# API1: Handboek

## Inleiding

In deze herhaling vind je de stappen terug die je geleerd hebt bij het maken van een API1 met Laravel en het gebruik ervan in een front end applicatie.

## Planning

|  |  |  |  |
| --- | --- | --- | --- |
| Actie | Datum | Planning | Werkelijk |
| 1. Maak ontwerp: Eisen en Wensen   Beschrijf Applicatie die je wilt maken en de eisen en wensen die je eraan stelt | 2-mei | 1:00u | 0:30u |
| 1. Maak ontwerp: Use Case diagram   Dit diagram geruik je om met gebruiker en programmeur af te stemmen. | 2-mei | 0:30u | 1:00u |
| 1. Maak ontwerp: Datamodel | 2-mei | 1:00u | 1:00u |
| 1. Maak ontwerp: Api’s   Maak een overzicht van de api's die de applicatie nodig heeft. | 2-mei | 0:30u | 1:00u |
| 1. Bereid het project voor: Maak nieuw Laravel project | 2-mei | 0:30u | 0:30u |
| 1. Bereid het project voor: Maak nieuwe database  * MySql database PersoneelsApiDB * Migrations om de tabellen te maken | 2-mei | 0:30u | 0:30u |
| 1. Bereid het project voor: Genereer models en controllers | 2-mei | 0:30u | 0:30u |
| 1. Routing: Maak de routing volgens API ontwerp | 2-mei | 1:30u | 3:00u |
| 1. Models: Pas aan op basis van datamodel ontwerp | 2-mei | 1:30u | 3:00u |
| 1. Controllers: Pas aan op basis van API ontwerp | 2-mei | 1:00u | 1:00u |
| 1. Testen: Maak een seeder voor de database | 2-mei | 1:00u | 1:00u |
| 1. Testen: Maak de featuretests voor de API’s | 2-mei | 1:30u | 1:20u |
| 1. Maak de Front-End applicatie | 2-mei | 2:00u | 1:50u |
| 1. Testen: Maak een testplan en voer het uit | 2-mei | 1:30u | 1:20u |
| 1. Borrel: Lever applicatie op | 8-mei | 0:10u |  |
| Totaal |  | 10:00u | 11:10u |

## Maak ontwerp: Eisen en Wensen

Er moet een applicatie voor het administreren van de medewerkers worden gemaakt. Van de medewerkers wordt de naam, het telefoonnummer, emailadres en de functie vastgelegd.

De applicatie toont een overzicht van alle medewerkers die de functie hebben die in de combobox is geselecteerd. Wordt er een andere functie geselecteerd, dan wordt het overzicht direct aangepast.

Met de applicatie kun je ook medewerkers verwijderen en toevoegen.

**Must have**

* De front-end wisselt gegevens uit met de backend via Restful API’s.
* De combobox bevat alle functies die werknemers kunnen vervullen. Functies zijn uitsluitend raadpleegbaar (er is dus geen beheer op mogelijk).
* Het overzicht toont alle werknemers die de in de combobox geselecteerde functie uitoefenen.
* Door middel van de ‘x’ kan de gebruiker de betreffende werknemer uit de administratie verwijderen.
* Door middel van de knop ‘toevoegen’ kan de gebruiker een nieuwe werknemer aan de administratie toevoegen.

**Shoud have**

* -

**Could have**

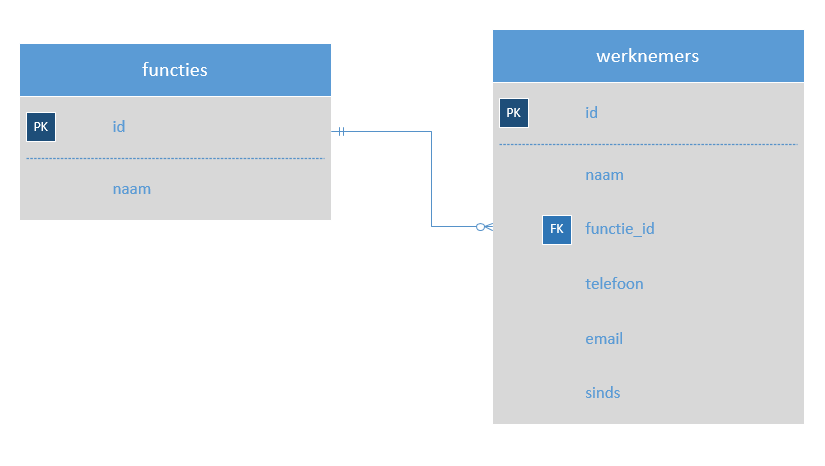
* -

**Won't have**

* -

## Maak ontwerp: Use Case diagram

## Maak ontwerp: Datamodel



|  |  |  |  |
| --- | --- | --- | --- |
| Table | Functies |  |  |
| Kolom | Type |  |  |
| Id | tinyint(4) | Key, required | Autoincrement |
| Naam | varchar(20) | Required |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Table | Werknemers |  |  |
| Kolom | Type |  |  |
| id | tinyint(4) | Key, required | Autoincrement |
| naam | varchar(50) | Required |  |
| functie\_id | tinyint(4) |  | FK Functies |
| telefoon | varchar(20) |  |  |
| email | varchar(50) | Moet geldig emailadres zijn |  |
| sinds | date | Geldige datum |  |

## Maak ontwerp: API’s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | methode | url | json parameters | header | json resultaat |
| **M** | **get** | **api/werknemers/** |  |  | **Alle werknemers** |
| **M** | **post** | **api/werknemers/** | **naam, telefoon, email, functie\_id** |  | **gecreëerde werknemer** |
| M | get | api/werknemers/{id} |  |  | werknemer met opgegeven id |
| W | patch | api/werknemers/{id} | naam en/of telefoon en/of email en/of functie\_id |  | werknemer met opgegeven id |
| **M** | **delete** | **api/werknemers/{id}** |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | methode | url | json parameters | header | json resultaat |
| **M** | **get** | **api/functies/** |  |  | **Alle functies** |
| M | Get | api/functies/{id} |  |  | functie met opgegeven id |

Nb: Noodzaak voor ‘api/functies/{id}’ is afhankelijk van de technische implementatie.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | methode | url | json parameters | header | json resultaat |
|  | get | api/werknemers?naam={naam} |  |  | Alle werknemers met de tekst naam in de naam |
|  | get | api/werknemers?sort={veld} |  |  | Alle werknemers in volgorde van veld |
|  | get | api/functies/{id}/werknemers |  |  | Werknemers met de functie\_id gelijk aan id |
|  | get | api/functies/{id}/werknemers?sort={veld} |  |  | Werknemers met de functie\_id gelijk aan id in volgorde van veld |
|  | delete | api/functies/{id}/werknemers |  |  |  |

## Bereid het project voor: Maak nieuw Laravel project

|  |
| --- |
| C:> laravel new api\_personeel |
| Commando maakt een nieuw laravel project met de naam api\_personeel |

## Bereid het project voor: Maak nieuwe database

### Maak nieuwe MySql Database met de naam api\_personeel\_db

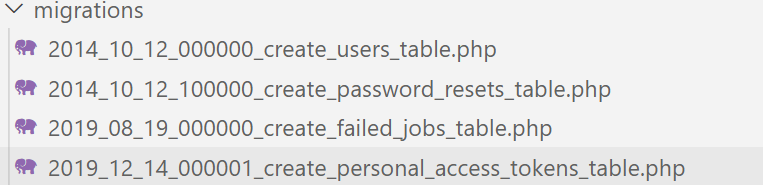
* Tip: gebruik als opslag engine innoDB
* Pas de .env file aan, zodat je applicatie de database api\_personeel\_db gebruikt

|  |
| --- |
| DB\_CONNECTION=mysql  DB\_HOST=127.0.0.1  DB\_PORT=3306  DB\_DATABASE=**api\_personeel\_db**  DB\_USERNAME=root  DB\_PASSWORD= |
| NB: Instellingen mogen afwijken. Het gaat hier om de naam van de database. Deze moet corresponderen met de naam van de MySql database |

### Maak de database migrations

#### Standaard

NB: In het laravel project krijg je alvast een aantal migrations cadeau:



Laat deze migrations staan; heb je nodig met API2.

