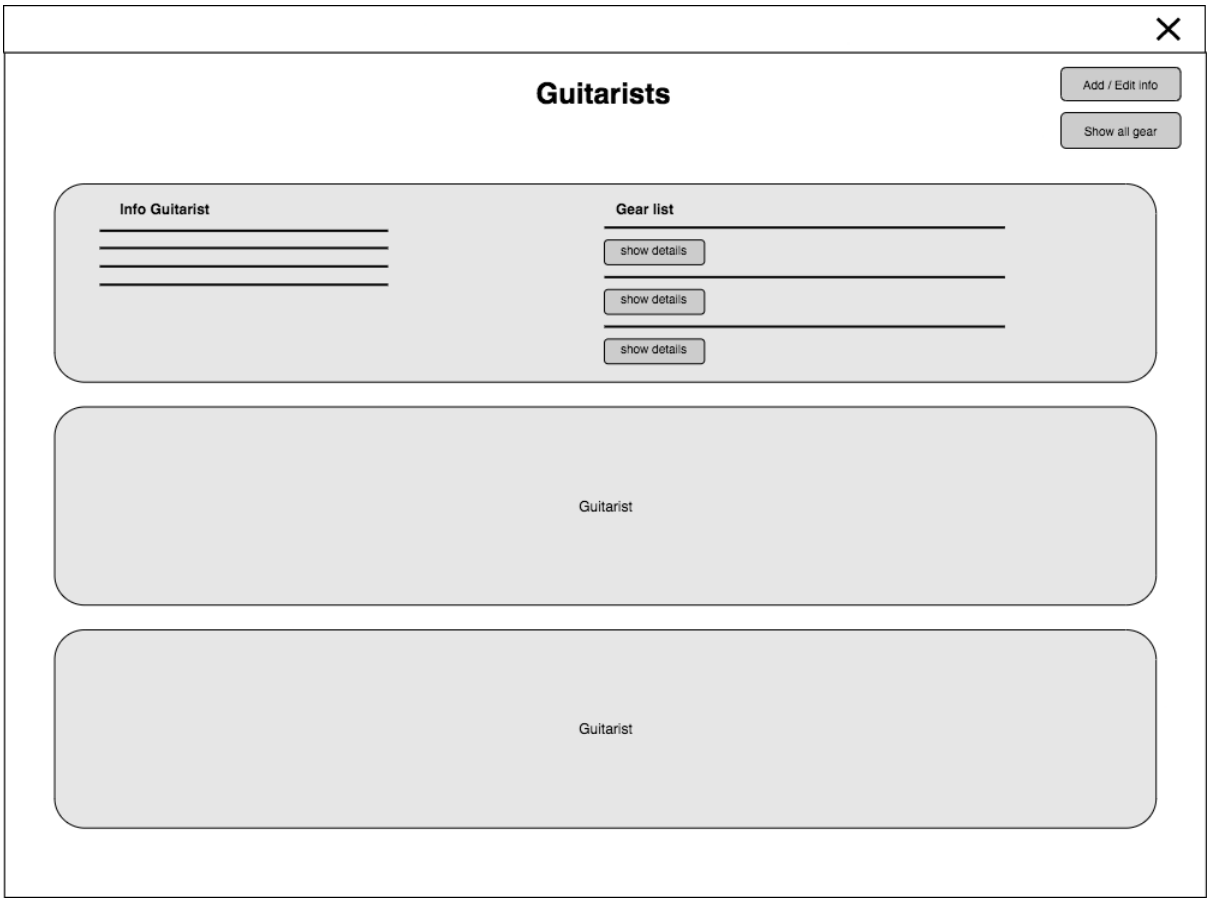
(LET OP! Om de webinterface van deze applicatie goed te laten werken is het belangrijk dat deze draait op een (lokale) webserver. Als de .html bestanden rechtstreeks vanaf de lokale opslag in een browser worden geopend, dan kunnen de in de database aanwezige gegevens niet worden gewijzigd!)

2. Functioneel ontwerp

Omschrijving functionaliteit door middel van wireframes

In de gehele frontend webinterface wordt gebruik gemaakt van een zwarte achtergrond. De menu's hebben allemaal een donkerblauwe achtergrond. Er is voor deze donkere achtergrond gekozen zodat de gebruiker uren achter elkaar met de webinterface kan werken zonder dat hij last krijgt van zijn ogen. Voor de duidelijkheid zijn de wireframes in lichte kleuren weergegeven.

Hoofdpagina



Figuur 1.1: Wireframe hoofdpagina

Op de hoofdpagina wordt er weergegeven welke gitaristen er in het systeem opgeslagen zijn. (Zie figuur 1.1.) Alle informatie over een gitarist kan in één oogopslag worden gezien. In de landschapweergave staat de achtergrondinformatie links en de lijst van apparatuur rechts. Als de webinterface in portretweergave (smaller dan 750 pixels) wordt weergegeven dan staat alle informatie onder elkaar.

Rechtsboven bevindt zich de navigatiebalk die ook zichtbaar blijft als er verder naar beneden wordt gescrold. Als er op de knop "Add / Edit info" wordt geklikt, verschijnt er een nieuwe pagina waarop informatie over gitaristen, apparatuur en producenten kan worden toegevoegd, gewijzigd of verwijderd.

Invoerpagina

Figuur 1.2: Invoerpagina

Op de invoerpagina kunnen nieuwe gitaristen, apparatuur en producenten worden toegevoegd, gewijzigd of verwijderd. (Zie figuur 1.2.) Om een nieuwe gitarist, apparatuur of producent toe te voegen zorgt u ervoor dat in het bovenste menu (Add / Edit) van het desbetreffende gedeelte de optie “Add new...” geselecteerd is. Dan vult u de tekstvelden daaronder in en klikt u op de corresponderende “Add” knop.

