

Inhoudsopgave

Team ConCATenate - Wie zijn wij.	2
Beschrijving van het project.	4
Scope van het project.	6
Kostprijsberekening - Nacalculatie	7
Purrfect Design	9
Mockups	10
Gantt overzicht van het project.	15
Database-structuur	16
Custom Code	20
Tech Stack van het project.	27
Calamiteiten tijdens het project	29
Teamwork! Wie deed wat.	30

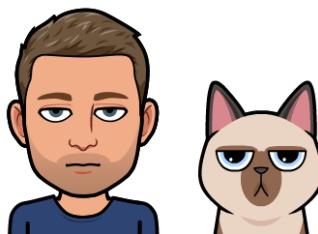
Team ConCATenate - Wie zijn wij



Sigurd Boels

Nadat ik een bachelor in Sociaal Werk behaalde, en gewerkt te hebben bij verschillende bedrijven besloot ik dat het tijd werd om een nieuwe weg in te slaan en een opleiding te volgen voor een job waar ik mezelf echt voor kon motiveren.

Hierdoor kwam ik terecht bij de opleiding Full Stack Developer aan SyntraPXL waar ik mijn (quasi) eerste stappen in de wereld van het programmeren zet.



Jari Boeckstaens

Na een aantal jaar informatica aan de UHasselt te hebben gestudeerd maar dit niet te hebben afgerond, heb ik dit jaar besloten om aan de opleiding van Full-Stack Developer deel te nemen. Dit sluit aan bij mijn voorkennis uit de vorige opleiding, wat mij dan ook een goede basis geeft om als developer deel te nemen aan de arbeidsmarkt.



Liesbeth Poelmans

Na een degelijke ervaring als Bid-Pricing Manager bij een Global Internet Service Provider besloot ik mijn carrière een nieuwe wending te geven en de opleiding Full Stack Developer te volgen aan Syntra PXL.

Gefascineerd door processen en hoe digitalisatie hierin een verschil kan maken, tast ik af op welke manier ik binnen de webDev wereld een verschil kan maken.

In mijn vrije tijd ben ik pleegmoeder voor een asiel en ervaar ik regelmatig hoe chaotisch deze samenwerking verloopt. Wat meteen de aanleiding was om dit als thema voor te stellen voor onze app.

Beschrijving van het project

Naam

AutoCat

Doelgroep

Dierenasiel

Alle kattenasielen die samenwerken met pleeggezinnen en structuur willen brengen in deze samenwerking.

Pleeggezinnen

Pleeggezinnen die kwetsbare katten opvangen die niet terecht kunnen in de algemene opvang van dierenasielen.

Probleemstelling

Momenteel gebeurt de coördinatie tussen het kattenasiel en de pleeggezinnen voornamelijk via berichten, Whatsapp, Messenger en email. M.a.w. alle informatie over de katten die bij een pleeggezin verblijven zit verspreid over deze kanalen.

Dit leidt tot een chaotische en tijdrovende manier van werken. De tijd die hiermee verloren raakt kan beter gebruikt worden om de zwerfkattenpopulatie onder controle te houden door trap-neuter-return acties uit te voeren om zo de zwerfkattenproblematiek op lange termijn aan te pakken.

App

Onze App brengt het volledige proces vanaf de aanmelding van een nieuwe kat in het asiel tot het adopteren van deze kat in kaart.

De gegevens van de katten en de pleeggezinnen worden via hun respectievelijke pagina's in de database van het asiel opgenomen. Deze informatie kan makkelijk geraadpleegd of geüpdateerd worden, waardoor het stellen van vragen aan het pleeggezin overbodig wordt.

Dankzij het algoritme achter het asiel-dashboard wordt een ideale match gemaakt tussen de kat en het pleeggezin waar deze gaat verblijven tot er een gouden mandje gevonden wordt. Via meldingen tussen het asiel-dashboard en dat van de pleegouder worden afspraken gemaakt en informatie uitgewisseld.

Zolang de kat bij het pleeggezin verblijft worden alle gegevens over het karakter, medische geschiedenis etc. door het pleeggezin bijgehouden en gedeeld met het asiel.

Missie

Onze applicatie richt zich tot de beheerders van kattenasielen en de pleeggezinnen die met deze asielen samenwerken.

Onze belangrijkste missie is het leven van de asielmedewerkers makkelijker maken zodat zij de nodige tijd kunnen vrijwaren voor de prioritaire taken binnen het asiel. We willen daarnaast ook de pleeggezinnen een gestructureerd platform aanbieden voor hun samenwerking met het asiel.

En last but not least zorgt onze app ervoor dat katten sneller en efficiënter hun ideale gouden mandje vinden en er meer katten opgevangen en dus ook gesteriliseerd kunnen worden.

De app heeft talloze uitbreidingsmogelijkheden:

- Documenteren van uitleenmateriaal (benches, kattenbakken, draagmanden,...)
- Populeren van verplichte overheidsformulieren.
- Documenteren van financiën (pleeggezinnen krijgen budgetten voor voeding,...)
- Documenteren van trap-neuter-return acties.
- Publiek luik voor potentiële adoptanten.
- Peter/Meter functionaliteit (fondsenwerving)
- ...

Dit alles bewerksteltig een gestroomlijnde werking van het asiel maar kan ook gebruikt worden om subsidieaanvragen te ondersteunen en financiën te optimaliseren.

Scope van het project

Algemeen

Must Have:

- Dynamische navbar & sidebar (incl. logout)
- Footer: Gegevens asiel, nuttige links & gegevens team ConCATenate

Login

Must Have:

- E-mail & Wachtwoord
- Aanmeldknop (redirect afhankelijk van profiel)
- Maak een account aan als asielbeheerder of als Pleeggezin

Could Have:

- Wachtwoord vergeten (incl mail)
- Onthoud mij

Registratie/Account VZW

Must Have:

- Algemene gegevens & wachtwoord
- Create, View, Update & Hashing

Could Have:

- Profelfoto

Registratie/Account Pleeggezin

Must Have:

- Algemene gegevens & wachtwoord
- Voorkeuren pleegzorg
- Geen toegang tot de overzichtspagina's & gegevens van andere pleeggezinnen
- Create, View, Update & Hashing

Could Have:

- Profelfoto
- Huisgenoten
- Huisdieren

Kat detailpagina

Must Have:

- Adoptie status & Pleeggezin
- Intake gegevens
- Informatie karakter / Adoptiemogelijkheden
- Wegeningen
- Dierenartsbezoeken
- Create, View & Update

Could Have:

- Profielfoto
- Foto album

VZW dashboard

Must Have:

- *Berichten:*
 - Weergave ontvangen berichten (incl delete)
 - Versturen van nieuwe berichten (dynamische dropdowns)
- *Match Maker:*
 - Dynamische selectie mogelijkheden:
Bij selectie van een kat kunnen enkel pleeggezinnen geselecteerd worden waarbij de voorkeuren overeenkomen met de eigenschappen van de kat en visa versa
 - Weergave beperkte informatie
 - Link naar detailpagina's

Pleeggezin dashboard

Must Have:

- *Berichten:*
 - Weergave ontvangen berichten (incl delete)
 - Versturen van nieuwe berichten (dynamische dropdowns)

Katten overzicht

Must Have:

- Cards met beperkte info (incl link naar detailpagina & pleeggezin)
- Filters
- Zoeken op naam

Pleeggezinnen overzicht

Must Have:

- Cards met beperkte info (incl link naar pleeggezin)
- Filters
- Zoeken op naam

Purrfect Design

Huisstijl

Bootstrap Template

We maken gebruik van een bootstrap template genaamd Purple. Deze template beschikt over frisse en speelse kleuren. De manier waarop we deze hebben toegepast kan je vinden in onderstaand overzicht.

Font

"Ubuntu" is de basis font-family en "Roboto" is de font-family voor de titels en de labels.

Kleuren

Door met accentkleuren te spelen geven we bepaalde informatie een categorie.

Zo zie je bijvoorbeeld in het dashboard van het asiel zeer duidelijk het verschil tussen de gegevens van de kat en het pleeggezin.

	#f5f5f5	rgb(245, 245, 245)	Achtergrondkleur
	#9de3cf	rgb(157, 227, 207)	Asiel Accentkleur
	#fe7c96	rgb(254, 124, 150)	Kat Accentkleur
	#198ae3	rgb(25, 138, 227)	Pleeggezin Accentkleur

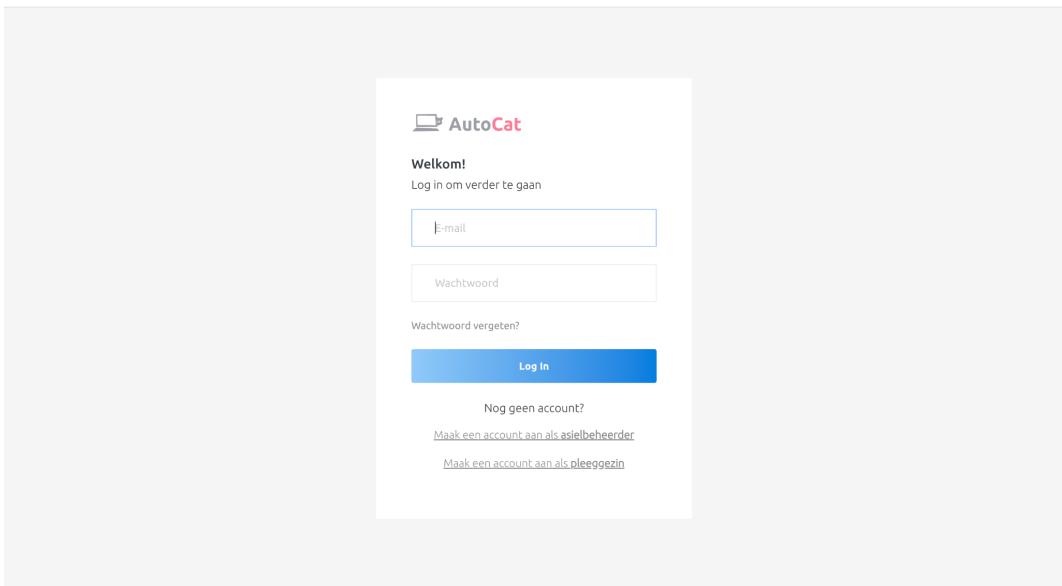
Logo

Het logo werd gemaakt door Emma Claessens. Dat er katten achter onze app zitten is overduidelijk maar we verwijzen hier ook naar die lieve poes die altijd bij jou zou willen zijn en wacht om geadopteerd te worden.