#### Functies

|  |
| --- |
| php artisan make:migration functies |
| *Migration staat in: apipersoneel\database\migrations* |
| public function up()  {      Schema::create('functies', function (Blueprint $table) {          $table->tinyInteger('id')->autoIncrement();          $table->string('naam', 20)->nullable(false)->unique();      });  }  public function down()  {      Schema::dropIfExists('functies');  } |

#### Werknemers

|  |
| --- |
| php artisan make:migration werknemers |
| *Migration staat in: apipersoneel\database\migrations* |
| public function up()  {      Schema::create('werknemers', function (Blueprint $table) {          $table->tinyInteger('id')->autoIncrement();          $table->string('naam', 50)->nullable(false);          $table->tinyInteger('functie\_id');          $table->string('telefoon', 20)->nullable(true);          $table->string('email', 50)->nullable(true);          $table->date('sinds')->nullable(true);          $table->foreign('functie\_id')->references('id')->on('functies')->restrictOnDelete();      });  }  public function down()  {      Schema::dropIfExists('werknemers');  } |

## Bereid het project voor: Genereer models en controllers

### Functie

|  |
| --- |
| php artisan make:controller FunctieController --api --model=Functie |
| Hiermee maak je een leeg *Functie* model en een *FunctieController* met de standard functies voor een api. Verderop in het proces ga je deze completeren. |

### Werknemer

|  |
| --- |
| php artisan make:controller WerknemerController --api --model=Werknemer |
| Hiermee maak je een leeg *Werknemer* model en een *WerknemerController* met de standard functies voor een api. Verderop in het proces ga je deze completeren. |

## Routing: Maak de routing volgens API ontwerp

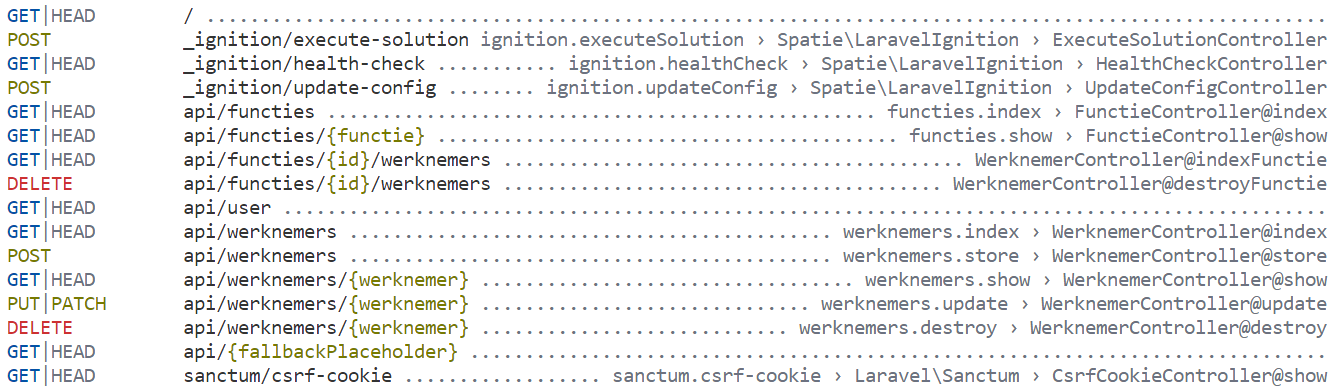
In de routing is al opgenomen welke action van welke controller de url */user* afhandeld. Deze kun je gerust laten staan, maar daar doen we in API1 niks mee.

Wel moet je voor alle overige urls uit het API ontwerp opnemen welke actions uit welke controllers, de urls worden afgehandeld. Je maakt hierbij maximaal gebruik van ed Rout::apiResource (standaard url’s) om typewerk te besparen.

NB: Als je buiten de standaard urls nog extra urls gebruikt, dan neem je deze in de lijst met routes ‘eerder’ op dan de standaard urls. Hiermee voorkom je dat een deze extra url via een standaard routing wordt verwerkt (veel voorkomende fout).

|  |
| --- |
| *Route toevoegen: staat in: apipersoneel\routes\api.php* |
| <?php  use Illuminate\Http\Request;  use Illuminate\Support\Facades\Route;  use App\Http\Controllers\FunctieController;  use App\Http\Controllers\WerknemerController;  Route::apiResource('werknemers', WerknemerController::class);  Route::get('functies/{id}/werknemers', [WerknemerController::class, 'indexFunctie']);  Route::delete('functies/{id}/werknemers', [WerknemerController::class, 'destroyFunctie']);  Route::apiResource('functies', FunctieController::class)->parameters(['functies' => 'functie'])                  ->only(['index', 'show']);  Route::fallback(function () {      return response()->json([          'message' => 'Page Not Found. If error persists, contact info@website.com '], 404);  }); |

Controleer de routes uit het ontwerp met het commando ***php artisan route:list***



## Models: Pas aan op basis van datamodel ontwerp

### Functie

|  |
| --- |
| *Model Functie aanpassen: staat in: apipersoneel\app\models* |
| <?php  namespace App\Models;  use Illuminate\Database\Eloquent\Factories\HasFactory;  use Illuminate\Database\Eloquent\Model;  class Functie extends Model  {      use HasFactory;      public $timestamps = false;      protected $fillable = ['naam'];      // protected $with = ['werknemers'];      public function werknemers()      {          return $this->hasMany(Werknemer::class, 'functie\_id', 'id');      }  } |

### Werknemer

|  |
| --- |
| *Model Werknemer aanpassen: staat in: apipersoneel\app\models* |
| <?php  namespace App\Models;  use Illuminate\Database\Eloquent\Factories\HasFactory;  use Illuminate\Database\Eloquent\Model;  class Werknemer extends Model  {      use HasFactory;      public $timestamps = false;      protected $fillable = ['naam', 'functie\_id', 'telefoon', 'email', 'sinds'];      public function functie()      {          return $this->belongsTo(Functie::class, 'functie\_id', 'id');      }  } |

## Controllers: Pas aan op basis van API ontwerp

### FunctieController

|  |
| --- |
| *Controller staat in: apipersoneel\app\http\controllers* |
| <?php  namespace App\Http\Controllers;  use App\Models\Functie;  use Illuminate\Http\Request;  class FunctieController extends Controller  { |
| public function index()      {          $response = [              'success' => true,              'data'    => Functie::All(),              // 'data'    => Functie::with('werknemers')->get(),          ];          return response()->json($response, 200);      } |
| * *NB: Commentaar op method (action) niveau is hier niet getoond, maar laat dit wel staan. Dit commentaar is verplicht tijdens je examen.* * *Door middel van width worden ook alle gerelateerde werknemers in de output gezet. Doordat je er commentaar van hebt gemaakt, gebeurt dat hier niet.* |
| public function show(Functie $functie)      {          $response = [              'success' => true,              'data'    =>  $functie,          ];          return response()->json($response, 200);        } |
| } |
| *Alleen de actions Index en Show zijn geïmplementeerd. Overige waren volgens de eisen en wensen niet nodig.*  *Database exception worden hier nog niet afgevangen. Treedt deze op, zorgt laravel voor een exception naar de applicate die de api gebruikt.Wil je een specifieke melding, dan kun je exceptions zelf afvangen, bijvoorbeeld via een try-catch of via de app\exceptions\handler.* |

### WerknemerController

|  |
| --- |
| *Controller staat in: apipersoneel\app\http\controllers* |
| <?php  namespace App\Http\Controllers;  use App\Models\Werknemer;  use Illuminate\Http\Request;  use Illuminate\Support\Facades\Log;  use Illuminate\Support\Facades\Validator;  class WerknemerController extends Controller  { |
| public function index(Request $request)      {          Log::info(              'werknemers index',              ['ip' => $request->ip(),              'data' => $request->all()          ]);          if ($request->has('naam')) {              $data = Werknemer::where('naam', 'like', '%' . $request->naam . '%')->get();          } else if ($request->has('sort')) {              $data =  Werknemer::orderBy($request->sort)->get();          } else {              $data = Werknemer::all();              // $data = Werknemer::with('functie')->get();          }          $content = [              'success' => true,              'data'    => $data,          ];          return response()->json($content, 200);      } |
| * *Aan de function is de parameter $request toegevoegd, zodat url-query’s (?naam = Stef) opgevraagd kunnen worden* * *Er wordt gelogd dat werknemergegevens zijn opgevraagd (niveau info)* |