Let hierbij op dat u de juiste informatie in de tekstvelden invoert. Hieronder een overzicht van wat voor een soort invoer er in elk veld wordt verwacht:

Guitarist (Gitarist)

- name: Letters en spaties
- birth place: Letters, kommas en spaties
- birth year: Cijfers (gelijk aan, of groter dan 0)
- genre: Letters en spaties

Gear (Apparatuur)

- manufacturer: Kies uit de lijst van in het systeem aanwezige producenten.
- name: Letters, cijfers en spaties
- type: Letters en spaties
- weight in grams: Cijfers (gelijk aan, of groter dan 0)

Manufacturer (Producent)

- name: Letters en spaties
- main product type: Letters en spaties
- place founded: Letters, kommas en spaties
- year founded: Cijfers (gelijk aan, of groter dan 0)

Als een van de velden verkeerd is ingevuld wanneer er op de corresponderende “Add” knop wordt geklikt, dan kleurt deze rood. Wijzig dan de invoer zodat deze voldoet aan de hierboven gestelde eisen en klik nogmaals op “Add”. Als de informatie is geaccepteerd dan ververst het scherm en zijn alle velden leeg.

Om een gitarist, apparatuur of een producent te wijzigen kies u in het corresponderende “Add / Edit” menu welke gitarist, apparatuur of producent u wilt wijzigen. De reeds aanwezige informatie verschijnt dan in het de corresponderende tekstvelden. Na het aanpassen van de gegevens klikt u op “Edit”. Let er hierbij op dat de ingevulde tekstvelden wederom aan de hierboven gestelde eisen voldoen.

Om apparatuur aan de zgn.“gear list” van een gitarist toe te voegen of te verwijderen kiest u in het onderste gedeelte (“Edit gear list”) in het bovenste menu van welke gitarist u iets aan de “gear list” wilt toevoegen of verwijderen. Dan kiest u in het menu “Add to gear list” uit de lijst van alle apparatuur de gewenste apparatuur en klikt u op de knop “Add to gear list”.

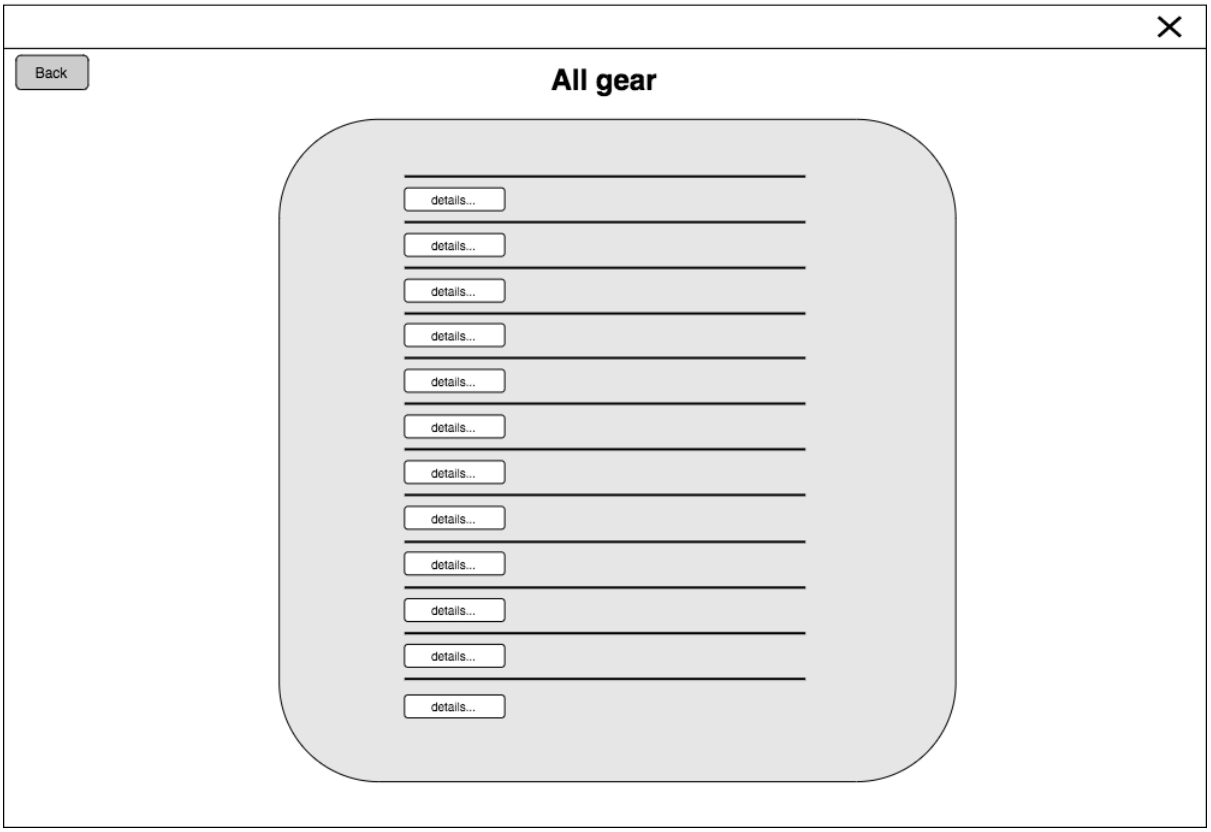
Na het succesvol toevoegen van apparatuur aan een “gear list” zal het scherm verversen.

Om apparatuur te verwijderen kiest u in het menu “Delete from gear list” uit de lijst van apparatuur die de geselecteerde gitarist bezit de apparatuur die u wilt verwijderen. Vervolgens klikt u op de knop “Delete from gear list”. Let er hierbij op dat appratuur die nog op de “gear list” van een gitarist staat en niet verwijderd kan worden. Ook producenten die nog geassocieerd zijn met apparatuur kunnen niet verwijderd worden. Om die producent te verwijderen moet er eerst voor gezorgd worden dat alle apparatuur die met de producent geassocieerd is gewijzigd of verwijderd is. (Zie ook: pagina 13)

Als de webinterface in portretweergave (smaller dan 750 pixels) wordt weergegeven dan staat alle informatie onder elkaar i.p.v. naast elkaar.