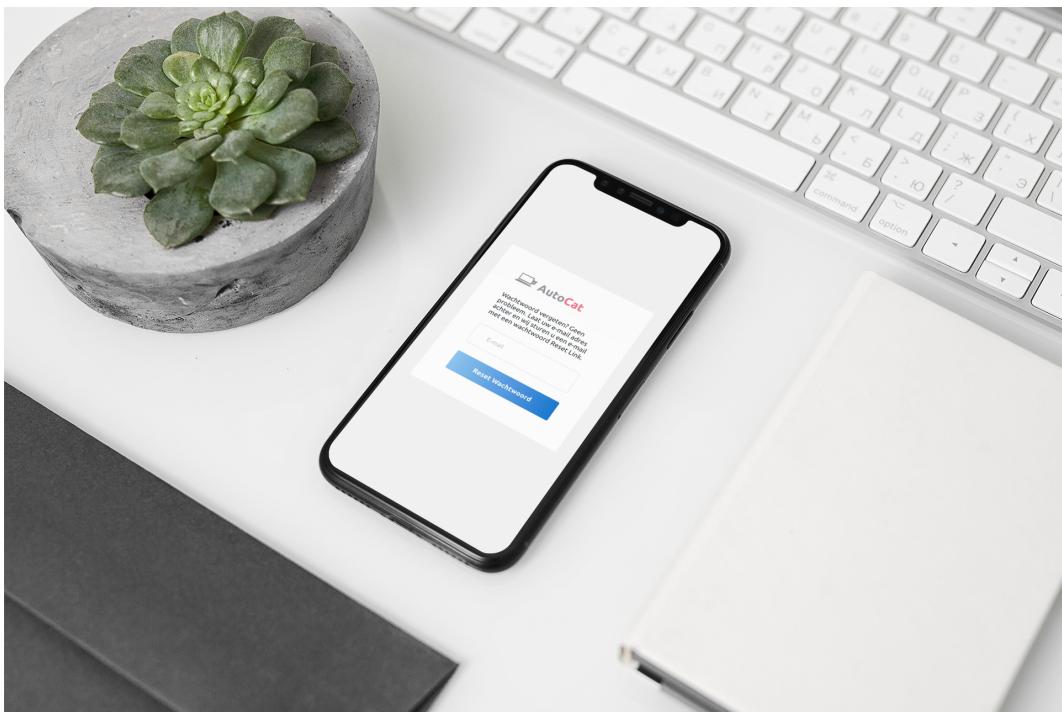


Mockups

Login



Wachtwoord Vergeten



Registratie VZW

AutoCat

[Registreer als VZW](#) [Terug naar login](#)

Mijn gegevens

Naam asiel	Telefoonnummer asiel	HK-nummer
Naam	Voornaam	Geboortedatum dd/mm/yyyy <input type="button" value=""/>
Straat	Huisnummer	Postcode
Telefoonnummer	E-mail	Wachtwoord
Website <input type="text"/>		

Registratie Pleeggezin

AutoCat

[Registreer als pleeggezin](#) [Terug naar login](#)

Mijn gegevens

Naam	Voornaam	Geboortedatum dd/mm/yyyy <input type="button" value=""/>
Straat	Huisnummer	Postcode
Telefoonnummer	E-mail	Wachtwoord

Ik sta open voor

Welkom

The screenshot shows the homepage of the AutoCat platform. At the top left is the logo "AutoCat". On the right, there is a user profile for "Massimo Denittis" with a dropdown arrow and a help icon. The main header reads "DAG MASSIMO" and "Welkom bij AutoCat". Below the header is a large image of a light-colored kitten lying on a carpet. To the left of the image is a sidebar with navigation links: "Dashboard", "Kat Aanmelden", "Katten Overzicht", and "Pleeggezinnen Overzicht". The central content area contains a list of five tips:

- Communiceer via het dashboard met de pleeggezinnen
- Vind met de Matchmaker snel het ideale pleeggezin voor jouw opvangertjes
- Meld makkelijk een nieuw opvangertje aan
- Raadpleeg alle informatie van de opvangertjes via de overzichtspagina
- Navigeer tussen de pleeggezinnen via de overzichtspagina

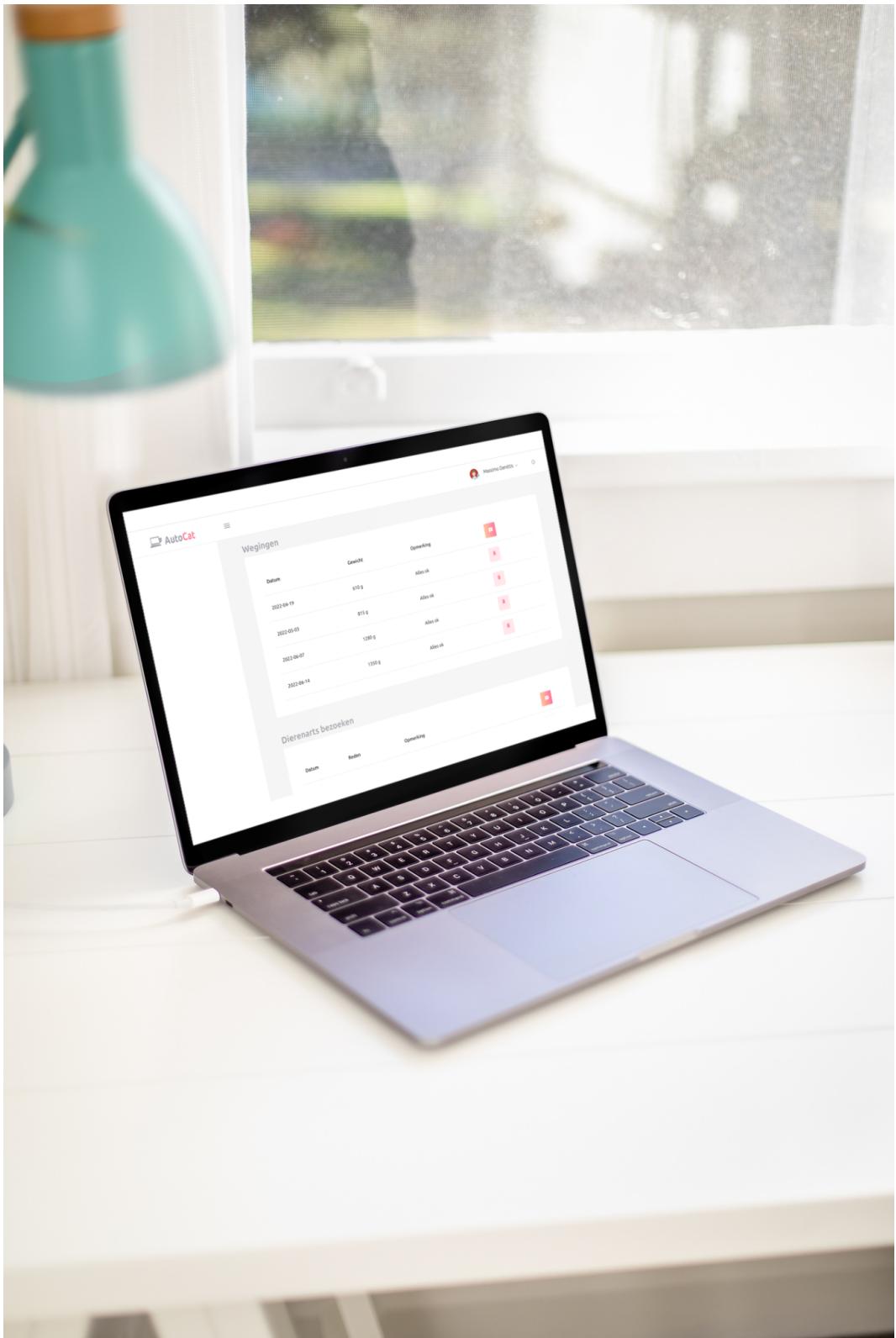
Pleeggezin Account

The screenshot displays the "Huisgenoten" (Housemates) section of the Pleeggezin Account. It shows two tables of household members. The first table lists a "Partner" born on 1975-11-30 and a "Kind" born on 2019-06-09. The second table lists another "Partner" born on 2018-09-12 and a "Kind" born on 2021-06-20. Below these tables is a form titled "Ik sta open voor" (I am open for) with fields for "Aantal beschikbare opvangertjes" (Number of available placement children) and checkboxes for "Vader", "Moeder", "Zoon", "Daughter", "Kitten", "Welp", "Raadplegen", "Zie niet interessante voorbeeld", "Zie andere interessante vering", and "Informatie". At the bottom of the page are links for "Nuttige Links" (Useful Links) and "Team ConCATAteam".

Kat Pagina

The screenshot shows the 'Kat Detail' section of the AutoCat application. At the top, there are navigation links: 'Dashboard', 'Mijn Katten', 'Kat Aanmelden', and a user profile for 'Liesbeth Poelmans'. Below the navigation is a title 'Meld nieuwe kat aan' (Report new cat). The form fields include 'Adoptiestatus' (Selected 'Aangemeld') and 'Pleeggezin' (Selected 'Selecteer'). The main input area is titled 'Intake gegevens' (Intake data) and contains fields for 'Naam' (Name), 'Naam aanmelder' (Name of applicant), 'Telefoonnummer aanmelder' (Phone number of applicant), 'Ras' (Breed), 'Kleur' (Color), and 'Vachtlengte' (Fur length). It also includes date fields for 'Geboorte datum' (Birth date) and 'Geslacht' (Gender), and dropdowns for 'Socialisatie' (Socialization) and 'Selecteer'.

The screenshot shows the 'Informatie karakter' (Character information) section of the AutoCat application. At the top, there is a title 'Beschrijving' (Description) with the text 'Enorm speels en iets te wild met zijn zusjes'. Below this is a section titled 'Adoptie mogelijkheden' (Adoption opportunities) with three radio buttons for 'Solo Plaatsing': 'Moet' (Must), 'Mag' (May), and 'Nee' (No). To the right, under 'Kan geplaatst worden met' (Can be placed with), there are checked boxes for 'Jonge kinderen' (Young children), 'Katten' (Cats), and 'Wil in de slaapkamer' (Will in the bedroom). There are also sections for 'Bij ander huisdier' (With another pet) and 'Toegang tot tuin' (Access to garden), each with three radio button options. On the right side, there is a 'Karakter' (Character) section with checked boxes for 'Speelse kat' (Playful cat) and 'Wil in de slaapkamer' (Will in the bedroom).