|  |
| --- |
| public function store(Request $request)      {          Log::info(              'werknemers store',              [                  'ip' => $request->ip(),                  'data' => $request->all(),              ]          );          $validator = Validator::make($request->all(), [              'email' => 'email',              'naam' => 'required'          ]);          if ($validator->fails()) {              Log::error("werknemers toevoegen Fout");              $content = [                  'success' => false,                  'data'    => $request->all(),                  'foutmelding' => 'Data niet correct',              ];              return response()->json($content, 400);          } else {              $content = [                  'success' => true,                  'data'    => Werknemer::create($request                      ->only(['naam', 'functie\_id', 'telefoon', 'email', 'sinds'])),              ];              return response()->json($content, 201);          }      } |
| * *Zie commentaar bij Index* * *Er vindt validatie plaats op basis van de eisen uit het datamodel ontwerp.* * *Er wordt ook een foutmelding getoond als de aangeleverde data niet correct is.* |
| public function show(Request $request, Werknemer $werknemer)      {          Log::info(              'werknemers show',              [                  'ip' => $request->ip(),                  'data' => $request->all()              ]          );          $content = [              'success' => true,              'data'    => $werknemer,          ];          return response()->json($content, 200);      } |
| * *Request is nodig i.v.m. de logging* |
| public function update(Request $request, Werknemer $werknemer)      {          Log::info(              'werknemers update',              ['ip' => $request->ip(), 'oud' => $werknemer, 'nieuw' => $request->all()]          );          $validator = Validator::make($request->all(), [              'naam' => 'required',              'email' => 'email',          ]);          if ($validator->fails()) {              Log::error("werknemers wijzigen Fout");              $content = [                  'success' => false,                  'data'    => $request->all(),                  'foutmelding' => 'Gewijzigde data niet correct',              ];              return response()->json($content, 400);          } else {              $content = [                  'success' => $werknemer->update($request->all()),                  'data'    => $request->only(['naam', 'functie\_id', 'telefoon', 'email', 'sinds']),              ];              return response()->json($content, 200);          }      } |
| * *De update action is volgens de eisen en wensen niet nodig, maar hier toch toegevoegd want (zit ook in het API ontwerp)* |
| public function destroy(Request $request, Werknemer $werknemer)      {          Log::info(              'werknemers destroy',              ['ip' => $request->ip(), 'oud' => $werknemer]          );          $werknemer->delete();          $content = [              'success' => true,              'data'    => $werknemer,          ];          return response()->json($content, 202);      } |
| public function indexFunctie(Request $request, $id)      {          Log::info(              'werknemers indexFunctie',              [                  'ip' => $request->ip(),                  'data' => $request->all(),                  'id' => $id              ]          );          if ($request->has('sort')) {              $data =  Werknemer::where('functie\_id', $id)->orderBy($request->sort)->get();          } else {              $data = Werknemer::where('functie\_id', $id)->get();          }          $content = [              'success' => true,              'data'    => $data,          ];          return response()->json($content, 200);      } |
| public function destroyFunctie(Request $request, $id)      {          Log::info(              'werknemers destroyFunctie',              [                  'ip' => $request->ip(),                  'data' => $request->all(),                  'functie id' => $id              ]          );          Werknemer::where('functie\_id', $id)->delete();          $content = [              'success' => true,              'data'    => $id,          ];          return response()->json($content, 202);      } |
| } |

## Testen: Maak een seeder voor de database

### InitialDataSeeder

|  |
| --- |
| php artisan make:seeder InitialDataSeeder |
| <?php  namespace Database\Seeders;  use App\Models\Functie;  use App\Models\Werknemer;  use Illuminate\Database\Seeder;  class InitialDataSeeder extends Seeder  {      /\*\*       \* Run the database seeds.       \*/      public function run(): void      {         // Voeg test functies toe en bewaar deze in variabele zodat je er bij de  // testwerknemers aan kunt verwijzen.          $softwareEngineer = Functie::create(['naam' => 'software engineer',]);          $softwareArchitect = Functie::create(['naam' => 'software architect',]);          $tester = Functie::create(['naam' => 'tester',]);          Functie::create(['naam' => 'sales',]);          $manager = Functie::create(['naam' => 'manager',]);          $fieldEngineer = Functie::create(['naam' => 'field engineer',]);         // Voeg test werknemers toe. Maak gebruik van de testfuncties die je hiervoor in  // variabelen hebt bewaard. NB: De testwerknemers heb je hier niet meer nodig, dus  // je hoeft ze niet in een variabele te bewaren.          $jan = Werknemer::create([              'naam' => 'Jan',              'functie\_id' => $softwareEngineer->id,              'telefoon' => '0631874312', 'email' => 'jan@summa.nl', 'sinds' => '2002-11-01'          ]);            Werknemer::create([              'naam' => 'Erdinc',              'functie\_id' => $softwareEngineer->id,              'telefoon' => '0634981234', 'email' => 'erdinc@summa.nl', 'sinds' => '1989-11-01'          ]);            Werknemer::create([              'naam' => 'Carla',              'functie\_id' => $manager->id,              'telefoon' => '0634120975', 'email' => 'carla@summa.nl', 'sinds' => '1993-05-01'          ]);            Werknemer::create([              'naam' => 'Ellen',              'functie\_id' => $softwareEngineer->id,              'telefoon' => '0631451228', 'email' => 'elly@summa.nl', 'sinds' => '1995-08-15'          ]);            Werknemer::create([              'naam' => 'Vijay',              'functie\_id' => $softwareArchitect->id,              'telefoon' => '0631784466', 'email' => 'vijay@summa.nl', 'sinds' => '1997-08-01'          ]);            Werknemer::create([              'naam' => 'Piet',              'functie\_id' => $fieldEngineer->id,              'telefoon' => '0631552341', 'email' => 'piet@summa.nl', 'sinds' => '1999-07-01'          ]);            Werknemer::create([              'naam' => 'Juma',              'functie\_id' => $tester->id,              'telefoon' => '0631224550', 'email' => 'juma@summa.nl', 'sinds' => '2000-09-01'          ]);      }  } |
| *NB: De testdata wordt via de eerder gemaakte models aan de database toegevoegd.*  *In plaats van het toevoegen van de testdata via de models, kun je ook gebruik maken van DB. Bijvoorbeeld:*  DB::table('werknemers')->insert([              'naam' => 'Jan',              'functie\_id' => 1,              'telefoon' => '0631874312', 'email' => 'jan@summa.nl',  'sinds' => '2002-11-01'          ]); |

### DatabaseSeeder

|  |
| --- |
| Pas bestand DatabaseSeeder aan, zodat deze de *InitialDataSeeder* start |
| <?php  namespace Database\Seeders;  use Illuminate\Database\Seeder;  class DatabaseSeeder extends Seeder  {      /\*\*       \* Seed the application's database.       \*       \* @return void       \*/      public function run()      {          $this->call([             InitialDataSeeder::class,          ]);      }  } |

*NB: Alternatief is dat je een SQL script runt (niet uitgewerkt):*

|  |
| --- |
| <?php  namespace Database\Seeders;  use Illuminate\Database\Seeder;  class DatabaseSeeder extends Seeder  {      public function run()      {          exec("C:\\xampp\mysql\bin\mysql --user=root --password= --host=localhost --database personeel < ..\personeel.sql");          // sql-bestand zit 1 niveau dieper dan de projectmap          // C:\\ ipv C:\ omdat \x betekent iets      }  } |

### Maak de database opnieuw en vul deze met de testdata

|  |
| --- |
| C:> php artisan migrate:fresh --seed |
| * Dit commando gaat ervan uit dat je al een MySQL database hebt gemaakt, met de naam die in je .env bestand staat. * Het commando voer je elke keer uit, voordat je gaat testen. |

## Testen: Maak de featuretests voor de API’s

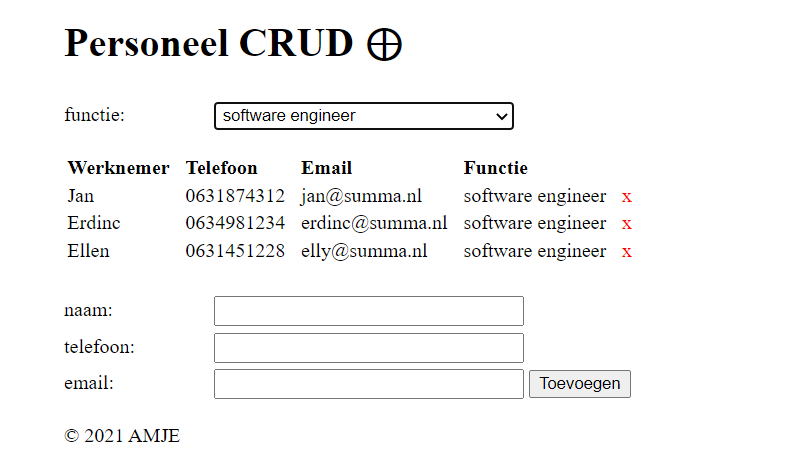
### PersoneelTest

|  |
| --- |
| php artisan make:test PersoneelTest |
| *Feature test staat in test/feauture* |
| <?php  namespace Tests\Feature;  use Illuminate\Foundation\Testing\RefreshDatabase;  use Illuminate\Foundation\Testing\WithFaker;  use Tests\TestCase;  class PersoneelTest extends TestCase  {      /\*\*       \* Test dat api/werknemers/{id} werkt (GET)       \* @return void       \*/      public function test\_werknemer\_op\_id()      {          $response = $this->get('api/werknemers/1');          $response              ->assertStatus(200)              ->assertJsonPath('success', true)              ->assertJsonPath('data.naam', 'Jan')              ->assertJsonPath('data.email', 'jan@summa.nl');      }      /\*\*       \* Test dat api/werknemers?naam={naam} (GET)       \* @return void       \*/      public function test\_werknemer\_op\_naam()      {          $response = $this->get('api/werknemers?naam=J');          $response              ->assertStatus(200)              ->assertJsonCount(3, 'data')              ->assertJsonPath('data.0.naam', 'Jan')              ->assertJsonPath('data.0.email', 'jan@summa.nl')              ->assertJsonFragment(                  ['naam' => 'Vijay', 'email' => 'vijay@summa.nl',]              );      }      /\*\*       \* Test dat api/werknemers werkt, om werknemer toe te voegen (POST)       \* @return void       \*/      public function test\_insert\_werknemer()      {          $data = ['naam' => 'Karel', 'email' => 'karel@summa.nl', 'functie\_id' => 1];          $response = $this->json('POST', 'api/werknemers', $data);          $this->assertDatabaseHas(              'werknemers', ['naam' => 'Karel', 'email' => 'karel@summa.nl',]          );          $response              ->assertStatus(201)              ->assertJsonPath('data.naam', 'Karel')              ->assertJsonPath('data.email', 'karel@summa.nl');      }      /\*\*       \* Test dat api/werknemers werkt, om werknemer toe te voegen (POST)       \* @return void       \*/      public function test\_delete\_werknemer()      {          $response = $this->delete('api/werknemers/7');          $response->assertStatus(202);      }  } |
| Uitvoeren van de test: php artisan test  NB: Wil je opnieuw testen, ververs dan eerst je database :  php artisan migrate:fresh --seed |