Als er op “back” geklikt wordt keert u terug naar de hoofdpagina.

Apparatuurpagina



Figuur 1.3: Wireframe apperatuurpagina

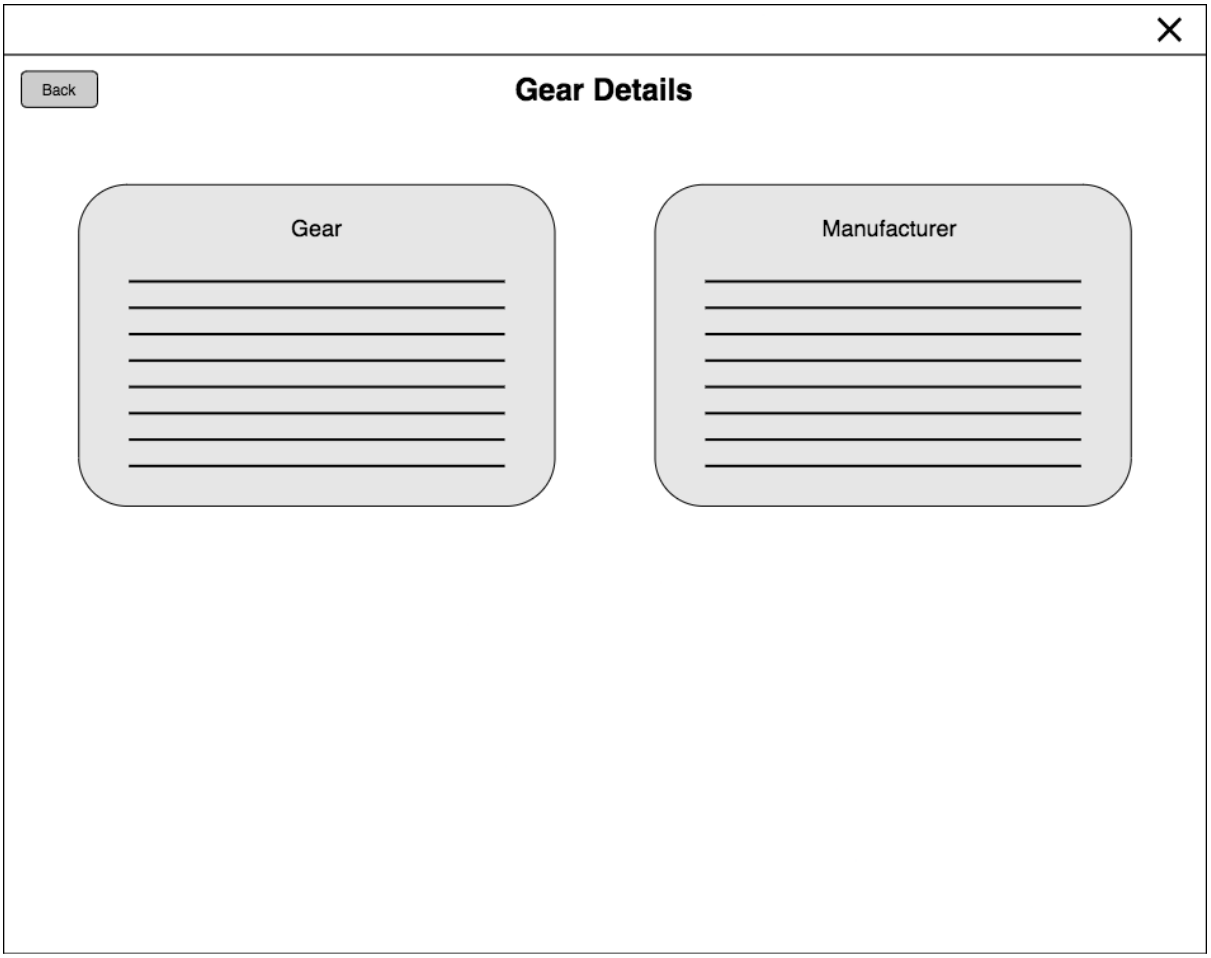
Als er vanuit de navigatieknoppen op de hoofdpagina voor de optie “Show all gear” wordt gekozen, dan wordt de apparatuurpagina getoond. (Zie figuur 1.3.) Op deze pagina wordt een lijst weergegeven van alle in het systeem aanwezige apparatuur.

Als er op de “details...” knop van een van de weergegeven stukken apparatuur wordt geklikt dan wordt de apparatuurdetailspagina van dit stuk apparatuur weergegeven. Op deze manier kan de informatie van alle in de database aanwezige apparatuur worden opgevraagd, ongeacht of deze in het bezit van een gitarist is of niet.

Als de webinterface in portretweergave (smaller dan 750 pixels) wordt weergegeven dan past de weergave zich aan om beter weergegeven te worden in portretmodus.

Als er op “back” geklikt wordt keert u terug naar de hoofdpagina.