Dashboard voor VZW

The screenshot shows the 'Dashboard' page of the AutoCat application. At the top, there's a navigation bar with the AutoCat logo, a menu icon, and a user profile for 'Massimo Denittis'. Below the header, a sidebar on the left lists 'Dashboard', 'Kat Aanmelden', 'Katten Overzicht', and 'Pleeggezinnen Overzicht'. The main content area is titled 'Meldingen' (Messages) and displays a table with the following data:

Pleeggezin	Kat	Melding Type	Bericht	Aanpassen
Liesbeth Poelmans	Cara	Adoptant goedgekeurd	Zodra Cara en Nala gesteriliseerd zijn kunnen ze naar Nora vertrekken	[button]
Liesbeth Poelmans	Nala	Adoptant goedgekeurd	Zodra Cara en Nala gesteriliseerd zijn kunnen ze naar Nora vertrekken	[button]
Liesbeth Poelmans	Elsa	Dierenarts bezoek	Nieuw medicijn voor de epilepsie. De aanvallen verhogen in frequentie	[button]
Leentje Bout	Tijgertje	Dierenarts bezoek	Sterilisatie goed verlopen	[button]
Sigrid Medats	Felix	Adoptant afgeweerd	Bart is te weinig thuis om te kunnen omgaan met de ziekte van Felix	[button]

The screenshot shows the 'Match Maker' feature. It compares two profiles: 'Basje' (the cat) and 'Heleen Vijgen' (the owner). Each profile has three sections: 'Algemene Informatie' (General Information), 'Aandachtspunten' (Attention Points), and 'Eigenschappen' (Properties).

Basje (Cat Profile):

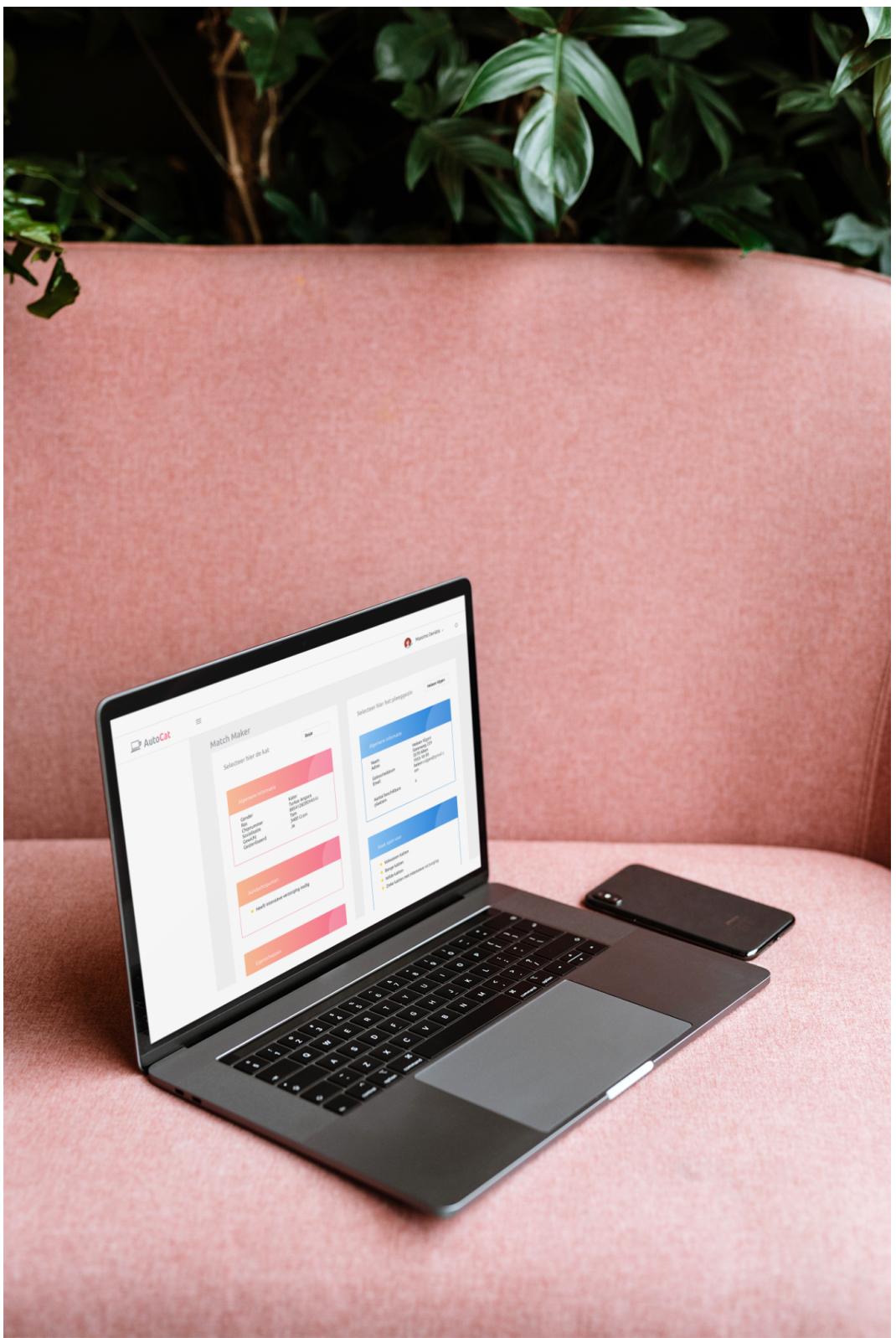
- Gender: Kater
- Ras: Turke Angora
- Chipnummer: 885412828094046
- Socialisatie: Tam
- Gewicht: 3400 Gram
- Gesteriliseerd: Ja
- Kater

Heleen Vijgen (Owner Profile):

- Naam: Heleen Vijgen
- Adres: Steenweg 209
3570 Alken
- Geboortedatum: 1955-10-30
- Email: heleen.vijgen@gmail.com
- Aantal beschikbare plaatsen: 4

Common Fields:

- Staat open voor:
 - Volwassen katten
 - Bange katten
 - Wilde katten
 - Zieke katten met intensieve verzorging



Katten Overzicht voor VZW

The screenshot shows the 'Katten Overzicht' (Cat Overview) page. At the top, there is a search bar labeled 'Zoek op naam'. Below it, there are four dropdown filters: 'Selecteer pleeggezin...', 'Selecteer Geslacht...', 'Selecteer Leeftijd...', 'Selecteer Adoptiestatus...', 'Selecteer Adoptie mogelijkheden...', and 'Selecteer Karakter...'. The main area displays six cat cards:

Name	Age	Status
Tux	2 maanden	Bij Pleeggezin <u>Liesbeth Poelmans</u> Meer info
Luna	2 maanden	Bij Pleeggezin <u>Liesbeth Poelmans</u> Meer info
Cara	2 maanden	In opzione <u>Liesbeth Poelmans</u> Meer info
Nala	2 maanden	
Charlie	12 jaar	
Felix	2 jaar	

Pleeggezinnen Overzicht voor VZW

The screenshot shows the 'Pleeggezinnen Overzicht' (Foster Parent Overview) page. At the top, there is a search bar labeled 'Zoek op naam'. Below it, there are three dropdown filters: 'Selecteer een aantal plaatsen...', 'Selecteer katvoorker(en)...', and 'Selecteer thuisituatie...'. The main area displays six foster parent cards:

Name	Available Places	Description
Dorien Bex	6 beschikbare plaatsen	Meer info
Sophie Coemans	4 beschikbare plaatsen	Meer info
Leentje Bout	-1 beschikbare plaats	Meer info
Jari Boeckstaens	3 beschikbare plaatsen	Meer info
Jan Van den Berg	10 beschikbare plaatsen	Meer info
Sara Frère	2 beschikbare plaatsen	Meer info

Dashboard voor Pleeggezin

The screenshot shows the 'Dashboard voor Pleeggezin' interface. A modal window titled 'Nieuwe melding' (New Report) is open in the center. It contains dropdown menus for 'Kat' (Cat) and 'Melding Type' (Report Type), and a text area for 'Bericht' (Message). Below the message area are two buttons: 'Sluit' (Close) and 'Verstuur' (Send). In the background, there's a list of reports under the heading 'Meldingen'. One report for 'Tux' is visible, with the message 'Afspraak adoptant maken' (Make appointment with adopter) and the recipient 'Contacteer Robby Robben (0468 26 75 29)'. There are also other reports for 'Luna', 'Elsa', and 'Tux'.

Mijn Katten voor Pleeggezin

The screenshot shows the 'Mijn Katten voor Pleeggezin' page. On the left, there's a sidebar with navigation links: 'Dashboard', 'Mijn Katten' (which is currently selected and highlighted in orange), and 'Kat Aanmelden'. The main content area has a title 'Ga naar Detailpagina' (Go to Detail Page). Below it, there's a grid of five cards, each representing a cat:

- Tux** (Kater, Tam, Bij Pleeggezin, 2 maanden) - Status: 2 maanden
- Luna** (Kattin, Tam, Bij Pleeggezin, 2 maanden) - Status: 2 maanden
- Cara** (Kattin, Tam, In optie, 2 maanden) - Status: 2 maanden
- Nala** (Kattin, Tam, In optie, 2 maanden) - Status: 2 maanden
- Elsa** (Kattin, Bang, Bij Pleeggezin, 0 week) - Status: 0 week

Each card includes a 'Meer info' (More info) button at the bottom.

Tech Stack van het project

Wij bij ConCATenate hebben zoals eerder in de paper reeds vermeld een webapplicatie ontwikkeld. Zoals verwacht maken we hiervoor gebruik van HTML, CSS, JavaScript en PHP, dit is echter niet het enige wat we gebruikt hebben qua technologieën binnen de applicatie. Om dus wat meer duiding te geven bij de gebruikte talen en frameworks, worden deze hier opgeliist en kort uitgelegd.

Programmeertalen

Hier bespreken we de gebruikte programmeertalen

HTML 5

Voor het opbouwen van de webpagina's maken we gebruik van HyperText Markup Language of kortweg, HTML. Omdat we een applicatie bouwen die zo goed en zo lang mogelijk ondersteund wordt, maken we gebruik van de meest recente versie van HTML, namelijk HTML5. In deze versie van HTML zitten reeds een aantal elementen ingebouwd die in de vorige versies niet voorzien waren.

CSS

Om onze pagina's hun mooie look & feel te geven, gebruiken we Cascading Style Sheets (css). Op dit moment is de meest actuele implementatie van css: css3, deze gebruiken we dan ook logischerwijs in het project. Wel moet hier opgemerkt worden dat er binnen het project extensief gebruik gemaakt wordt Bootstrap, meer hierover bij de frameworks.

Javascript/ Ecmascript

Om ons project dynamisch te maken, gebruiken we EcmaScript 6 (ES6). Dit is de tweede grote versie van Javascript. Dit maakt het leven van een developer ietwat eenvoudiger aangezien er tijdens het ontwikkelen steeds gezien kan worden wat het type is dat verwacht wordt.

PHP

Voor het back-end eindwerk, maken we logischerwijs hoofdzakelijk gebruik van een server side-taal. Hier maken we gebruik van PHP 8 (PHP: Hypertext Preprocessor), dit in de vorm van het Laravel framework met bijhorende Blade Templates. Meer hierover onder het deel Frameworks. Het voordeel aan hoofdzakelijk PHP te gebruiken om data in te laden in een pagina, is dat de gebruiker geen weet heeft van de interne logica van de applicatie. Op deze manier kunnen we onze gegevens beter beschermen dan dat we dit in een zuiver front-end applicatie zouden kunnen doen.