## Maak de front-end applicatie

### Beschrijving

De front-end applicatie is gemaakt met HTML/CSS en JavaScript. Hier zetten we alle code in 1 bestand: ***personeelCRUD+.html****.* In werkelijkheid doe jij dat natuurlijk netter (verplicht html, css en javascript in aparte bestanden).



De combobox is een <select>-tag.

De werknemers staan in een tabel <table>

De gegevens van een nieuwe werknemers staan in een aantal <input>-tags

Zodra het programma is geladen:

* worden eenmalig alle functies uit de database ingelezen. Hierbij wordt de api *api/functies* gebruikt. De opgehaalde functies:
  + worden in de hulptabel gezet, zodat ze later gebruikt kunnen worden om de naam van een functie te tonen bij een werknemers (in de tabel met werknemers staat de naam van de functie).
  + Worden omsloten door een <option>-tag in de <select> tag gezet.
* Worden alle werknemers waarvan de functie\_id de waarde heeft die in de combobox is geselecteerd. Hierbij wordt de api *api/functies/{id}/werknemers* gebruikt. Van de medewerkers worden de naam, telefoon, email getoond, en de omschrijving van functie. De omschrijving wordt uit opgehaald uit de hulptabel. Er wordt hier van een hulptabl gebruik gemaakt, om te voorkomen dat er onnodig veel api-calls uitgevoerd moeten worden.

Zodra een gebruiker in de combobox een andere functie kiest, wordt de tabel met werknemers leeggemaakt en opnieuw gevuld: alle werknemers waarvan de functie\_id de waarde heeft die in de combobox is geselecteerd. Dit gebeurd, zoals hiervoor beschreven.

Wordt op de knop Toevoegen gedrukt, dan wordt een nieuwe medewerker toegevoegd. Hierbij wordt de api *api/werknemers* gebruikt i.c.m. met een POST method. De gegevens van de werknemer staan in de body van het request: naam, telefoon, email uit het input-tag, functie\_id uit de geselecteerde functie van de combobox.

### Code

|  |
| --- |
| <!DOCTYPE html>  <html lang="nl" xmlns:mso="urn:schemas-microsoft-com:office:office"      xmlns:msdt="uuid:C2F41010-65B3-11d1-A29F-00AA00C14882">  <head>      <title> Personeel CRUD+</title>      <meta charset="utf-8" />      <style>          div,          footer,          form { margin-top: 20px; }          input,          select { margin-top: 5px; width: 40%; padding: 2px; }          body { width: 80%; max-width: 600px; margin: auto; margin-top: 50px; }          th,          td { text-align: left; padding-right: 10px; }          div { width: 100%; }          label { width: 20%; display: inline-block; }          td:last-child { color: red; cursor: pointer; }      </style>      <script src="https://unpkg.com/axios/dist/axios.min.js"></script>  </head> |
| Style is niet in een apart bestand gezet. Dat zou wel moeten!  Voor de api-calls wordt gebruik gemaakt van axios. |
| <body onload="laad()">      <h1>Personeel CRUD &oplus;</h1>      <div>          <label for="functie">functie:</label>  <select id="functie" onclick="laadFunctieWerknemers()"></select>      </div>      <br />      <table>          <tr>              <th>Werknemer</th>              <th>Telefoon</th>              <th>Email</th>              <th>Functie</th>          </tr>          <tbody id="tabelInhoud" />      </table>      <div>          <label for="naam">naam:</label><input type="text" id="naam" required /><br />          <label for="telefoon">telefoon:</label><input type="text" id="telefoon" /><br />          <label for="email">email:</label><input type="email" id="email" />          <button onclick="voegToe()" type="button">Toevoegen</button>      </div>      <footer>&copy; 2021 AMJE</footer>  </body> |
| * Laad zorgt ervoor dat de applicatie bij het starten gegevens toont. * De Id van de combobox is functie. Deze id heeft de function *laadFunctieWerknemers* nodig om de waarde van de geselecteerde <option>-tags op te halen. * De functie *laadFunctieWerknemers* leest alle werknemers waarvan de functie\_id gelijk is aan de id uit de combobox. * De body van de tabel heeft als id *tabelInhoud* . Deze id heeft *laadFunctieWerknemers* om medewerkers in te zetten. * De id's functie, naam, telefoon en email heeft de function *voegToe* nodig om de gegevens van de toe te voegen werknemer samen te stellen * De function *voegToe* voegt een nieuwe werknemer toe |
| <script>      "use strict"      const apiBasis = "http://127.0.0.1:8000/api/"      const apiWerknemers = apiBasis + "werknemers"      const apiFuncties = apiBasis + "functies"      let functies = []      let werknemers = [] |
| * Functies bevat een array waarin alle namen van functies uit de database worden gezet. Van functie 4 wordt de naam gezet in functies[4]. Bij de werknemer staat zijn functie in de kolom *functie\_id*. Als je de naam van zijn functie wilt tonen, staat deze in *functies[functie\_id]*. * Werknemers bevat alle werknemers van de geselecteerde functie. |
| const laadFuncties = async () => {          const response = await axios.get(apiFuncties)          if (response.status != 200) {              alert(`Fout bij het lezen van de functies: ${response.status} (${response.statusText})`)              return          }          const json = await response.data          let selectInhoud = ''          json.data.map(el => {              functies[el.id] = el.naam              selectInhoud += `<option value="${el.id}">${el.naam}</option>`          })          document.querySelector("#functie").innerHTML = selectInhoud      } |
| De function *laadFuncties* leest met een api alle functies in en zet deze in de tabel met functies.  Het resultaat bevat een JSON tabel waarvan de rijen in de tabel *functies* worden gezet en die bovendien in een <option>-tag worden. Alle <option>-tags worden in de combobox (<select>) met id *functie* gezet |
| const laadFunctieWerknemers = async () => {          const functie  = document.querySelector("#functie").value          const apiFunctiesWerknemers = `${apiFuncties}/${functie}/werknemers?sort=naam`          const response = await axios.get(apiFunctiesWerknemers)          if (response.status != 200){              alert(  `Fout bij het lezen van de werknemers: ${response.status} (${response.statusText})`  )              return          }          const json = await response.data          werknemers = json.data          toon()      } |
| De function *laadFunctieWerknemers* leest alle werknemers met een bepaalde functie in en sorteert ze op naam. Hiervoor wordt gebruik gemaakt van een api.  Het resultaat bevat een JSON tabel die in de variabele json wordt gezet. Daarna worden de werknemers met de function *toon* in weergegeven |
| const toon = () => {          let tabelInhoud = ''          werknemers.map(el => tabelInhoud += `<tr><td>${el.naam}</td><td>${el.telefoon}</td><td>${el.email}</td>                          <td>${functies[el.functie\_id]}</td><td onclick="verwijder(${el.id})"> x </td></tr>`)          document.querySelector("#tabelInhoud").innerHTML = tabelInhoud      } |
| De function *toon*, vult een string met tabelrijen (<tr>-tags). Elke tabelrij bevat gegevens van een werknemer (*el*) en een kolom x.  Deze kolom x heeft een function *verwijder* die uitgevoerd wordt als er op geclickt wordt (*onclick*). In *el.id* staat de id van de werknemer die verwijderd wordt.  De string wordt tenslotte toegevoegd aan de tabel met id *tabelinhoud*. |
| const laad = async () => {          await laadFuncties()          await laadFunctieWerknemers()      } |
| De function laad wordt via onload gestart  Eerst worden alle functies ingelezen met de function laadFuncties. Daarna worden de werknemers ingelezen die dezelfde functie hebben als de functie die in de combobox is geselecteerd. |
| const voegToe = async () => {          var werknemer = {};          werknemer.naam = document.querySelector("#naam").value;          werknemer.telefoon = document.querySelector("#telefoon").value;          werknemer.email = document.querySelector("#email").value;          werknemer.functie\_id = document.querySelector("#functie").value;          const response = await axios.post(apiWerknemers, werknemer, {              headers: {                  'Content-Type': 'application/json',                  'Accept': 'application/json'  }          })          if (response.status != 201) {              alert(  `Fout bij het toevoegen van de werknemer: ${response.status} (${response.statusText})`  )              return          }          const json = await response.data          document.querySelector("#naam").value = ''          document.querySelector("#telefoon").value = ''          document.querySelector("#email").value = ''          await laadFunctieWerknemers()      } |
| De function *voegtoe* voegt met een POST method een nieuwe werknemer toe. Eerst wordt een JSON gemaakt met de gegevens van de werknemer. Deze JSON wordt daarna als argument aan de api doorgegespeeld. Nadat de werknemer is toegevoegd, worden de ingevulde gegevens op het scherm leeg gemaakt en wordt de tabel met werknemers opnieuw gevuld met de function *laadFunctieWerknemers*. Als het goed is, is de toegevoegde werknemer dan te zien. |
| const verwijder = async (id) => {          const apiDelete = `${apiWerknemers}/${id}`          const response = await axios.delete(apiDelete, {              headers: {                  'Content-Type': 'application/json',                  'Accept': 'application/json'              }          })          if (response.status != 202) {              alert(  `Fout bij het verwijderen van werknemers: ${response.status} (${response.statusText})`  )          }          const json = await response.data          werknemers = await response.data          await laadFunctieWerknemers()      } |
| De function *verwijder* verwijdert met een DELETE method de werknemer waarvan de id in de parameter staat. Na het verwijderen wordt de function *laadFunctieWerknemers* uitgevoerd, zodat de tabel met werknemers wordt bijgewerkt: als het goed is, is de verwijderde werknemer dan niet meer zichtbaar. |
| </script>  </html> |