Apperatuurdetailspagina



Figuur 1.4: Wireframe apparatuurdetailspagina

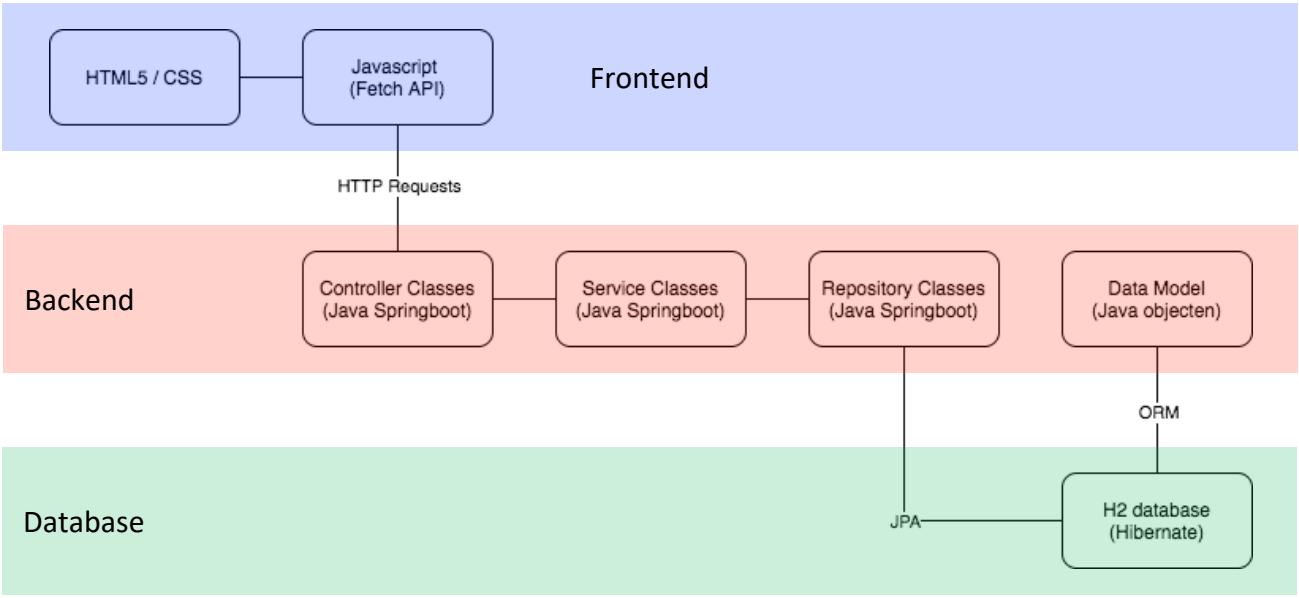
De apparatuurdetailspagina kan bereikt worden door op de hoofdpagina of op de apparatuurpagina bij een van de weergegeven stukken apparatuur op de knop “details..” te klikken. Op de apparatuurdetailspagina wordt de beschikbare informatie over de geselecteerde apparatuur en de producent van deze apparatuur weergegeven.

Als de webinterface in portretweergave (smaller dan 750 pixels) wordt weergegeven dan staat alle informatie onder elkaar.

Als er op “back” geklikt wordt keert u terug naar de pagina waar u vandaan kwam.

3. Technisch ontwerp

Omschrijving systeem



Figuur 3.1: Diagram om werking applicatie mee te verduidelijken.

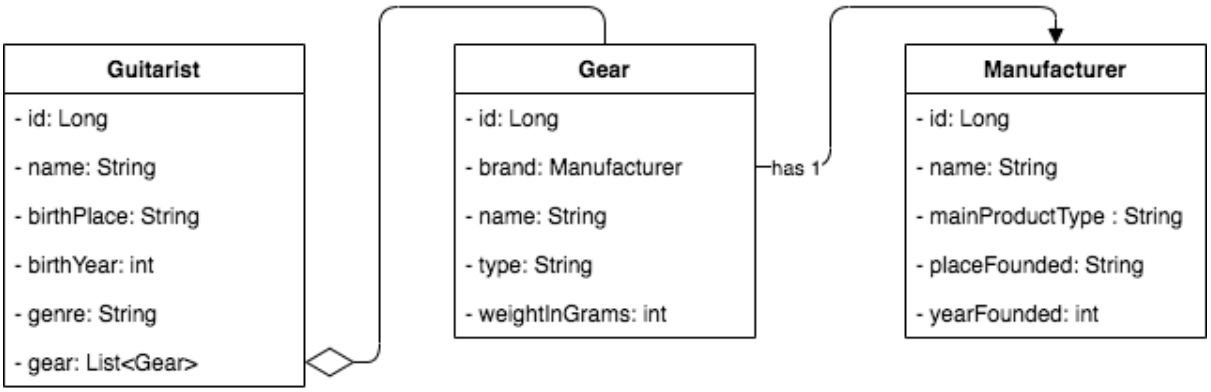
"GuitaristGear" bestaat uit 3 delen: Een frontend (geschreven in HTML, CSS en JavaScript), een backend (geschreven in Java, gebruikmakende van Spring Boot) en een database (Hibernate H2). In figuur 3.1 zijn deze 3 delen gerepresenteerd door de 3 verschillende lagen van het diagram. Zoals in figuur 3.1 te zien is communiceert de frontend met de backend en communiceert de backend met de database, welke opgebouwd wordt op basis van de POJOs (Plain Old Java Objects) in de backend applicatie.

In de backend heeft elk van de POJOs heeft een eigen Repository-, Service en Controller klasse. De Repository klasse wordt gebruikt om de database mee uit te kunnen lezen en te kunnen manipuleren. De Service klasse wordt gebruikt voor de het verwerken van de data tussen de Repository en de Controller klassen. In de Controller klasse worden middels zgn. mappings "API Request URLs" en "HTTP Verbs" (GET, POST, PUT, DELETE) toegewezen aan methoden, die de Controller klasse aan de Service klasse verbinden.

Met de frontend kunnen er dan HTTP Requests naar de backend gestuurd worden om te communiceren met de database.

Door het klassendiagram (Figuur 3.2) en de ERD (Entity Relationship Diagram) (Figuur 3.3) met elkaar te vergelijken wordt duidelijk hoe de POJOs en de door Springboot / Hibernate gegenereerde database aan elkaar gerelateerd zijn.

Klassendiagram



Figuur 3.2: Klassendiagram Datamodel GuitaristGear

Zoals weergegeven in het klassendiagram (figuur 3.2) heeft een gitarist (Guitarist) een naam, een geboorteplaats, een geboortejaar, een lijst van genres en een lijst van apparatuur (Gear). Deze lijst mag leeg zijn.