SQL

Om een query uit te voeren op onze relationele database, stellen we deze op in sql ofwel Structured Query Language. Binnen de applicatie maken we gebruik van de Mysql, voor de structuur van onze database verwijzen we u graag naar het hoofdstuk Database-structuur.

Frameworks

Hier bespreken we de gebruikte frameworks

Laravel

Onze applicatie is gebouwd met het Laravel framework, hierin hebben we onze modellen (samen met onze migrations voor de database) opgesteld. Vervolgens maken we van de routes gebruik om functies in de controllers uit te voeren die onze database aanpassen. Deze database aanpassingen voeren we uit door functies aan te roepen van de Eloquent ORM. Deze Object Relational Mapping, is een techniek waarbij complexere queries in een korte begrijpbare functie worden ondergebracht. Dit zijn reeds twee van de componenten van het MVC patroon, voor de views maken we gebruik van de Blade templates. In deze templates kunnen we, met behulp van een specifieke syntax: {}, @foreach, @if, ... , op een zeer eenvoudige manier onze data inladen in de pagina's.

Laravel Breeze

Om het authenticatie proces af te handelen binnen onze applicatie, maken we gebruik van Laravel Breeze. Dit verzekert ons een veilige manier voor onze gebruikers om zich te kunnen registreren en in te loggen.

Bootstrap

Zoals reeds vermeld onder de kop CSS, maken we gebruik van het Bootstrap framework, meer bepaald Bootstrap 5. Dit zorgt voor een website die op een correcte en elegante manier "breekt" op toestellen met lagere resoluties. Daarenboven maken we voor de styling ook nog gebruik van een bootstrap template, meer hierover is te vinden onder de titel Purrfect Design.

Ajax

Om te vermijden dat de applicatie onbruikbaar wordt, zijn er een aantal pagina's waar we gebruik maken van AJAX-requests. Deze zorgen ervoor dat de overzichtspagina's van de katten en pleeggezinnen niet telkens volledig herladen moeten worden als iemand een nieuwe filter toepast. Ook voor het inladen van de data in de matchmaker wordt hier gebruik van gemaakt.

Database-structuur

cats	
id	bigint unsigned
gender	varchar(255)
name	varchar(255)
dateOfBirth	date
breed	varchar(255)
furColor	varchar(255)
furLength	varchar(255)
chipNumber	varchar(255)
adoptionStatus	varchar(255)
notifierName	varchar(255)
notifierPhone	varchar(255)
socialization	varchar(255)
startWeight	int
sterilized	varchar(255)
extraInfo	varchar(255)
medication	varchar(255)
personality	varchar(255)
solo	varchar(255)
withPet	varchar(255)
gardenAccess	varchar(255)
buddyId	int
image	varchar(255)
fosterFamily_id	int
created_at	timestamp
updated_at	timestamp

catpreferences	
id	bigint unsigned
cat_id	bigint unsigned
bottleFeeding	tinyint(1)
pregnancy	tinyint(1)
intensiveCare	tinyint(1)
noIntensiveCare	tinyint(1)
isolation	tinyint(1)
kids	tinyint(1)
dogs	tinyint(1)
cats	tinyint(1)
lapCat	tinyint(1)
playfulCat	tinyint(1)
outdoorCat	tinyint(1)
calmCat	tinyint(1)
bedroomAccess	tinyint(1)
created_at	timestamp
updated_at	timestamp

weighings	
id	bigint unsigned
cat_id	int
date	date
weight	int
comments	varchar(255)
created_at	timestamp
updated_at	timestamp

vetVisits	
id	bigint unsigned
cat_id	int
date	date
reason	varchar(255)

foster_preferences		fosterFamilies	
id	bigint unsigned	id	bigint unsigned
fosterFamily_id	bigint unsigned	lastName	varchar(255)
adult	int	firstName	varchar(255)
pregnant	int	dateOfBirth	date
kitten	int	street	varchar(255)
bottleFeeding	int	number	varchar(255)
scared	int	city	varchar(255)
feral	int	zipCode	varchar(255)
intensiveCare	int	phone	varchar(255)
noIntensiveCare	int	availableSpots	int
isolation	int	created_at	timestamp
created_at	timestamp	updated_at	timestamp
updated_at	timestamp		

shelters		pets	
id	bigint unsigned	id	bigint unsigned
shelterName	varchar(255)	fosterFamily_id	int
shelterPhone	varchar(255)	species	varchar(255)
hkNumber	varchar(255)	dateOfBirth	date
shelterFirstName	varchar(255)	created_at	timestamp
shelterLastName	varchar(255)	updated_at	timestamp
shelterDateOfBirth	date		
shelterStreet	varchar(255)		
shelterNumber	varchar(255)		
shelterCity	varchar(255)		
shelterZipCode	varchar(255)		
phoneNumber	varchar(255)		
website	varchar(255)		
picture	varchar(255)		
created_at	timestamp		
updated_at	timestamp		

roommates		notifications	
id	bigint unsigned	id	bigint unsigned
fosterFamily_id	int	cat_id	bigint unsigned
relation	varchar(255)		
dateOfBirth	date		
created_at	timestamp		
updated_at	timestamp		

users	
id	bigint unsigned
email	varchar(255)
email_verified_at	timestamp
password	varchar(255)
fosterFamily_id	bigint unsigned
shelter_id	bigint unsigned
remember_token	varchar(100)
created_at	timestamp
updated_at	timestamp

personal_access_tokens	
id	bigint unsigned
tokenable_type	varchar(255)
tokenable_id	bigint unsigned
name	varchar(255)
token	varchar(64)
abilities	text
last_used_at	timestamp
created_at	timestamp
updated_at	timestamp

password_resets	
email	varchar(255)
token	varchar(255)
created_at	timestamp

Custom Code

GH = De andere cases in de functie zijn analoog maar hiervoor verwijzen we graag naar de code op github

AuthenticatedSessionController

```
AuthController.php ×
app > Http > Controllers > Auth > AuthController.php
34  public function store(LoginRequest $request)
35  {
36      $request->authenticate();
37      $request->session()->regenerate();
38      // Route to correct dashboard if shelter or foster
39      if ($foster_id_crypt = auth()->user()->fosterFamily_id) {
40          $foster_id_crypt = Crypt::encryptString(auth()->user()->fosterFamily_id);
41          return redirect()->route('welcome', $foster_id_crypt); //['fosterId' => $request->fosterFamily]);
42      } elseif ($shelter_id_crypt = auth()->user()->shelter_id) {
43          $shelter_id_crypt = Crypt::encryptString(auth()->user()->shelter_id);
44          return redirect()->route(['welcome', $shelter_id_crypt]);
45      }
46  }
47
48 /**
49 * Destroy an authenticated session.
50 *
51 * @param \Illuminate\Http\Request $request
52 * @return \Illuminate\Http\RedirectResponse
53 */
54 public function destroy(Request $request)
55 {
56     Auth::guard('web')->logout();
57     $request->session()->invalidate();
58     $request->session()->regenerateToken();
59     return redirect('/');
60 }
61
62 // Route to fosterAccount
63 public function getFosterAccount($id)
64 {
65     // Decrypt ID & show fosterfamily details
66     $fosterFamilyDecryptID = Crypt::decryptString($id);
67     $user = User::find(auth()->user()->id);
68     $fosterFamily = FosterFamily::where('id', '=', $fosterFamilyDecryptID)->firstOrFail();
69     $fosterPreference = FosterPreference::where('fosterFamily_id', '=', $fosterFamily->id)->firstOrFail();
70     // Show pets & roommates
71
72     $roommates = PetsAndRoommatesController::showRoommatesByFosterId($fosterFamily->id);
73     $pets = PetsAndRoommatesController::showPetsByFosterId($fosterFamily->id);
74     $species = ['Kat', 'Hond', 'Knaagdier', 'Vogel'];
75     $relation = ['Partner', 'Kind', 'Ouder'];
76
77     return view('auth.fosterAccount', compact('fosterFamily', 'user', 'fosterPreference', 'roommates', 'pets', 'species', 'relation'));
78 }
79
80 // Route to shelterAccount
81 public function getShelterAccount($id)
82 {
83     // Decrypt ID & show shelterAccount details
84     $shelterDecryptID = Crypt::decryptString($id);
85     $user = User::find(auth()->user()->id);
86     $shelter = Shelter::where('id', '=', $shelterDecryptID)->firstOrFail();
87     return view('auth.shelterAccount', compact('user', 'shelter'));
88 }
```