## Testen: Maak een testplan

### Inleiding

Beschrijf voor elke functionaliteit uit de MoSCoW alle mogelijke scenario’s. Geef per scenario aan wat je als resultaat verwacht. Schrijf vervolgens per scenario op wat het werkelijke resultaat is.

NB: Hier zijn alleen de testgevallen voor de front end opgenomen. Vergelijkbaar moet je ook een dergelijk testplan maken voor de API’s (backend) zelf.

### Testrapport

#### Opvragen medewerkers met een functie

##### Scenario: Geen backend

|  |  |
| --- | --- |
| Scenario | Backend niet beschikbaar |
| Verwacht resultaat | Melding waarschuw Service desk |
| Werkelijk resultaat | Console laat zien dat er een fout is opgetreden |
| Acties | Student lost het op! |

##### Scenario: Geen database

|  |  |
| --- | --- |
| Scenario | Backend beschikbaar  Database niet beschikbaar |
| Verwacht resultaat | Melding waarschuw Service desk |
| Werkelijk resultaat | Console laat zien dat er een fout is opgetreden |
| Acties | Student lost het op! |

##### Scenario: Selecteer Medewerkers met functie Engineer

|  |  |
| --- | --- |
| Scenario | Backend beschikbaar  Database beschikbaar  Selecteer de functie Engineer |
| Verwacht resultaat | Alle medewerkers met de functie Engineer: |
| Werkelijk resultaat | Zoals verwacht |
| Acties | n.v.t. |

#### Toevoegen medewerkers

##### Scenario: Geen backend

|  |  |
| --- | --- |
| Scenario | Backend niet beschikbaar |
| Verwacht resultaat | Melding waarschuw Service desk |
| Werkelijk resultaat | Console laat zien dat er een fout is opgetreden |
| Acties | Student lost het op! |

##### Scenario: Geen database

|  |  |
| --- | --- |
| Scenario | Backend beschikbaar  Database niet beschikbaar |
| Verwacht resultaat | Melding waarschuw Service desk |
| Werkelijk resultaat | Console laat zien dat er een fout is opgetreden |
| Acties | Student lost het op! |

##### Scenario: Ongeldige naam en email (Naam en email leeg)

|  |  |
| --- | --- |
| Scenario | Backend beschikbaar  Database beschikbaar  Gebruiker laat naam en email leeg. |
| Verwacht resultaat | Medewerker wordt niet toegevoegd.  Er volgt een melding. |
| Werkelijk resultaat | Medewerker wordt niet toegevoegd.  Er volgt geen melding. In plaats daarvan in de console log een melding. |
| Acties | Student lost het op! |

##### Scenario: Ongeldige naam en email (Naam leeg, en email ongeldig)

|  |  |
| --- | --- |
| Scenario | Backend beschikbaar  Database beschikbaar  Gebruiker laat naam en vult ongeldig email in. |
| Verwacht resultaat | Medewerker wordt niet toegevoegd.  Er volgt een melding. |
| Werkelijk resultaat | Medewerker wordt niet toegevoegd.  Er volgt geen melding. In plaats daarvan in de console log een melding. |
| Acties | Student lost het op! |

##### Scenario: Ongeldige naam en email (Naam leeg, en geldig email)

|  |  |
| --- | --- |
| Scenario | Backend beschikbaar  Database beschikbaar  Gebruiker laat naam en vult geldig email in. |
| Verwacht resultaat | Medewerker wordt niet toegevoegd.  Er volgt een melding. |
| Werkelijk resultaat | Medewerker wordt niet toegevoegd.  Er volgt geen melding. In plaats daarvan in de console log een melding. |
| Acties | Student lost het op! |

##### Scenario: Ongeldige naam en email (Naam geldig, en email leeg)

|  |  |
| --- | --- |
| Scenario | Backend beschikbaar  Database beschikbaar  Gebruiker vult naam in en laat email leeg. |
| Verwacht resultaat | Medewerker wordt niet toegevoegd.  Er volgt een melding. |
| Werkelijk resultaat | Medewerker wordt niet toegevoegd.  Er volgt geen melding. In plaats daarvan in de console log een melding. |
| Acties | Student lost het op! |

##### Scenario: Ongeldige naam en email (Naam geldig, en email ongeldig)

|  |  |
| --- | --- |
| Scenario | Backend beschikbaar  Database beschikbaar  Gebruiker vult naam in en vult ongeldig email in. |
| Verwacht resultaat | Medewerker wordt niet toegevoegd.  Er volgt een melding. |
| Werkelijk resultaat | Medewerker wordt niet toegevoegd.  Er volgt geen melding. In plaats daarvan in de console log een melding. |
| Acties | Student lost het op! |

##### Scenario: Ongeldige naam en email (Naam geldig, en email geldig)

|  |  |
| --- | --- |
| Scenario | Backend beschikbaar  Database beschikbaar  Gebruiker vult naam en geldig email in. |
| Verwacht resultaat | Gebruiker wordt toegevoegd.  Scherm wordt bijgewerkt. |
| Werkelijk resultaat | Volgens verwacht resultaat. |
| Acties | n.v.t. |

#### Verwijderen medewerkers

##### Scenario: Geen backend

|  |  |
| --- | --- |
| Scenario | Backend niet beschikbaar  Gebruiker clickt op x achter medewerker. |
| Verwacht resultaat | Melding contact met service desk |
| Werkelijk resultaat | Er volgt geen melding |
| Acties | Student past aan! |

##### Scenario: Geen database

|  |  |
| --- | --- |
| Scenario | Backend beschikbaar  Database niet beschikbaar  Gebruiker clickt op x achter medewerker. |
| Verwacht resultaat | Melding contact met service desk |
| Werkelijk resultaat | Er volgt geen melding |
| Acties | Student past aan! |

##### Scenario: Verwijder medewerker

|  |  |
| --- | --- |
| Scenario | Backend beschikbaar  Database beschikbaar  Gebruiker clickt op x achter medewerker. |
| Verwacht resultaat | Medewerker wordt verwijderd.  Scherm wordt bijgewerkt. |
| Werkelijk resultaat | Volgens verwacht resultaat. |
| Acties | n.v.t. |

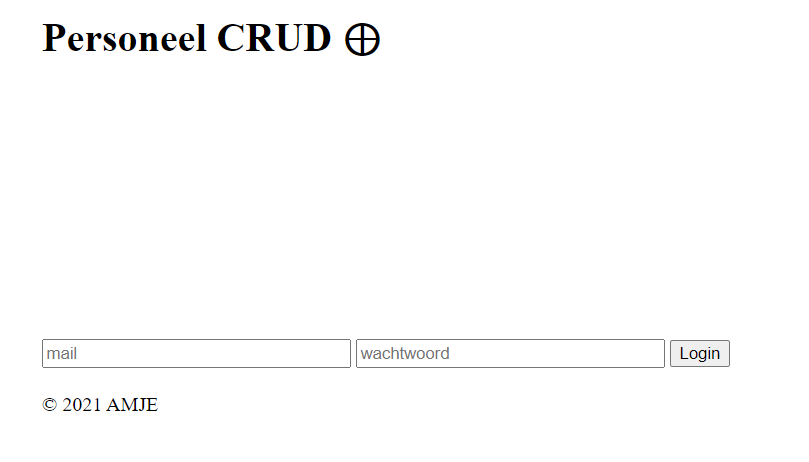
## Lever applicatie op

Laat testrapport zien: alle issues zijn opelost. Geef Demonstratie en uitleg code.