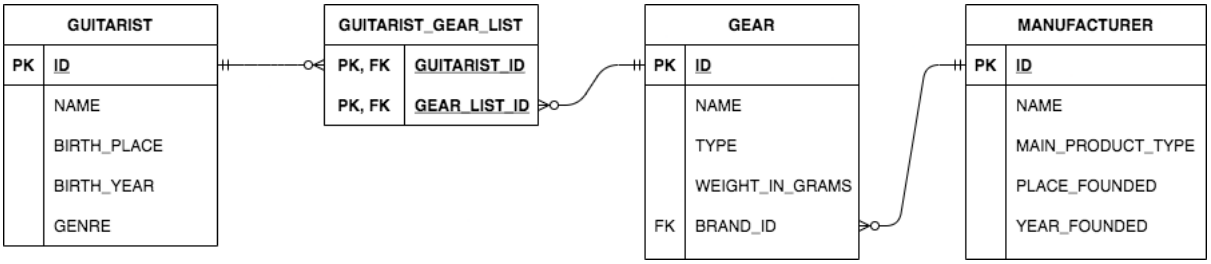
Omdat een gitarist een grote verscheidenheid aan apparatuur kan bezitten (versterkers, rack effecten, pedalen, gitaren, kabels, plectrums, koffers, accessoires enz.) is ervoor gekozen om geen subklassen van Gear te maken voor elk van deze soorten apparatuur. Om het simpel te houden wordt het type apparatuur in Gear daarom aangegeven met de String “type”. Verder wordt er in Gear opgeslagen: hoe de apparatuur heet, hoe zwaar de apparatuur is en wie de apparatuur heeft gemaakt.

Elk stuk apparatuur is namelijk vervaardigd door een producent (Manufacturer). Van een producent wordt opgeslagen wat de naam is van het bedrijf, welk soort apparatuur zij voornamelijk produceren, in welke plaats het bedrijf opgericht is en in welk jaar het bedrijf opgericht is.

De API zou uitgebreid kunnen worden door bij elk Guitarist object ook een lijst van Manufacturer objecten (sponsoren) op te slaan. Om de API simpel te houden is ervoor gekozen om deze relatie voor nu weg te laten.

Elk object heeft zijn eigen unieke id. Dit id wordt opgeslagen als Long object zodat deze ook leeg (*null*) kan zijn. Deze keuze is gemaakt omdat er met een database wordt gewerkt en het daarbij handig kan zijn om ontbrekende waarden weer te kunnen geven als *null.*

Entity Relationship Diagram (ERD)



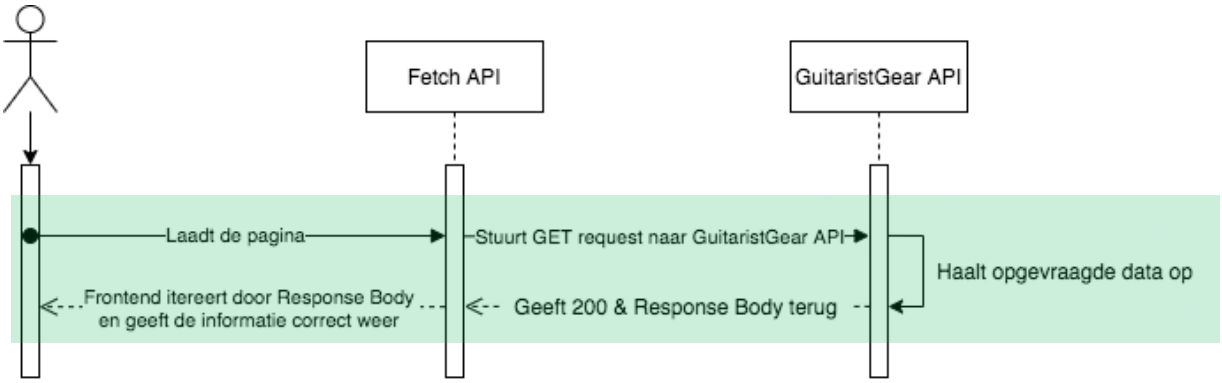
Figuur 3.3: ERD database GuitaristGear

Een gitarist (Guitarist) heeft nul of meer stuks apparatuur (Gear) in zijn bezit en een stuk apparatuur kan in bezit zijn van meerdere gitaristen. Dit is dus een meer-op-meer relatie, die zich in de database vertaalt tot een zgn. “join table” (GUITARIST_GEAR_LIST) waar de IDs van de gitaristen en de IDs van de apparatuur die zij bezitten, in zijn opgeslagen.

Een stuk apparatuur heeft één producent (Manufacturer), maar een producent kan verschillende apparatuur produceren. Tussen Gear en Manufacturer bestaat dus een meer-op-een relatie. In de database wordt dit bereikt door in GEAR het ID van de MANUFACTURER op te slaan als een zgn. “foreign key” (BRAND_ID).

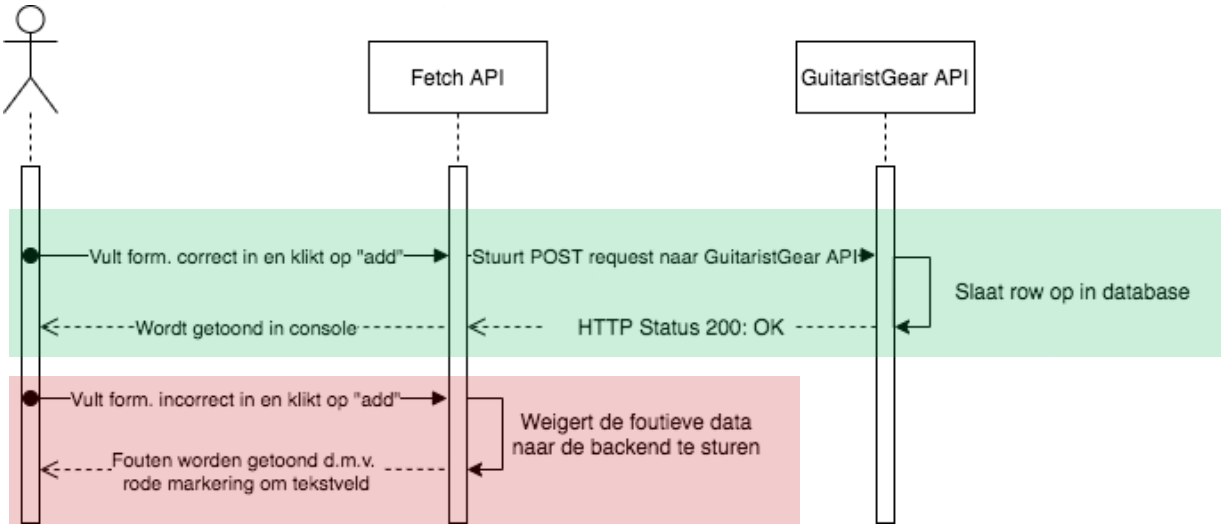
Sequence diagrammen

In de Sequence Diagrammen (figuur 3.4, 3.5, 3.6 en 3.7) staan verschillende scenario’s uitgebeeld om uit te leggen hoe de applicatie als geheel functioneert. Hieronder volgt een beschrijving van elk van de scenario’s.