RegisteredUserController

```
RegisteredUserController.php X
app > Http > Controllers > Auth > RegisteredUserController.php
25 |     public function createFoster()
26 |     {
27 |         return view('auth.fosterAccount');
28 |     }
29 |
30 |     public function createShelter()
31 |     {
32 |         return view('auth.shelterAccount');
33 |     }
34 |
35 |
36 |    /**
37 |     * Handle an incoming registration request.
38 |     *
39 |     * @param \Illuminate\Http\Request $request
40 |     * @return \Illuminate\Http\RedirectResponse
41 |     *
42 |     * @throws \Illuminate\Validation\ValidationException
43 |     */
44 |    public function storeFoster(Request $request)
45 |    {
46 |        // FOSTERFAMILY TABLE //
47 |        $request->validate([
48 |            'lastName' => ['required', 'string'],
49 |            'firstName' => ['required', 'string'],
50 |            'dateOfBirth' => ['required', 'date'],
51 |            'street' => ['required', 'string'],
52 |            'number' => ['required', 'string'],
53 |            'city' => ['required', 'string'],
54 |            'zipCode' => ['required', 'string'],
55 |            'phone' => ['required', 'string'],
56 |            'availableSpots' => ['required', 'integer'],
57 |        ]);
58 |
59 |        $foster = FosterFamily::firstOrCreate([
60 |            'lastName' => $request->input('lastName'),
61 |            'firstName' => $request->input('firstName'),
62 |            'dateOfBirth' => $request->input('dateOfBirth'),
63 |            'street' => $request->input('street'),
64 |            'number' => $request->input('number'),
65 |            'city' => $request->input('city'),
66 |            'zipCode' => $request->input('zipCode'),
67 |            'phone' => $request->input('phone'),
68 |            'availableSpots' => $request->input('availableSpots'),
69 |        ]);
70 |
71 |        // USER TABLE //
72 |        $request->validate([
73 |            'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
74 |            'password' => ['required', Rules\Password::defaults()],
75 |            'fosterFamily_id' => 'integer',
76 |            'shelter_id' => 'null'
77 |        ]);
78 |
79 |        $user = User::firstOrCreate([
80 |            'email' => $request->input('email'),
81 |            'password' => Hash::make($request->input('password')),
82 |            'fosterFamily_id' => $foster->id,
83 |            'shelter_id' => null
84 |        ]);
85 |
86 |        // FOSTER_PREFERENCES TABLE //
87 |        $request->validate([
88 |            'fosterFamily_id' => ['integer'],
89 |            'adult' => ['integer'],
90 |            'pregnant' => ['integer'],
91 |            'kitten' => ['integer'],
92 |            'bottleFeeding' => ['integer'],
93 |            'scared' => ['integer'],
94 |            'feral' => ['integer'],
95 |            'intensiveCare' => ['integer'],
96 |            'noIntensiveCare' => ['integer'],
97 |            'isolation' => ['integer']
98 |        ]);
99 |    }
```

```

99
100    $fosterPreference = FosterPreference::firstOrCreate([
101        'fosterFamily_id' => $foster->id,
102        'adult' => $request->input('adult'),
103        'pregnant' => $request->input('pregnant'),
104        'kitten' => $request->input('kitten'),
105        'bottleFeeding' => $request->input('bottleFeeding'),
106        'scared' => $request->input('scared'),
107        'feral' => $request->input('feral'),
108        'intensiveCare' => $request->input('intensiveCare'),
109        'noIntensiveCare' => $request->input('noIntensiveCare'),
110        'isolation' => $request->input('isolation'),
111    ]);
112
113    event(new Registered($user));
114
115    Auth::login($user);
116
117    $roommates = PetsAndRoommatesController::showRoommatesByFosterId($foster->id);
118    $pets = PetsAndRoommatesController::showPetsByFosterId($foster->id);
119    $species =(['Kat', 'Hond', 'Knaagdier', 'Vogel']);
120    $relation =(['Partner', 'Kind', 'Ouder']);
121
122    $foster_id_crypt = Crypt::encryptString(auth()->user()->fosterFamily_id);
123    return redirect()->route('welcome', $foster_id_crypt);
124}
125
126 // Update FosterAccount page with User, FosterPreference & FosterFamily information
127 // Same page as registerpage
128 public function updateFoster(Request $request, $user)
129 {
130     $id = $request->input('fosterFamily_id');
131     $fosterFamily = FosterFamily::find($id);
132     $fosterFamily->lastName = $request->input('lastName');
133     $fosterFamily->firstName = $request->input('firstName');
134     $fosterFamily->dateOfBirth = $request->input('dateOfBirth');
135     $fosterFamily->street = $request->input('street');

136     $fosterFamily->number = $request->input('number');
137     $fosterFamily->city = $request->input('city');
138     $fosterFamily->zipCode = $request->input('zipCode');
139     $fosterFamily->phone = $request->input('phone');
140     $fosterFamily->availableSpots = $request->input('availableSpots');
141     $fosterFamily->save();

142     $user = User::find(auth()->user()->id);
143     $user->email = $request->input('email');
144     $user->save();

145     FosterPreference::where('fosterFamily_id', $id)->update([
146         'adult' => $request->input('adult'),
147         'pregnant' => $request->input('pregnant'),
148         'kitten' => $request->input('kitten'),
149         'bottleFeeding' => $request->input('bottleFeeding'),
150         'feral' => $request->input('feral'),
151         'intensiveCare' => $request->input('intensiveCare'),
152         'noIntensiveCare' => $request->input('noIntensiveCare'),
153         'isolation' => $request->input('isolation'),
154     ]);
155
156     $roommates = PetsAndRoommatesController::showRoommatesByFosterId($fosterFamily->id);
157     $pets = PetsAndRoommatesController::showPetsByFosterId($fosterFamily->id);
158     $species =(['Kat', 'Hond', 'Knaagdier', 'Vogel']);
159     $relation =(['Partner', 'Kind', 'Ouder']);
160     // Encrypt ID in URL for security purposes
161     $foster_id_crypt = Crypt::encryptString(auth()->user()->fosterFamily_id);
162
163     return redirect()->route('fosterAccount', $foster_id_crypt);
164 }
165
166 }
```

FosterMiddleware

```

app > Http > Middleware > 🗃 FosterMiddleware.php X
17     // FosterMiddleware restricts POST actions
18     public function handle(Request $request, Closure $next)
19     {
20         if (!auth()->check() || auth()->user()->fosterFamily_id == null) {
21             abort(code: 403, message: 'U bent niet bevoegd voor deze actie.');
22         }
23         return $next($request);
24     }
25 }
```

CatController



CatController.php

app > Http > Controllers > CatController.php

```
32  /**
33  * Convert the date of birth in a human readable string format
34  * @param Date $birthDate the date of birth to be converted in a human readable string format
35  * @return string the date of birth formatted in human readable string
36  */
37  public static function getCatAgeString($birthDate) {
38      $carbonBirthDate = Carbon::parse($birthDate);
39      $diffYears = $carbonBirthDate->DiffInYears(Carbon::now());
40      $diffMonths = $carbonBirthDate->diffInMonths(Carbon::now());
41      $diffWeeks = $carbonBirthDate->diffInWeeks(Carbon::now());
42      if ($diffYears > 0) {
43          return $diffYears . ' jaar';
44      } else if ($diffMonths > 0) {
45          return $diffMonths . ($diffMonths > 1) ? ' maanden' : ' maand';
46      } else {
47          return $diffWeeks . ($diffWeeks > 1) ? ' weken' : ' week';
48      }
49  }
```

```
67  /**
68  * Filter all cats based on their name
69  * @param string $string the string that we try to find in the name of the cats
70  * @return array a json-encoded array filled with all cats whose name includes the given string
71  */
72  public function filterCatsByString($string) {
73      $cats = DB::table('cats')->select('cats.*')->leftJoin('fosterFamilies', 'cats.fosterFamily_id', '=', 'fosterFamilies.id');
74      $cats = $cats->where('cats.name', 'LIKE', '%' . $string . '%');
75      $result = $cats->addSelect('fosterFamilies.dateOfBirth AS fosterBirth')->addSelect('fosterFamilies.firstName AS fosterFirstName');
76      $array = json_decode(json_encode($result), true);
77      $result = [];
78      foreach ($array as $row) {
79          //dd($row);
80          $row += ['stringDate' => CatController::getCatAgeString($row['dateOfBirth'])];
81          array_push($result, $row);
82      }
83  }
84 }
```



CatController.php

```
92  /**
93  * Apply a filter to the given query that filters on the placement options of the cats based on the given value
94  * @param array $value an array of strings that specify which placement options of cats should be filtered
95  * @param QueryBuilder $query the query builder that we need to add additional where clauses to based on the given placement options
96  * @return QueryBuilder the given query builder appended by the where clauses for each given value
97  */
98  public static function filterPlacement($value, $query) {
99      if (is_array($value)) {
100          $first = true;
101          foreach ($value as $val) {
102              if (str_starts_with($val, 'Solo')) {
103                  if ($first) {
104                      $first = false;
105                      $query = $query->where('Solo', explode(' ', $val)[1]);
106                  } else {
107                      $query = $query->orWhere('Solo', explode(' ', $val)[1]);
108                  }
109              } else if (str_starts_with($val, 'Huisdier')) {
110                  if ($first) {
111                      $first = false;
112                      $query = $query->where('withPet', explode(' ', $val)[1]);
113                  } else {
114                      $query = $query->orWhere('withPet', explode(' ', $val)[1]);
115                  }
116          }
117      }
118  }
```

GH

```

136  /**
137  * Apply a filter to the given query that filters on the character of the cats based on the given value
138  * @param array $value an array of strings that specify which characters of cats should be filtered
139  * @param QueryBuilder $query the query builder that we need to add additional where clauses to based on the given character-strings
140  * @return QueryBuilder the given query builder appended by the where clauses for each given value
141  */
142 public static function filterCharacter($value, $query) {
143     if (is_array($value)) {
144         $query->where(function ($query) use ($value) {
145             $first = true;
146             foreach ($value as $val) {
147                 switch ($val) {
148                     case 'playfulCat':
149                         if ($first) {
150                             $first = false;
151                             $query = $query->where('catPreferences.playfulCat', 1);
152                         } else {
153                             $query = $query->orWhere('catPreferences.playfulCat', 1);
154                         }
155                         break;
156                     case 'lapCat':
157                         if ($first) {
158                             $first = false;
159                             $query = $query->where('catPreferences.lapCat', 1);
160                         } else {
161                             $query = $query->orWhere('catPreferences.lapCat', 1);
162                         }
163                         break;

```

GH

```

208 /**
209 * Apply a filter to the given query that filters on the age of the cats based on the given value
210 * @param array $value an array of strings that specify which age categories of cats should be filtered
211 * @param QueryBuilder $query the query builder that we need to add additional where clauses to based on the given values
212 * @return QueryBuilder the given query builder appended by the where clauses for each given value
213 */
214 public static function filterAge($value, $query) {
215     if (is_array($value)) {
216         $first = true;
217         foreach ($value as $val) {
218             switch ($val) {
219                 case 'kitten':
220                     if ($first) {
221                         $first = false;
222                         $query = $query->whereDate('cats.dateOfBirth', '>=', Carbon::now()->subYear()->toDateString());
223                     } else {
224                         $query = $query->orWhereDate('cats.dateOfBirth', '>=', Carbon::now()->subYear()->toDateString());
225                     }
226                     break;
227                 case 'adolescent':
228                     if ($first) {
229                         $first = false;
230                         $query = $query->whereBetween('cats.dateOfBirth', [Carbon::now()->subYears(2)->toDateString(), Carbon::now()]);
231                     } else {
232                         $query = $query->orWhereBetween('cats.dateOfBirth', [Carbon::now()->subYears(2)->toDateString(), Carbon::now()]);
233                     }
234                     break;

```

GH

```

302 /**
303 * Get the category of ages in which a given cat is located at this moment in time
304 * @param Cat $cat the cat that we need an age category for
305 * @return string the age category the given cat is in.
306 * <1year => kitten, <2years => adolescent, <8years => adult, >8years => senior
307 */
308 public static function getCatAgeCategory($cat) {
309     $catDate = new Carbon($cat->dateOfBirth);
310     $kittenDate = Carbon::now()->subYear();
311     $adolescentDate = Carbon::now()->subYears(2);
312     $adultDate = Carbon::now()->subYear(8);
313     if ($catDate->greaterThanOrEqualTo($kittenDate) ) { return "kitten"; }
314     else if ($catDate->greaterThanOrEqualTo($adolescentDate) ) { return "adolescent"; }
315     else if ($catDate->greaterThanOrEqualTo($adultDate) ) { return "adult"; }
316     else { return "senior"; }
317 }