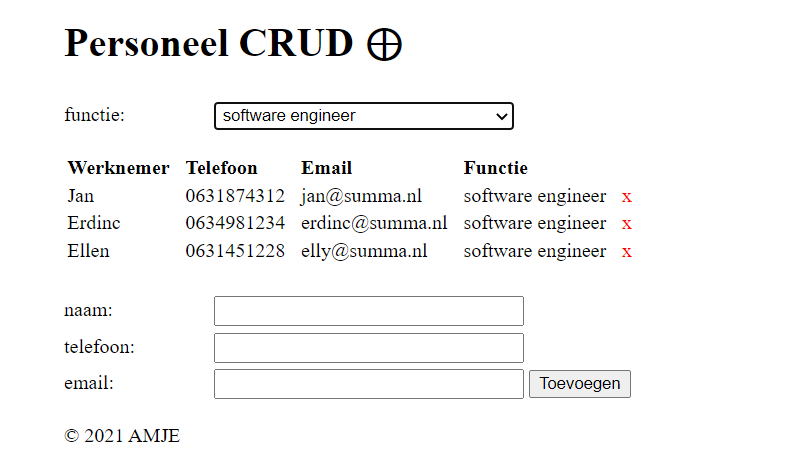
# API2: Handboek

## Inleiding

De applicatie moet worden beveiligd met behulp van een bearer token. Het scherm van de applicatie wijzigt niet, maar er wordt wel een inlog scherm toegevoegd. Dit scherm wordt bij het starten van de applicatie getoond, zodat de gebruiker zich kan autenticeren met email en password. Het zou mooi zijn als er ook een registratiepagine komt.



Zodra de gebruiker is ingelogd, wordt de applicatie informatie getoond (zie API1)



## Planning

In principe ga je alle stappen die je met API1 maakte opnieuw doorlopen. Onderzoek wat je moet doen en plan de tijd. Hieronder alle stappen, inclusief de stappen die na grondige analyse onnodig bleken (zijn doorgehaald).

|  |  |  |  |
| --- | --- | --- | --- |
| Actie | Datum | Planning | Werkelijk |
| 1. Update ontwerp: Eisen en Wensen | 2-mei | 1:00u | 0:30u |
| 1. Update ontwerp: Use Case diagram  * Gebruiker moet zich autenticeren voordat hij inzage heeft in de applicatie data | 2-mei | 0:30u | 1:00u |
| 1. ~~Update ontwerp: Datamodel~~  * Niet van toepassing. Alle tabellen zijn al aanwezig, mede dankzij het gebruik van Sanctum (standaard laravel) tool om de api’s te beveiligen. |  |  |  |
| 1. Update ontwerp: Api’s | 2-mei | 0:30u | 1:00u |
| 1. ~~Bereid het project voor: Maak nieuw Laravel project~~  * Een bestaande applicatie wordt aangepast. |  |  |  |
| 1. ~~Update Migrations voor de bestaande database~~  * Migrations voor de tabellen die Sanctum gebruikt, zijn reeds aanwezig (standaard laravel) |  |  |  |
| 1. Bereid het project voor: Maak AuthenticationController   Ten behoeve van het registeren, inloggen en uitloggen zal er een aparte controller worden toegevoegd: |  |  |  |
| 1. Update Routing: Maak de routing volgens API ontwerp  * Zorg dat de API’s voor het beheer van de medewerkergegevens beveiligd zijn | 2-mei | 1:30u | 3:00u |
| 1. ~~Update Models~~  * Models hoeven niet bijgewerkt te worden |  |  |  |
| 1. Update Controllers  * Voeg beveiliging toe aan Create/Update/Delete functie van de medewerkers. * Maak controllers om te registeren, in te loggen en uit te loggem | 2-mei | 1:00u | 1:00u |
| 1. ~~Testen: Update seeder voor de database~~  * Seeders hoeven niet bijgewerkt te worden |  |  |  |
| 1. Testen: Update featuretests voor de API’s  * Testen voor de werknemers moeten uitgebreid worden met de beveiligingseisen | 2-mei | 1:30u | 1:20u |
| 1. Update Front-End applicatie  * Implementeer inlogscherm * Implementeer beveiligde api’s | 2-mei | 2:00u | 1:50u |
| 1. Testen: Update testplan en voer het uit | 2-mei | 1:30u | 1:20u |
| 1. Borrel: Lever applicatie op | 8-mei | 0:10u |  |
| Totaal |  | 10:00u | 11:10u |

## Update ontwerp: Eisen en Wensen

**Must have**

* De front-end wisselt gegevens uit met de backend via Restful API’s.
* De combobox bevat alle functies die werknemers kunnen vervullen. Functies zijn uitsluitend raadpleegbaar (er is dus geen beheer op mogelijk).
* Het overzicht toont alle werknemers die de in de combobox geselecteerde functie uitoefenen.
* Door middel van de ‘x’ kan de gebruiker de betreffende werknemer uit de administratie verwijderen.
* Door middel van de knop ‘toevoegen’ kan de gebruiker een nieuwe werknemer aan de administratie toevoegen.
* API’s ten behoeve van het toevoegen, raadplegen, verwijderen en wijzigen van de werknemers zijn beveiligd door middel van een rotating bearer token.
* API’s ten behoeve van het raadplegen van de functies zijn niet beveiligd. Iedereen mag alle de api om functies te raadplegen opvragen. De api’s mogen nooit werknemergegevens tonen.

**Shoud have**

* De gebruiker die toegang tot de applicatie moet hebben, moet zich door middel van een email en password autenticeren.
* De service desk moet een gebruiker kunnen registreren (postman). Vooralsnog zal de leidinggevende, procedureel, een email en wachtwoord aan de gebruiker beschikbaar stellen. Gebruikers kunnen zich dus niet zelf registreren via een scherm.

**Could have**

* De gebruiker moet zichzelf kunnen uitloggen.

**Won't have**

## Update ontwerp: Use Case diagram

## Update ontwerp: Api’s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | methode | url | json parameters | header | json resultaat |
| M | get | api/werknemers/ |  | **Access token** | Alle werknemers |
| M | post | api/werknemers/ | naam, telefoon, email, functie\_id | **Access token** | gecreëerde werknemer |
| M | get | api/werknemers/{id} |  | **Access token** | werknemer met opgegeven id |
| W | patch | api/werknemers/{id} | naam en/of telefoon en/of email en/of functie\_id | **Access token** | werknemer met opgegeven id |
| M | delete | api/werknemers/{id} |  | **Access token** |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | methode | url | json parameters | header | json resultaat |
| M | get | api/functies/ |  |  | Alle functies |
| M | Get | api/functies/{id} |  |  | functie met opgegeven id |

Nb: Noodzaak voor ‘api/functies/{id}’ is afhankelijk van de technische implementatie.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | methode | url | json parameters | header | json resultaat |
|  | get | api/werknemers?naam={naam} |  | **Access token** | Alle werknemers met de tekst naam in de naam |
|  | get | api/werknemers?sort={veld} |  | **Access token** | Alle werknemers in volgorde van veld |
|  | get | api/functies/{id}/werknemers |  | **Access token** | Werknemers met de functie\_id gelijk aan id |
|  | get | api/functies/{id}/werknemers?sort={veld} |  | **Access token** | Werknemers met de functie\_id gelijk aan id in volgorde van veld |
|  | delete | api/functies/{id}/werknemers |  | **Access token** |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | methode | url | json parameters | header | json resultaat |
| **S** | **post** | api/login/ | {  "email" : "<<email>>",  "password" : "<<password>>"  } |  | {  "access\_token : "<<token>>",    "token\_type" :  "Bearer"  } |
| **S** | **post** | **api/register/** | {  "name": "<<name>>",  "email": "<<email>>",  "password": "<<password>>",  "password\_confirmation"  : "<<password>>"  } |  | {  "message": "<<message>>"  } |
| **C** | **post** | **api/logout/** |  | **Access token** | {  "message": "<<message>>"  } |

## Bereid het project voor: Maak AuthenticationController

De AuthenticationController bevat action om nieuwe gebruikers te registeren, bestaande gebruikers in te loggen en ingelogde gebruikers uit te loggen. Zie ook: <https://laravel.com/docs/10.x/sanctum>.