Figuur 3.4: Sequence Diagram “Laden pagina”

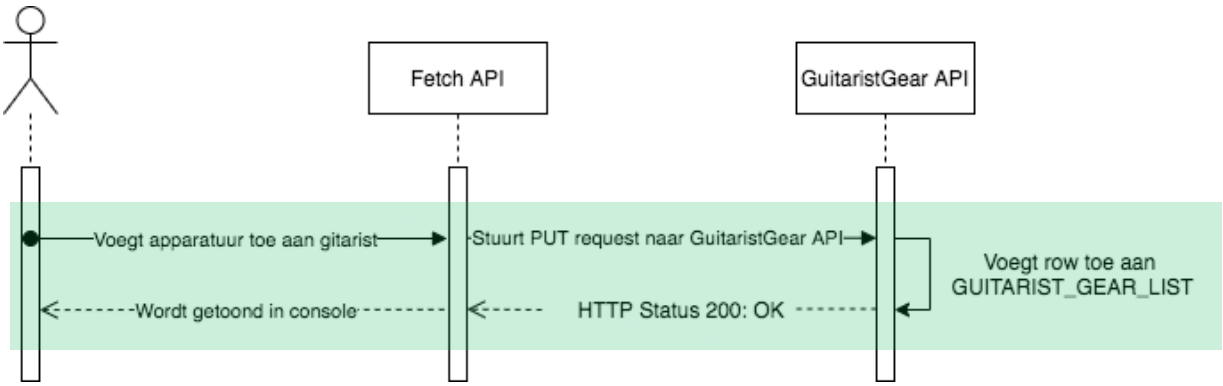
De gebruiker opent de pagina. Bij het laden van de pagina wordt met “onload” de methode makeGuitaristHTML() aangeroepen, welke een GET Request stuurt naar de GuitaristGear API. De API antwoordt met HTTP status 200 en geeft een Response Body in JSON formaat terug met daarin alle informatie over de gitaristen in de database. De frontend gebruikt dan een loop om van elk van de gitaristen alle informatie weer te geven. Per gitarist wordt er dan nog een genestelde loop gebruikt om alle apparatuur die zij bezitten weer te geven. De frontend geeft dan de informatie weer op een manier die geschikt is voor het formaat van het scherm dat de gebruiker gebruikt om de webinterface mee te bekijken. Omdat de gebruiker hieraan niets fout kan doen is er geen alternatief voor dit scenario afgebeeld.



Figuur 3.5: Sequence Diagram "Toevoegen data"

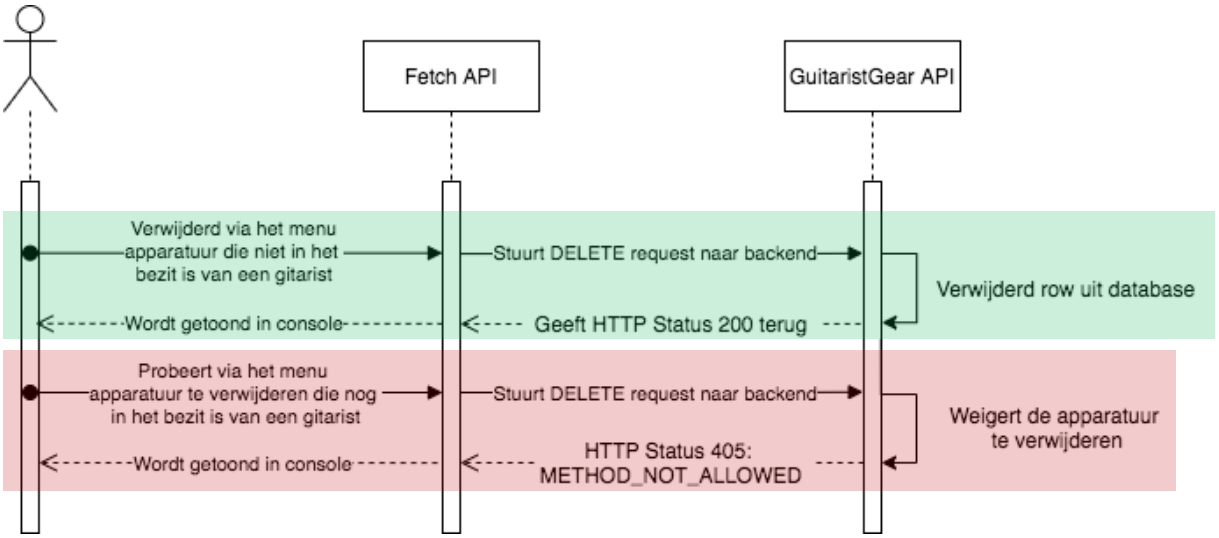
De gebruiker vult een van de formulieren in om een gitarist, apparatuur of een producent toe te voegen. Na het formulier correct ingevuld te hebben klikt de gebruiker op de knop "Add". De frontend stuurt dan een POST Request naar de GuitaristGear API die de aangeleverde informatie in een row in de database omzet. De GuitaristGear API geeft dan een HTTP status 200 terug die de gebruiker terug kan zien in de console.