```

```

319     /**
320      * Filter all cats according to the selected parameters.
321      * @param Request $request A POST-request object containing a possible parameter (which are arrays) for each filter option
322      * @return array A json-encoded array containing all cats that match the selected parameters
323     */
324    public function filterCats(Request $request) {
325        /*$validation = $request->validate([
326            'status'      => 'required',
327            'email'       => 'required',
328            'mobile'      => 'required',
329            'message'     => 'required',
330        ]);*/
331        $data = $request->all();
332        $cats = DB::table('cats')->select('cats.*')->join('catPreferences', 'cats.id', '=', 'catPreferences.cat_id')->leftJoin('fosterF
333        // Filter
334        //dd($data);
335        if (isset($data['character'])) {
336            $cats = CatController::filterCharacter($data['character'], $cats);
337        }
338        if (isset($data['gender'])) {
339            $cats = CatController::basicFilter($data['gender'], $cats, 'gender');
340        }
341        if (isset($data['status'])) {
342            $cats = CatController::basicFilter($data['status'], $cats, 'adoptionStatus');
343        }
344        if (isset($data['fosterFamily'])) {
345            $cats = CatController::basicFilter($data['fosterFamily'], $cats, 'fosterFamily_id');
346        }
347
348        if (isset($data['placement'])) {
349            $cats = CatController::filterPlacement($data['placement'], $cats);
350        }
351        if (isset($data['age'])) {
352            $cats = CatController::filterAge($data['age'], $cats);
353        }
354        //dd($cats->toSql());
355
356        // Use a join to include all the needed fields that are filtered on
357
358        $result = $cats->addSelect('fosterFamilies.dateOfBirth AS fosterBirth')->addSelect('fosterFamilies.firstName AS fosterFirstName');
359        $array = json_decode(json_encode($result), true);
360        $result = [];
361        foreach ($array as $row) {
362            //dd($row);
363            $row += ['stringDate' => CatController::getCatAgeString($row['dateOfBirth'])];
364            if (isset($row['fosterFamily_id'])) {
365                $row += ['fosterHashed' => Crypt::encryptString($row['fosterFamily_id'])];
366            }
367            array_push($result, $row);
368        }
369        return json_encode($result);
370    }

```

FosterFamilyController

```
File: FosterFamilyController.php
Line: 25
    /**
     * Get the amount of cats that a fosterFamily is currently housing
     * @param integer $fosterFamilyId the id of the fosterFamily we are interested in
     * @return integer the amount of cats that the given fosterFamily is currently housing
     */
    public static function getAmountOfCats($fosterFamilyId)
    {
        $count = Cat::where('fosterFamily_id', '=', $fosterFamilyId)
            ->where(function ($subQuery) {
                $subQuery->where('adoptionStatus', '=', 'Bij Pleeggezin')
                ->orWhere('adoptionStatus', '=', 'Klaar voor adoptie')
                ->orWhere('adoptionStatus', '=', 'In optie')
                ->orWhere('adoptionStatus', '=', 'Adoptie goedgekeurd');
            });
        //dd($count->get());
        return $count->get()->count();
    }

Line: 55
    /**
     * Get a fosterFamily with the given id
     * @param integer $id the id of the requested fosterFamily
     * @return array an json-encoded array containing the requested fosterFamily along with the hashed id and the current available spots
     */
    public function getFosterFamilyById($id)
    {
        $result = FosterFamily::findOrFail($id);
        $result = json_decode(json_encode($result), true);
        $result['hashed'] = Crypt::encryptString($result['id']);
        $result['availableSpots'] = $result['availableSpots'] - FosterFamilyController::getAmountOfCats($id);
        return json_encode($result);
    }

Line: 69
    /**
     * Get all fosterPreferences of a FosterFamily
     * @param integer $id the id of the FosterFamily that we want the preferences of
     * @return array a json-encoded array containing the preferences of the given fosterFamily along with their email, pets and roommates
     */
    public function getPreferenceByFosterId($id)
    {
        $result = [];
        $result['preferences'] = json_decode(json_encode(FosterFamily::findOrFail($id)->preferences));
        $result['pets'] = json_decode(json_encode(FosterFamily::findOrFail($id)->pets));
        $result['roommates'] = json_decode(json_encode(FosterFamily::findOrFail($id)->roommates));
        $result['email'] = User::where('fosterFamily_id', $id)->select('email')->first();
        return json_encode($result);
    }
```

```

84  /**
85  * Filter all fosterFamilies based on the given values for possible catPreferences
86  * @param array $value an array of strings representing what catPreferences the fosterFamily should accept
87  * @param QueryBuilder $query the query builder that we need to add additional where clauses to based on the given values
88  * @param bool $noOr an optional parameter indicating whether we want to AND instead of OR between our where clauses
89  * @return QueryBuilder the given query builder appended by the where clauses for each given value
90  */
91 public static function filterByCatPref($value, $query, $noOr = false)
92 {
93     if (is_array($value)) {
94         $first = true;
95         foreach ($value as $val) {
96             if (!strcmp($val, 'adult')) {
97                 if ($first) {
98                     $first = false;
99                     $query = $query->where('foster_preferences.adult', 1);
100                } else if ($noOr) {
101                    $query = $query->where('foster_preferences.adult', 1);
102                } else {
103                    $query = $query->orWhere('foster_preferences.adult', 1);
104                }
105            } else if (!strcmp($val, 'pregnant')) {
106                if ($first) {
107                    $first = false;
108                    $query = $query->where('foster_preferences.pregnant', 1);
109                } else if ($noOr) {
110                    $query = $query->where('foster_preferences.pregnant', 1);
111                } else {
112                    $query = $query->orWhere('foster_preferences.pregnant', 1);
113                }
114            }
115        }
116    }

```

GH

```

191 /**
192 * Filter all fosterFamilies based on the given values for their home situation
193 * @param array $value an array of strings representing what home situations are NOT possible
194 * @param QueryBuilder $query the query builder that we need to add additional where clauses to based on the given values
195 * @param bool $noOr an optional parameter indicating whether we want to AND instead of OR between our where clauses
196 * @return QueryBuilder the given query builder appended by the where clauses for each given value
197 */
198 public static function filterHomeSituation($value, $query, $noOr = false)
199 {
200     // $query = DB::table('fosterFamilies')->leftJoin('pets', 'fosterFamilies.id', '=', 'pets.fosterFamily_id')->select('fosterFamilies.*');
201     if (is_array($value)) {
202         $first = true;
203         foreach ($value as $val) {
204             if (str_ends_with($val, 'kids')) {
205                 $now = Carbon::now()->subYears(12)->toDateString();
206                 $subQuery = DB::table('fosterFamilies')->join('roommates', 'fosterFamilies.id', '=', 'roommates.fosterFamily_id')->
207                 $subQuery = $subQuery->where('roommates.dateOfBirth', '>', $now);
208                 if ($first) {
209                     $first = false;
210                     $query = $query->whereNotIn('fosterFamilies.id', $subQuery);
211                 } else if ($noOr) {
212                     $query = $query->whereNotIn('fosterFamilies.id', $subQuery);
213                 } else {
214                     $query = $query->orWhereNotIn('fosterFamilies.id', $subQuery);
215                 }
216             } else if (str_ends_with($val, 'pets')) {
217                 $subQuery = DB::table('fosterFamilies')->join('pets', 'fosterFamilies.id', '=', 'pets.fosterFamily_id')->select('fosterFamilies.*');
218                 if ($first) {
219                     $first = false;
220                     $query = $query->whereNotIn('fosterFamilies.id', $subQuery);
221                 } else if ($noOr) {
222                     $query = $query->whereNotIn('fosterFamilies.id', $subQuery);
223                 } else {
224                     $query = $query->orWhereNotIn('fosterFamilies.id', $subQuery);
225                 }
226             }
227         }
228     }

```

```

262 /**
263 * Filter all fosterFamilies based on their name
264 * @param string $string the string that we try to find in the name of the fosterFamilies
265 * @return array a json-encoded array filled with all fosterFamilies whose name includes the given string
266 */
267 public function filterFosterFamiliesByString($string)
{
    $fosterFamilies = DB::table('fosterFamilies')->select('fosterFamilies.*');
    $fosterFamilies = $fosterFamilies->where('fosterFamilies.firstName', 'LIKE', '%' . $string . '%')
        ->orWhere('fosterFamilies.lastName', 'LIKE', '%' . $string . '%');
    $result = $fosterFamilies->get();
    $result = json_decode(json_encode($result), true);
    foreach ($result as $row) {
        $row['hashed'] = Crypt::encryptString($row['id']);
        $row['availableSpots'] = FosterFamilyController::getAmountOfCats($row['id']);
    }
    return json_encode($result);
}

```

```

281 /**
282 * Filter all fosterFamilies according to the selected parameters.
283 * @param Request $request A POST-request object containing a possible parameter (which are arrays) for each filter option
284 * @return array A json-encoded array containing all fosterFamilies that match the selected parameters, with the actual availableSp
285 */
286 public function filterFosterFamilies(Request $request)
{
    $data = $request->all();

    $fosterFamilies = DB::table('fosterFamilies')->leftJoin('foster_preferences', 'fosterFamilies.id', '=', 'foster_preferences.fos
    // $fosterFamilies = $fosterFamilies->where('foster_preferences.adult', '=', 1);
    if (isset($data['availableSpots'])) {
        $fosterFamilies = $fosterFamilies->where('availableSpots', '=', $data['availableSpots']);
    }
    if (isset($data['livingStatus'])) {
        $fosterFamilies = FosterFamilyController::filterHomeSituation($data['livingStatus'], $fosterFamilies);
    }
    if (isset($data['catPreferences'])) {
        $fosterFamilies = FosterFamilyController::filterByCatPref($data['catPreferences'], $fosterFamilies);
    }
    $result = $fosterFamilies->get();
    $result = json_decode(json_encode($result), true);
    $final = [];
    foreach ($result as &$row) {
        $row['hashed'] = Crypt::encryptString($row['id']);
        $row['availableSpots'] = FosterFamilyController::getAmountOfCats($row['id']);
        // $row[''];
    }
    // dd($result);
    return json_encode($result);
}