### AuthenticationController

|  |
| --- |
| php artisan make:controller AuthenticationController |
| *Controller staat in: apipersoneel\app\http\controllers* |
| <?php  namespace App\Http\Controllers;  use Illuminate\Support\Facades\Auth;  use Illuminate\Support\Facades\Log;  use Illuminate\Http\Request;  use App\Models\User;  class AuthenticationController extends Controller  { |
| public function register(Request $request)      {          Log::info('register', ['ip' => $request->ip(), 'data' => $request->all()]);          $attr = $request->validate([              'name' => 'required|string|max:255',              'email' => 'required|string|email|unique:users,email',              'password' => 'required|string|min:6|confirmed'          ]);          $user = User::create([              'name' => $attr['name'],              'password' => bcrypt($attr['password']),              'email' => $attr['email']          ]);          return response()->json(['message' => 'Registration successful'], 200);      } |
| * *Register zorgt ervoor dat de gebruiker zich kan registreren.* * *De registratie wordt gelogd: informatief, met IP adres en alle data.* * *NB: De LOG logt hier alle data, ook het password van de gebruiker (beveiligingsissue)* |
| public function login(Request $request)      {          Log::info('login', ['ip' => $request->ip(), 'data' => $request->all()]);          $attr = $request->validate([              'email' => 'required|string|email|',              'password' => 'required|string|min:6'          ]);          if (!Auth::attempt($attr)) {              return $this->error('Credentials not match', 401);          }          $response = [              // 'access\_token' => $request->bearerToken(),              'access\_token' => auth()->user()->createToken('API Token')->plainTextToken,              'token\_type' => 'Bearer'          ];          return response()->json($response, 200);      } |
| * *Login zorgt ervoor dat de gebruiker kan inloggen en een nieuw bearer token krijgt.* * *De credentials worden eerst gevalideerd.* * *Zijn ze goed, dan wordt de gebruiker geauthentiseerd met Auth.* * *Is de gebruiker geauthentiseerd, dan wordt een nieuw token geretourneerd (huidige token staat in $request->bearerToken()).* * *NB: De LOG logt hier alle data, ook het password van de gebruiker (beveiligingsissue)* |
| public function logout(Request $request)      {          Log::info('logout', ['ip' => $request->ip(), 'data' => $request->all(), 'user' => auth()->user()]);          auth()->user()->tokens()->delete();          return [              'message' => 'Tokens Revoked'          ];      } |
| * *Logout verwijdert alle tokens van de gebruiker.* * *NB: De LOG logt hier alle data, ook het password van de gebruiker (beveiligingsissue)* |
| } |

## Update Routing: Maak de routing volgens API ontwerp

|  |
| --- |
| *Route toevoegen: staat in: apipersoneel\routes\api.php* |
| <?php  use Illuminate\Http\Request;  use Illuminate\Support\Facades\Route;  use App\Http\Controllers\FunctieController;  use App\Http\Controllers\WerknemerController;  use App\Http\Controllers\AuthenticationController;  Route::post('/register', [AuthenticationController::class, 'register']);  Route::post('/login', [AuthenticationController::class, 'login']);  Route::apiResource('functies', FunctieController::class)  ->parameters(['functies' => 'functie'])  ->only(['index', 'show']);  Route::group(['middleware' => ['auth:sanctum']], function () {      Route::get('profile', function (Request $request) {          return auth()->user();      });      Route::apiResource('werknemers', WerknemerController::class);      Route::get('functies/{id}/werknemers', [WerknemerController::class, 'indexFunctie']);      Route::delete('functies/{id}/werknemers', [WerknemerController::class, 'destroyFunctie']);      Route::post('/logout', [AuthenticationController::class, 'logout']);  });  Route::fallback(function () {      return response()->json([          'message' => 'Page Not Found. If error persists, contact info@website.com '], 404);  }); |

## Update Controllers

### Pas de FunctieController aan

Functiecontroller kan ongeautoriseerd worden gebruikt. Aanpassingen zijn niet nodig.

### Pas de WerknemerController aan

De methods dienen als volgt aangepast te worden:

* Initieel wordt de actuele token waarmee de gebruiker toegang tot de api kreeg verwijderd.
* Aan het resultaat van de api (de retunwaarde) wordt een nieuw access token voor de gebruiker toegevoegd.

|  |
| --- |
| *Controller staat in: apipersoneel\app\http\controllers* |
| public function index(Request $request)      {          $request->user()->currentAccessToken()->delete();    // Verwijder de actuele token          Log::info(              'werknemers index',              [                  'ip' => $request->ip(),                  'data' => $request->all()              ]          );          if ($request->has('naam')) {              $data = Werknemer::where('naam', 'like', '%' . $request->naam . '%')->get();          } else if ($request->has('sort')) {              $data =  Werknemer::orderBy($request->sort)->get();          } else {              $data = Werknemer::all();              // $data = Werknemer::with('functie')->get();          }          $content = [              'success' => true,              'data'    => $data,              'access\_token' => auth()->user()->createToken('API Token')->plainTextToken,              'token\_type' => 'Bearer',          ];          return response()->json($content, 200);      } |
| * *Omdat je steeds met een nieuwe token wilt werken, begin je met het verwijderen van de het actuele token. Doe je dat niet, blijft het ‘oude’ token nog steeds geldig, ook al heb je een nieuw token gemaakt.* |

|  |
| --- |
| public function store(Request $request)      {          $request->user()->currentAccessToken()->delete();    // Verwijder de actuele token          Log::info(              'werknemers store',              [                  'ip' => $request->ip(),                  'data' => $request->all(),              ]          );          $validator = Validator::make($request->all(), [              'email' => 'email',              'naam' => 'required'          ]);          if ($validator->fails()) {              Log::error("werknemers toevoegen Fout");              $content = [                  'success' => false,                  'data'    => $request->all(),                  'foutmelding' => 'Data niet correct',                  'access\_token' => auth()->user()->createToken('API Token')->plainTextToken,                  'token\_type' => 'Bearer',              ];              return response()->json($content, 400);          } else {              $content = [                  'success' => true,                  'data'    => Werknemer::create($request                      ->only(['naam', 'functie\_id', 'telefoon', 'email', 'sinds'])),                  'access\_token' => auth()->user()->createToken('API Token')->plainTextToken,                  'token\_type' => 'Bearer',              ];              return response()->json($content, 201);          }      } |
| * *Zie commentaar bij Index* * *Hier wordt ook een foutmelding getoond als de aangeleverde data niet correct is. Ook dan moet je een nieuwe token teruggeven.* |
| public function show(Request $request, Werknemer $werknemer)      {          $request->user()->currentAccessToken()->delete();    // Verwijder de actuele token          Log::info(              'werknemers show',              [                  'ip' => $request->ip(),                  'data' => $request->all()              ]          );          $content = [              'success' => true,              'data'    => $werknemer,              'access\_token' => auth()->user()->createToken('API Token')->plainTextToken,              'token\_type' => 'Bearer',          ];          return response()->json($content, 200);      } |
| public function update(Request $request, Werknemer $werknemer)      {          $request->user()->currentAccessToken()->delete();    // Verwijder de actuele token          Log::info(              'werknemers update',              ['ip' => $request->ip(), 'oud' => $werknemer, 'nieuw' => $request->all()]          );          $validator = Validator::make($request->all(), [              'naam' => 'required',              'email' => 'email',          ]);          if ($validator->fails()) {              Log::error("werknemers wijzigen Fout");              $content = [                  'success' => false,                  'data'    => $request->all(),                  'foutmelding' => 'Gewijzigde data niet correct',                  'access\_token' => auth()->user()->createToken('API Token')->plainTextToken,                  'token\_type' => 'Bearer',              ];              return response()->json($content, 400);          } else {              $content = [                  'success' => $werknemer->update($request->all()),                  'data'    => $request->only(['naam', 'functie\_id', 'telefoon', 'email', 'sinds']),                  'access\_token' => auth()->user()->createToken('API Token')->plainTextToken,                  'token\_type' => 'Bearer',              ];              return response()->json($content, 200);          }      } |
| * *De update action is volgens de eisen en wensen niet nodig, maar hier toch toegevoegd* |
| public function destroy(Request $request, Werknemer $werknemer)      {          $request->user()->currentAccessToken()->delete();    // Verwijder de actuele token          Log::info(              'werknemers destroy',              ['ip' => $request->ip(), 'oud' => $werknemer]          );          $werknemer->delete();          $content = [              'success' => true,              'data'    => $werknemer,              'access\_token' => auth()->user()->createToken('API Token')->plainTextToken,              'token\_type' => 'Bearer',          ];          return response()->json($content, 202);      } |
| public function indexFunctie(Request $request, $id)      {          $request->user()->currentAccessToken()->delete();    // Verwijder de actuele token          Log::info(              'werknemers indexFunctie',              [                  'ip' => $request->ip(),                  'data' => $request->all(),                  'id' => $id              ]          );          if ($request->has('sort')) {              $data =  Werknemer::where('functie\_id', $id)->orderBy($request->sort)->get();          } else {              $data = Werknemer::where('functie\_id', $id)->get();          }          $content = [              'success' => true,              'data'    => $data,              'access\_token' => auth()->user()->createToken('API Token')->plainTextToken,              'token\_type' => 'Bearer',          ];          return response()->json($content, 200);      } |
| public function destroyFunctie(Request $request, $id)      {          $request->user()->currentAccessToken()->delete();    // Verwijder de actuele token          Log::info(              'werknemers destroyFunctie',              [                  'ip' => $request->ip(),                  'data' => $request->all(),                  'functie id' => $id              ]          );          Werknemer::where('functie\_id', $id)->delete();          $content = [              'success' => true,              'data'    => $id,              'access\_token' => auth()->user()->createToken('API Token')->plainTextToken,              'token\_type' => 'Bearer',          ];          return response()->json($content, 202);      } |
| } |

## Testen: Update featuretests voor de API’s

Omdat de api’s beveiligd zijn met een token, moet de test ervoor zorgen dat er een token wordt gebruikt. Hiervoor kun je de *Sanctum* method *actingAs* gebruiken. Doe je dat niet, krijg lukt het testen nietl melding: *Route [login] not defined*.