In het geval dat de gebruiker een of meerdere velden verkeerd heeft ingevuld, wordt hij middels een rode markering hierop attent gemaakt en wordt de POST Request nooit naar de API verstuurd.



Figuur 3.6: Sequence Diagram "Apparatuur toevoegen aan gitarist"

De gebruiker kiest in het menu "Edit gear list" de gitarist aan wiens lijst van apparatuur hij iets toe wilt voegen. Dan kiest hij welke apparatuur hij aan de lijst toe wilt voegen en klikt hij op "Add to gear list". De Fetch API stuurt dan een PUT Request met aan het eind van de URL het id van de geselecteerde gitarist en in de message body het id van de apparatuur die toegevoegd moet worden. (Zie ook de API specificatie in het volgende hoofdstuk.) Dan voegt de backend een rij toe aan de tabel "GUITARIST_GEAR_LIST" in de database en wordt er een HTTP Status van 200 teruggestuurd, welke te zien is in de console. Omdat de frontend zo is ontworpen dat de gebruiker hier niets fout aan kan doen is er geen alternatief voor dit scenario afgebeeld.



Figuur 3.7: Sequence Diagram "Verwijderen apparatuur uit gear list"

De gebruiker kiest in het menu "Edit gear list" de gitarist van wiens lijst van apparatuur hij iets wilt verwijderen. Dan kiest hij welke apparatuur hij uit de lijst wilt verwijderen en klikt hij op "Delete from gear list". De Fetch API stuurt dan een DELETE Request met aan het eind van de URL het id van de geselecteerde gitarist en in de message body het id van de apparatuur die verwijderd moet worden. Dan verwijdert de backend de row uit de database tabel "GUITARIST_GEAR_LIST" die correspondeert aan de meegestuurde id waarden en wordt er een HTTP Status van 200 teruggestuurd, welke getoond in de console.

Als de gebruiker apparatuur probeert te verwijderen welke nog in het bezit van een gitarist is dan weigert de backend deze apparatuur te verwijderen (zie ook: pagina 6) en wordt er een HTTP code 405 teruggestuurd, welke wordt getoond in de console.

REST specificatie

Hieronder volgt de REST specificatie van de GuitaristGear API met daarin alle Requests die naar de GuitaristGear API gestuurd kunnen worden.

GET Requests

GET	**/gg/guitarist**		
Haalt een lijst van alle Guitarist objecten op.			
Parameters:	**Name**	**Type**	**Description**
** required*	genre	Query	Filter op genre.
	birthYear	Query	Filter op geboortejaar.
Responses:	**Code**	**Description / example if successful**	
	200	Lijst van alle gitarist objecten, eventueel gefilterd op genre en/of geboortejaar. Kan leeg zijn.	

GET	**/gg/guitarist/{id}**		
Haalt één Guitarist object op, gebaseerd op id.			
Parameters:	**Name**	**Type**	**Description**
** required*	id *	Path	Het id van het Guitarist object dat opgehaald moet worden.
Responses:	**Code**	**Description / example if successful**	
	200	Het gevonden Guitarist object, gebaseerd op id.	
	404	Wordt gegooid als er geen Guitarist object bestaat met de opgevraagde id.	

GET	**/gg/gear**		
Haalt een lijst van alle Gear objecten op.			
Parameters:	**Name**	**Type**	**Description**
** required*	brandName	Query	Filter op merknaam.
	type	Query	Filter op type.
Responses:	**Code**	**Description / example if successful**	
	200	Een lijst met alle Gear objecten. Eventueel gefilterd op merknaam en/of type. Kan leeg zijn.	

GET	/gg/gear/{id}		
Haalt één Gear object op, gebaseerd op id.			
Parameters:	**Name**	**Type**	**Description**
** required*	id *	Path	Het id van het Gear object dat opgehaald moet worden.
Responses:	**Code**	**Description / example if successful**	
	200	Het Gear object met de opgevraagde id.	
	404	Wordt gegooid als er geen Gear object bestaat met de opgevraagde id.	

GET	/gg/gear/guitarist/{id}		
Haalt een lijst met alle Gear objecten van één Guitarist object op.			
Parameters:	**Name**	**Type**	**Description**
** required*	id *	Path	Het id van het Guitarist object waarvan alle Gear objecten opgehaald moet worden.
Responses:	**Code**	**Description / example if successful**	
	200	Een lijst met alle Gear objecten van het opgevraagde Guitarist object.	
	404	Als het het id van het Guitarist object niet bestaat.	

GET	/gg/manufacturer		
Haalt een lijst met alle Manufacturer objecten op.			
Parameters:	**Name**	**Type**	**Description**
	mainProductType	Query	Filter op het type producten dat hoofdzakelijk wordt geproduceerd.
Responses:	**Code**	**Description / example if successful**	
	200	Een lijst van alle Manufacturer objecten. Eventueel gefilterd op type producten. Kan leeg zijn.	

GET	/gg/manufacturer/{id}		
Haalt één Manufacturer object op, gebaseerd op id.			
Parameters:	**Name**	**Type**	**Description**
** required*	id *	Path	Het id van het Manufacturer object.
Responses:	**Code**	**Description / example if successful**	
	200	Het Manufacturer object met de opgevraagde id.	
	404	Wordt gegooid als er geen Manufacturer object bestaat met de opgevraagde id.	

POST Requests

POST	/gg/guitarist		
Voegt een Guitarist object toe.			
Parameters:	**Name**	**Type**	**Description**
** required*	Guitarist *	Body	Het Guitarist object dat toegevoegd moet worden. Variabelen: name, birthPlace, birthYear, genre.
Responses:	**Code**	**Description / example if successful**	
	200	De response body is leeg.	
	400	Als er iets niet klopt aan de inhoud van de body.	

POST	/gg/gear		
Voegt een Gear object toe.			
Parameters:	**Name**	**Type**	**Description**
** required*	Gear *	Body	Het Gear object dat toegevoegd moet worden. Variabelen: name, type, weightInGrams.
Responses:	**Code**	**Description / example if successful**	
	200	De response body is leeg.	
	400	Als er iets niet klopt aan de inhoud van de body.	