```

DashboardController

```
24 /**
25 * Store a new Notification in the database.
26 *
27 * @param \Illuminate\Http\Request $request
28 * @return \Illuminate\Http\Response
29 */
30 public function store(Request $request)
31 {
32     // Validate the request...
33     //validated = $request->validate(['cat' => 'required|integer', 'type' => 'required', 'message' => 'required']);
34
35     $notification = new Notification;
36
37     if (is_numeric($request->cat)) { $notification->cat_id = $request->cat; }
38     else { $notification->cat_id = null; }
39     $notification->type = $request->type;
40     $notification->message = $request->message;
41
42     $notification->sentByShelter = 0;
43
44     $notification->fosterFamily_id = Auth::user()->fosterFamily_id;
45
46
47     $notification->save();
48     return redirect()->route('notifications', ['fosterId' => $request->fosterFamily]);
49 }
```

```
50 /**
51 * Store a new shelterNotification in the database.
52 *
53 * @param \Illuminate\Http\Request $request
54 * @return \Illuminate\Http\Response
55 */
56 public function storeShelter(Request $request)
57 {
58     // Validate the request...
59
60     $notification = new Notification;
61
62     if (is_numeric($request->cat)) { $notification->cat_id = $request->cat; }
63     else { $notification->cat_id = null; }
64     $notification->type = $request->type;
65     $notification->message = $request->message;
66
67     $notification->sentByShelter = 1;
68
69     $notification->fosterFamily_id = $request->fosterFamily;
70
71
72     $notification->save();
73     return redirect()->route('shelterNotifications');
74 }
```

```
76 /**
77 * Delete a notification
78 * @param integer $id the id of the notification to be deleted
79 */
80 public function delete($id)
81 {
82     $notification = Notification::find($id);
83     $sentByShelter = $notification->sentByShelter;
84     $fId = $notification->fosterFamily_id;
85     $notification->delete();
86     //dd($notifications[0]->fosterFamilyId);
87     if ($sentByShelter) {
88         return redirect()->route('notifications', ['fosterId' => $fId]);
89     } else {
90         return redirect()->route('shelterNotifications');
91     }
92 }
```

```

118 * Apply a filter to the given query that filters the fosterFamilies based on the preferences of the cat that is given
119 * @param Cat $cat the cat on whose preferences is to be filtered
120 * @param QueryBuilder $query the query builder that we need to add additional where clauses to based on the given cat
121 * @return QueryBuilder $query the given query builder appended by the where clauses for each preference of the given cat
122 */
123 private static function checkFostersOnCatPref($cat, $query)
124 {
125     $filterInput = [($catController::getCatAgeCategory($cat) == "kitten" ? "kitten" : "adult")];
126     if (isset($cat->preferences)) {
127         $catPreferences = $cat->preferences;
128         if ($cat->socialization == "Bang") { $filterInput[] = "scared"; }
129         if ($cat->socialization == "Wild") { $filterInput[] = "feral"; }
130         if ($catPreferences->bottleFeeding) { $filterInput[] = "bottleFeeding"; }
131         if ($catPreferences->noIntensiveCare){ $filterInput[] = "noIntensiveCare"; }
132         if ($catPreferences->intensiveCare) { $filterInput[] = "intensiveCare"; }
133         if ($catPreferences->pregnancy) { $filterInput[] = "pregnant"; }
134         if ($catPreferences->isolation) { $filterInput[] = "isolation"; }
135         $query = $fosterFamilyController::filterByCatPref($filterInput, $query, true);
136         $houseFilter = [];
137         if (!$catPreferences->dogs) { $houseFilter[] = "no dogs"; }
138         if (!$catPreferences->cats) { $houseFilter[] = "no cats"; }
139         if (!$catPreferences->kids) { $houseFilter[] = "no kids"; }
140         if ($houseFilter != []) { $query = $fosterFamilyController::filterHomeSituation($houseFilter, $query, true); }
141     }
142
143     //dd($query->get());
144     return $query;
145 }

```

```

147 /**
148 * Get all fosterFamilies that can be matched to the selected cat
149 * @param integer $catId The id of the selected cat
150 * @return array $cats Return json_encoded array of the fosterFamilies that can be placed with the selected cat
151 */
152 public function getFosterFamiliesBySelectedCat($catId)
153 {
154     if ($catId < 0 ) {return json_encode($cat::whereNull("fosterFamily_id")->get());}
155     else {
156         $selectedCat = $cat::findOrFail($catId);
157         $fosterFamilies = DB::table('fosterFamilies')->select("fosterFamilies.*")->leftJoin('foster_preferences', 'fosterFamilies'.
158         $fosterFamilies = $dashBoardController::checkFostersOnCatPref($selectedCat, $fosterFamilies);
159         $tmp = json_decode(json_encode($fosterFamilies->get()), true);
160         $result = [];
161         foreach ($tmp as $foster) {
162             if ($foster['availableSpots']-$fosterFamilyController::getAmountOfCats($foster['id']) > 0) {
163                 $result[] = $foster;
164             }
165         }
166         return json_encode($result);
167     }
168 }

```

FosterOverview

```
94     function createAndFillCards(data, container) {
95         filteredCats = data;
96         var cats = container;
97         cats.empty();
98         console.log(filteredCats);
99         filteredCats.forEach(cat => {
100             //console.log(cat);
101             let string = `<div class="col-md-4 stretch-card grid-margin">
102                 <div class="card card-img-holder">
103                     <div class="card">
104                         <div class="card-header bg-gradient-info">
105                             <b>` + cat.firstName + ` ` + cat.lastName + `</b></h4>
107                         </div>
108                         <div class="card-footer card-border-info">
109                             <div class="mb-3">` + cat.availableSpots + `(cat.availableSpots>1? ` beschikbare plaatsen`:
110                                         {-- TODO: add route once detail of foster family is available --})
111                             <div><a href="/pleeggezinAccount/${cat.hashed}" class="text-black"><u>Meer info</u></a>
112                         </div>
113                     </div>
114                 </div>`;
115             cats.append(string);
116         });
117         console.log(data);
118     }
119 }
120 
```



```
129 $('#searchTerm').on('keyup', function() {
130     var searchTerm = $(this).val();
131     var fosterFamilies = $("#fosterFamilies");
132     if (searchTerm == '') {
133         $.ajax({
134             url: '/fosterfamilies/ajax/',
135             type: "POST",
136             data: {
137                 "_token": "{{ csrf_token() }}"
138             },
139             dataType: "json",
140             success: function(data) {
141                 createAndFillCards(data, fosterFamilies);
142             }
143         });
144     } else {
145         $.ajax({
146             url: '/fosterfamilies/ajax/' + searchTerm,
147             type: "GET",
148             dataType: "json",
149             success: function(data) {
150                 createAndFillCards(data, fosterFamilies);
151             }
152         });
153     }
154 });

```



```
155 $('.fosterFilter').on('change', function() {
156     var availableSpots = $('#availableSpots').val();
157     var catPreferences = $('#catPreferences').val();
158     var livingStatus = $('#livingStatus').val();
159     var fosterFamilies = $("#fosterFamilies");
160     $.ajax({
161         url: '/fosterfamilies/ajax/',
162         type: "POST",
163         data: {
164             "_token": "{{ csrf_token() }}",
165             "catPreferences": catPreferences,
166             "livingStatus": livingStatus,
167             "availableSpots": availableSpots
168         },
169         dataType: "json",
170         success: function(data) {
171             console.log(data);
172             createAndFillCards(data, fosterFamilies);
173         }
174     });

```

ShelterDashboard

```

357     function convertDateToAge(dateString) {
358         var today = new Date();
359         var birthDate = new Date(dateString);
360         var age = today.getFullYear() - birthDate.getFullYear();
361         var m = today.getMonth() - birthDate.getMonth();
362         if (m < 0 || (m === 0 && today.getDate() < birthDate.getDate())) {
363             age--;
364         }
365         if (age == 0) {return "minder dan 1";}
366         return age;
367     }

```

```

368     function getPreferencesFoster(fosterId) {
369         $.ajax({
370             url: '/fosterPref/ajax/' + fosterId,
371             type: "GET",
372             dataType: "json",
373             success: function(data) {
374                 $('#fosterPreferences').empty();
375                 $('#fosterRoommates').empty();
376                 $('#fosterPets').empty();
377                 console.log(data);
378                 data.pets.forEach(pet => {
379                     $('#fosterPets').append('<li>' + pet.species + ', ' + convertDateToAge(pet.dateOfBirth) + ' jaar oud</li>');
380                 });
381                 data.roommates.forEach(roommate => {
382                     $('#fosterRoommates').append('<li>' + roommate.relation + ', ' + convertDateToAge(roommate.dateOfBirth) + ' jaar oud</li>');
383                 });
384                 let preferences = data.preferences;
385                 if (preferences.adult) { $('#fosterPreferences').append('<li>Volwassen katten</li>'); }
386                 if (preferences.pregnant) { $('#fosterPreferences').append('<li>Zwangere katten</li>'); }
387                 if (preferences.kitten) { $('#fosterPreferences').append('<li>Kittens</li>'); }
388                 if (preferences.bottleFeeding) { $('#fosterPreferences').append('<li>Geven van flesvoeding</li>'); }
389                 if (preferences.scared) { $('#fosterPreferences').append('<li>Bange katten</li>'); }
390                 if (preferences.feral) { $('#fosterPreferences').append('<li>Wilde katten</li>'); }
391                 if (preferences.intensiveCare) { $('#fosterPreferences').append('<li>Zieke katten met intensieve verzorging</li>'); }
392                 if (preferences.NoIntensiveCare) { $('#fosterPreferences').append('<li>Zieke katten zonder intensieve verzorging</li>'); }
393                 if (preferences.isolation) { $('#fosterPreferences').append('<li>Katten in isolatie</li>'); }
394
395                 $('#emailFoster').empty();
396                 $('#emailFoster').append(data.email.email);

```

```

408     function getPreferencesCat(catId) {
409         $.ajax({
410             url: '/catPref/ajax/' + catId,
411             type: "GET",
412             dataType: "json",
413             success: function(data) {
414                 $('#catPreferences').empty();
415                 $('#catAttention').empty();
416                 let string;
417                 if (!data.kids) {
418                     string = (data.kids ? ' wel ' : ' niet ');
419                     $('#catPreferences').append('<li>kan ' + string + ' geplaatst worden met kinderen</li>');
420                 }
421                 if (!data.dogs) {
422                     string = (data.dogs ? ' wel ' : ' niet ');
423                     $('#catPreferences').append('<li>kan ' + string + ' geplaatst worden met honden</li>');
424                 }
425                 if (!data.cats) {
426                     string = (data.cats ? ' wel ' : ' niet ');
427                     $('#catPreferences').append('<li>kan ' + string + ' geplaatst worden met katten</li>');
428                 }
429                 if (data.intensiveCare) {
430                     $('#catAttention').append('<li>Heeft intensieve verzorging nodig</li>');
431                 }
432                 if (data.noIntensiveCare) {
433                     $('#catAttention').append(
434                         '<li>Is ziek maar heeft geen intensieve verzorging nodig</li>');
435                 }
436                 if (data.bottleFeeding) {
437                     $('#catAttention').append('<li>Heeft flesvoeding nodig</li>');
438                 }
439                 if (data.pregnancy) {
440                     $('#catAttention').append('<li>Is zwanger</li>');
441                 }
442                 if (data.isolation) {
443                     $('#catAttention').append('<li>Moet in isolatie kunnen zitten</li>');
444                 }