POST	/gg/manufacturer		
Voegt een Manufacturer object toe.			
Parameters:	**Name**	**Type**	**Description**
** required*	Manufacturer *	Body	Het Manufacturer object dat toegevoegd moet worden. Variabelen: name, mainProductType, placeFounded, yearFounded.
Responses:	**Code**	**Description / example if successful**	
	200	De response body is leeg.	
	400	Als er iets niet klopt aan de inhoud van de body.	

PUT Requests

PUT	/gg/guitarist/{id}		
Overschrijft de waarden van een Guitarist object met de waarden in de Request body.			
Parameters:	**Name**	**Type**	**Description**
** required*	id *	Path	Het id van het te overschrijven Guitarist object.
	Guitarist *	Body	De nieuwe body van het Guitarist object. Variabelen: name, birthPlace, birthYear, genre. Lege variabelen overschrijven bestaande variabelen met null.
Responses:	**Code**	**Description / example if successful**	
	200	Het overschrijven was succesvol. In de response body staat de body van het overschreven Guitarist object zoals hij nu bestaat.	
	400	Als er iets niet klopt aan de inhoud van de body.	
	404	Wordt gegooid als er geen Guitarist object bestaat met de opgevraagde id.	

PUT	/gg/gear/{id}		
Overschrijft de waarden van een Gear object met de waarden in de Request body.			
Parameters:	**Name**	**Type**	**Description**
** required*	id *	Path	Het id van het te overschrijven Gear object.
	Gear *	Body	De nieuwe body van het Gear object. Variabelen: name, type, weightInGrams. Lege variabelen overschrijven bestaande variabelen met null.
Responses:	**Code**	**Description / example if successful**	
	200	Het overschrijven was succesvol. In de response body staat de body van het overschreven Gear object zoals hij nu bestaat.	
	400	Als er iets niet klopt aan de inhoud van de body.	
	404	Wordt gegooid als er geen Gear object bestaat met de opgevraagde id.	

PUT	/gg/manufacturer/{id}		
Overschrijft de waarden van een Manufacturer object met de waarden in de Request body.			
Parameters:	**Name**	**Type**	**Description**
** required*	id *	Path	Het id van het te overschrijven Manufacturer object.
	Manufacturer *	Body	De nieuwe body van het Manufacturer object. Variabelen: name, mainProductType, placeFounded, yearFounded. Lege variabelen overschrijven bestaande variabelen met null.
Responses:	**Code**	**Description / example if successful**	
	200	Het overschrijven was succesvol. In de response body staat de body van het overschreven Manufacturer object zoals hij nu bestaat.	
	400	Als er iets niet klopt aan de inhoud van de body.	
	404	Wordt gegooid als er geen Manufacturer object bestaat met de opgevraagde id.	

PUT	/gg/gear/guitarist/{guitaristId}		
Wijst een Gear object toe aan een Guitarist object.			
Parameters:	**Name**	**Type**	**Description**
** required*	id *	Path	Het id van het Guitarist object waar het Gear object aan toegewezen moet worden.
	Gear *	Body	De body met daarin het id van het toe te wijzen Gear object. Alle andere variabelen worden genegeerd.
Responses:	**Code**	**Description / example if successful**	
	200	Het toewijzen was succesvol.	
	400	Als er iets niet klopt aan de inhoud van de body.	
	404	Wordt gegooid als er geen Gear object bestaat met de opgevraagde id.	

PUT	/gg/gear/manufacturer/{brandId}		
Wijst een Gear object toe aan een Manufacturer object.			
Parameters:	**Name**	**Type**	**Description**
** required*	id *	Path	Het id van het Manufacturer object waar het Gear object aan toegewezen moet worden.
	Gear *	Body	De body met daarin het id van het toe te wijzen Gear object. Alle andere variabelen worden genegeerd.
Responses:	**Code**	**Description / example if successful**	
	200	Het toewijzen was succesvol.	
	400	Als er iets niet klopt aan de inhoud van de body.	
	404	Wordt gegooid als er geen Gear object bestaat met de opgevraagde id.	

DELETE Requests

DELETE	**/gg/guitarist/{id}**		
Verwijderd een Guitarist object op basis van id.			
Parameters:	**Name**	**Type**	**Description**
** required*	id *	path	Het id van het Guitarist object dat verwijderd moet worden.
Responses:	**Code**	**Description / example if successful**	
	200	Guitarist object succesvol verwijderd.	
	404	Guitarist object niet gevonden.	

DELETE	**/gg/gear/{id}**		
Verwijderd een Gear object op basis van id.			
Parameters:	**Name**	**Type**	**Description**
** required*	id *	path	Het id van het Gear object dat verwijderd moet worden.
Responses:	**Code**	**Description / example if successful**	
	200	Gear object succesvol verwijderd.	
	404	Gear object niet gevonden.	
	405	Gear object is nog geassocieerd met een gitarist.	

DELETE	**/gg/gear/guitarist/{guitaristId}**		
Verwijderd een Gear object op basis van id, van een Guitarist object op basis een id.			
Parameters:	**Name**	**Type**	**Description**
** required*	guitaristId *	path	Het id van het Gear object dat verwijderd moet worden.
	Gear *	body	De body met daarin het id van het te verwijderen Gear object. Alle andere variabelen worden genegeerd.
Responses:	**Code**	**Description / example if successful**	
	200	Gear object succesvol verwijderd.	
	404	Gear object niet gevonden.	

DELETE	**/gg/manufacturer/{id}**		
Verwijderd een Manufacturer object op basis van id.			
Parameters:	**Name**	**Type**	**Description**
** required*	id *	path	Het id van het Manufacturer object dat verwijderd moet worden.
Responses:	**Code**	**Description / example if successful**	
	200	Manufacturer object succesvol verwijderd.	
	404	Manufacturer object niet gevonden.	
	405	Manufacturer object nog gekoppeld met Gear object.	