```

```

458     // Notification modal
459     $('select[name="fosterFamily"]').on('change', function() {
460         var fosterId = $(this).val();
461         if (fosterId) {
462             $.ajax({
463                 url: '/asielDashboard/ajax/' + fosterId,
464                 type: "GET",
465                 dataType: "json",
466                 success: function(data) {
467                     $('select[name="cat"]').empty();
468                     $('select[name="cat"]').append(
469                         '<option class="option">Selecteer een kat</option>');
470                     $.each(data, function(key, value) {
471                         $('select[name="cat"]').append('<option value="' +
472                             value['id'] + '">' + (value['name']) +
473                             '</option>');
474                     });
475                 }
476             });
477         } else {
478             $('select[name="cat"]').empty();
479         }
480     });
481 });
482 
```

```

484     // Matchmaker event-handling
485     $('select[name="catMatch"]').on('change', function() {
486         var catId = $(this).val();
487         if (catId >= 0) {
488             $.ajax({
489                 url: '/cat/ajax/' + catId,
490                 type: "GET",
491                 dataType: "json",
492                 success: function(data) {
493                     current_cat = data;
494                     //console.log(current_cat);
495                     $('#dateOfBirthCat').empty();
496                     if (current_cat.dateOfBirth == null) {
497                         $('#dateOfBirthCat').append("Onbekend");
498                     } else {
499                         $('#dateOfBirthCat').append(current_cat.dateOfBirth.toString());
500                     }
501                     $('#genderCat').empty();
502                     if (current_cat.gender == null) {
503                         $('#genderCat').append("Onbekend");
504                     } else {
505                         $('#genderCat').append(current_cat.gender);
506                     }
507                     $('#breedCat').empty();
508                     if (current_cat.breed == null) {
509                         $('#breedCat').append("Onbekend");
510                     } else {
511                         $('#breedCat').append(current_cat.breed);
512                     }
513                     $('#chipNumberCat').empty();
514                     $('#chipNumberCat').append(current_cat.chipNumber)
515                     $('#socializationCat').empty();
516                 }
517             });
518         }
519     });
520 
```

```

516     if (current_cat.socialization == null) {
517         $('#socializationCat').append("Onbekend");
518     } else {
519         $('#socializationCat').append(current_cat.socialization);
520     }
521     $('#startWeightCat').empty();
522     if (current_cat.startWeight == null) {
523         $('#startWeightCat').append("Onbekend");
524     } else {
525         $('#startWeightCat').append(current_cat.startWeight + " Gram");
526     }
527     $('#sterilizedCat').empty();
528     $('#sterilizedCat').append(Number(current_cat.sterilized) ? "Ja" :
529         "Nee");
530     // TODO: add more if extra properties are selected
531     $('#catLink').empty();
532     $('#catLink').append("Volledige fiche van ");
533     $('#catLink').append(current_cat.name);
534     $('#catLink').attr('href', '/katDetail/' + current_cat.id);
535     getPreferencesCat(current_cat.id);
536
537     });
538 } else {
539     current_cat = null;
540     console.log("OOPS");
541 }
542 );

```

```

544     $('select[name="fosterFamilyMatch"]').on('change', function() {
545         var fosterFamilyId = $(this).val();
546         if (fosterFamilyId >= 0) {
547             $.ajax({
548                 url: '/fosterfamily/ajax/' + fosterFamilyId,
549                 type: "GET",
550                 dataType: "json",
551                 success: function(data) {
552                     current_foster = data;
553                     $('#dateOfBirthFoster').empty();
554                     $('#dateOfBirthFoster').append(current_foster.dateOfBirth
555                         .toString());
556                     $('#nameFoster').empty();
557                     $('#nameFoster').append(current_foster.firstName + ' ' +
558                         current_foster.lastName);
559                     $('#addressOneFoster').empty();
560                     $('#addressOneFoster').append(current_foster.street + ' ' +
561                         current_foster.number);
562                     $('#addressTwoFoster').empty();
563                     $('#addressTwoFoster').append(current_foster.zipCode + ' ' +
564                         current_foster.city);
565                     $('#emailFoster').empty();
566                     $('#emailFoster').append(current_foster.email);
567                     $('#availableSpotsFoster').empty();
568                     $('#availableSpotsFoster').append(current_foster.availableSpots);
569                     $('#fosterLink').empty();
570                     $('#fosterLink').append("Volledige fiche van ");
571                     $('#fosterLink').append(current_foster.firstName + ' ' +
572                         current_foster.lastName);
573                     $('#fosterLink').attr('href', '/pleeggezinAccount/' +
574                         current_foster.hashed);
575                     getPreferencesFoster(fosterFamilyId);
576                     //$('#sterilizedCat').empty();
577                     //$('#sterilizedCat').append(current_foster.sterilized);
578                     // TODO: add more if extra properties are selected
579             });
580         } else {
581             current_foster = null;
582         }
583     );

```

```
584      $('select[name="catMatch"]').on('change', function() {
585          var catId = $(this).val();
586          if (catId) {
587              $.ajax({
588                  url: '/asielDashboard/matchmaker/cat/ajax/' + catId,
589                  type: "GET",
590                  dataType: "json",
591                  success: function(data) {
592                      current_foster = data;
593                      $('select[name="fosterFamilyMatch"]').empty();
594                      $('select[name="fosterFamilyMatch"]').append('<option value="-1" class="option">Selecteer</option>');
595                      data.forEach(element => {
596                          $('select[name="fosterFamilyMatch"]').append(`<option value="${element.id}" class="option">${el
597                          });
598                      });
599                  });
600              });
601          });
});
```

Calamiteiten tijdens het project

User tabel

Omdat we in onze applicatie met twee verschillende soorten gebruikers werken, nl. pleeggezinnen en asielbeheerders, leek het ons in het begin van het project logisch om de standaard 'users' tabel van Laravel te veranderen door twee tabellen, elk met hun eigen kenmerken. Dit zorgde echter voor problemen die we niet of met veel moeite konden oplossen. Eén van deze problemen was dat, omdat we met Laravel Breeze werken, we veel extra aanpassingen moesten doen aan de 'interne keuken' van Breeze aangezien zaken zoals de Providers bedoeld zijn voor de standaard users tabel. Na overleg met onze docent besloten we daarom de standaard-users tabel van Laravel te gebruiken voor de login-gegevens, (e-mail en wachtwoord) en de twee aparte tabellen te behouden voor alle overige informatie. Hiermee ging het proces van login en registratie te programmeren makkelijker, maar hadden we ook het voordeel dat we direct gebruik konden maken van de Breeze-functionaliteit, zoals 'error messages' bij een foutieve login, zonder dat er daar veel extra werk voor nodig was.

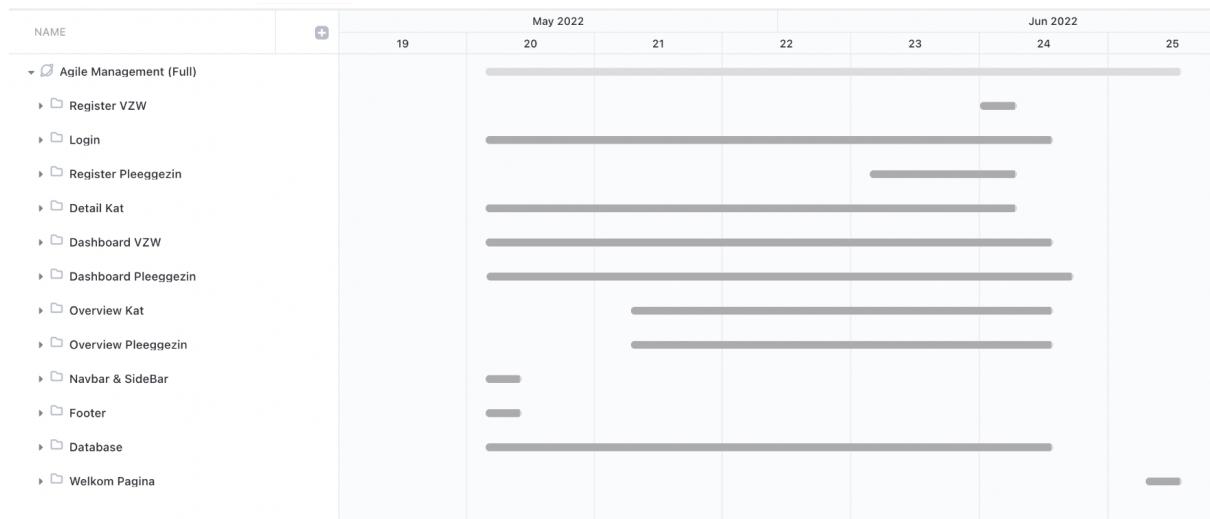
Preferences

Aanvankelijk was het plan om de voorkeuren of preferences van de katten en de pleeggezinnen bij in hun respectievelijke tabellen onder te brengen. Hier waren we dus ook reeds mee aan de slag gegaan. Een aantal weken later in het project hebben we in de Eloquent-documentatie gevonden dat dit ook op een elegantere manier opgelost kan worden. Dit met behulp van HasMany of HasOne methodes. Op deze manier was het mogelijk om deze aan te roepen zonder steeds manueel een join op de beide tabellen te moeten uitvoeren.

Available spots

Doorheen het project hebben we het aantal beschikbare plaatsen bij een pleeggezin bijgehouden. Gedurende de laatste weken, beseften we dat we echter deze waarde nooit aanpassen of bijhouden in een aparte kolom in de database. Om geen volledige update meer te doen aan de structuur van de database, gaan we deze data steeds at run-time berekenen wanneer nodig. Op deze manier kunnen we toch rekening houden met het actueel aantal beschikbare plaatsen, i.p.v. het maximum, bij het plaatsen van een nieuwe kat.

Gantt overzicht van het project



Kostprijsberekening

Uurtarief: €45

TAKEN	SUBTAKEN	UREN	PRIJS
Analyse		51	2295
	Projectidee & scope	4	180
	Applicatie logica & design	47	2115
Ontwikkeling		94	4230
	Views	22	990
	Controllers/Routes	72	3240
Database/Models		9	315
	Tabellen opstellen en normaliseren	3	135
	Models	4	180
	Data toevoegen	2	90
Mockups		2	90
Meetings & transport		16	720
Testing		8	360
Calamiteiten		40% (71,2u)	3204
		TOTAAL: 252,2	TOTAAL: 11214
Opvolging		2	90
Support		2	90
Hosting		/	80
Calamiteiten		25% (1u)	45

Teamwork! Wie deed wat

		Sigurd	Jari	Liesbeth
Concept	Thema van de app	10%	10%	80%
	Functionaliteiten & logica	10%	10%	80%
	Mock-up's, stijl & logo	10%	10%	80%
PHP / Laravel	Login	80%	10%	10%
	Asiel Registratie	80%	10%	10%
	Pleeggezin Registratie	60%	20%	20%
	Kat Pagina	10%	20%	70%
	Asiel Dashboard	10%	80%	10%
	Pleeggezin Dashboard	10%	80%	10%
	Header & footer	45%	10%	45%
	Routes	40%	40%	20%
Database		20%	40%	40%
HTML & CSS		10%	10%	80%
Paper		30%	30%	40%