Zie: <https://laravel.com/docs/10.x/sanctum#testing>

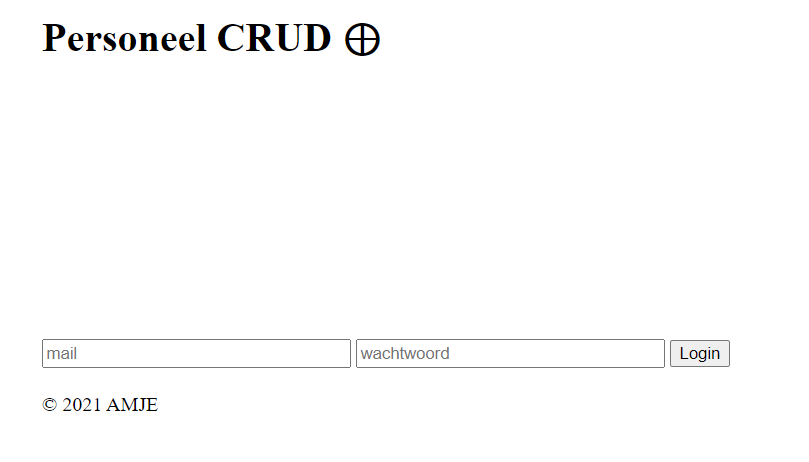
### PersoneelTest

|  |
| --- |
| *Feature test staat in test/feauture* |
| <?php  namespace Tests\Feature;  …GEEN AANPASSINGEN…  use App\Models\User;  use Laravel\Sanctum\Sanctum;  class PersoneelTest extends TestCase  {      /\*\*       \* Test dat api/werknemers/{id} werkt (GET)       \* @return void       \*/      public function test\_werknemer\_op\_id()      {          Sanctum::actingAs(User::factory()->create(),['\*']);    …GEEN AANPASSINGEN…      }      /\*\*       \* Test dat api/werknemers?naam={naam} (GET)       \* @return void       \*/      public function test\_werknemer\_op\_naam()      {          Sanctum::actingAs(User::factory()->create(),['\*']);    …GEEN AANPASSINGEN…      }      /\*\*       \* Test dat api/werknemers werkt, om werknemer toe te voegen (POST)       \* @return void       \*/      public function test\_insert\_werknemer()      {          Sanctum::actingAs(User::factory()->create(),['\*']);    …GEEN AANPASSINGEN…      }      /\*\*       \* Test dat api/werknemers werkt, om werknemer toe te voegen (POST)       \* @return void       \*/      public function test\_delete\_werknemer()      {          Sanctum::actingAs(User::factory()->create(),['\*']);    …GEEN AANPASSINGEN…      }  } |
| Uitvoeren van de test: php artisan test |

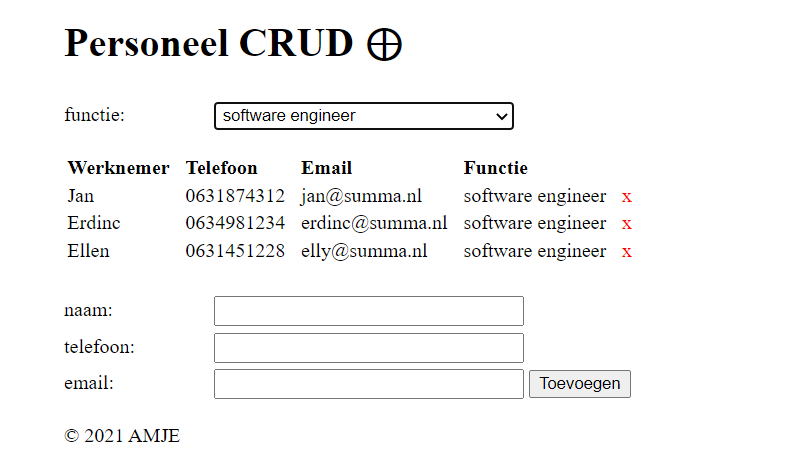
## Update Front-End applicatie

### Beschrijving

Bij start van de applicatie wordt het volgende scherm getoond, waarop de gebruiker kan inloggen.



Nadat de gebruiker is ingelogd, wordt de app info getoond. Dit scherm is verder niet veranderd.



### Code

|  |
| --- |
| <body>      <h1>Personeel CRUD &oplus;</h1>      <div id="appContent" style="visibility: hidden">          <div>              <label for="functie">functie:</label>  <select id="functie" onclick="laadFunctieWerknemers()"></select>          </div>          <br />          <table>              <tr>                  <th>Werknemer</th>                  <th>Telefoon</th>                  <th>Email</th>                  <th>Functie</th>              </tr>              <tbody id="tabelInhoud" />          </table>          <div>              <label for="naam">naam:</label><input type="text" id="naam" required /><br />              <label for="telefoon">telefoon:</label><input type="text" id="telefoon" /><br />              <label for="email">email:</label><input type="email" id="email" />              <button onclick="voegToe()" type="button">Toevoegen</button>          </div>      </div>      <div id="appLogin" style="visibility: visible">          <form action="javascript:void(0);">              <input type="email" placeholder="mail" id="mail" required>              <input type="password" placeholder="wachtwoord" id="wachtwoord" required>              <button type="submit" onclick="login()">Login</button>          </form>      </div>      <footer>&copy; 2021 AMJE</footer>  </body> |
| * Laad is alleen van toepassing bij API 1, bij API 2 weghalen * Er is een div toegevoegd, waarmee de app data gehide kan worden, zolang de gebruiker niet in ingelogd. * Zodra de gebruiker niet is ingelogd, wordt een form getoond waarmee de gebruiker kan inloggen. |
| <script>      "use strict"      const apiBasis = "http://127.0.0.1:8000/api/"      const apiWerknemers = apiBasis + "werknemers"      const apiFuncties = apiBasis + "functies"      const apiLogin = apiBasis + "login"      let access\_token = undefined      let functies = []      let werknemers = [] |
| * De api om in te loggen is toegevoegd * Er is een variabele toegevoegd voor het access token |
| const laadFuncties = async () => {  … GEEN AANPASSINGEN …      } |
| * De function *laadFuncties* leest met een api alle functies in en zet deze in de tabel met functies. * Het resultaat bevat een JSON tabel waarvan de rijen in de tabel *functies* worden gezet en die bovendien in een <option>-tag worden. Alle <option>-tags worden in de combobox (<select>) met id *functie* gezet |
| const laadFunctieWerknemers = async () => {          // console.log('Begin laadfunctiewerknemers', access\_token)          const functie = document.querySelector("#functie").value          const apiFunctiesWerknemers = `${apiFuncties}/${functie}/werknemers?sort=naam`          const response = await axios.get(apiFunctiesWerknemers, {              headers: {                  'Content-Type': 'application/json',                  'Accept': 'application/json',                  'Authorization': 'Bearer ' + access\_token              }          })          if (response.status != 200) {              alert(                  `Fout bij het lezen van de werknemers: ${response.status} (${response.statusText})`              )              return          }          const json = await response.data          access\_token = json.access\_token          werknemers = json.data          // console.log('Eind laadfunctiewerknemers', access\_token)          toon()      } |
| * Hier worden medewerkergegevens opgehaald. De api vereist een token en geeft een nieuwe token terug. * De nieuwe token wordt in de variabele access\_token gezet. * NB: Neem de console.log’s optioneel over (moeten in productie comments zijn) |
| const toon = () => {  … GEEN AANPASSINGEN …      } |
| const laad = async () => {  … GEEN AANPASSINGEN …      } |
| const voegToe = async () => {          // console.log('Begin voegToe', access\_token)          var werknemer = {};          werknemer.naam = document.querySelector("#naam").value;          werknemer.telefoon = document.querySelector("#telefoon").value;          werknemer.email = document.querySelector("#email").value;          werknemer.functie\_id = document.querySelector("#functie").value;          // Onderstaande is met een try-catch verwerkt. Axios genereert exceptions bij responsecodes          // buiten het 2xx bereik. Response code 400 krijg je uit de API (validatiefout)          let response = ''          try {              response = await axios.post(apiWerknemers, werknemer, {                  headers: {                      'Content-Type': 'application/json',                      'Accept': 'application/json',                      'Authorization': 'Bearer ' + access\_token                  }              })          } catch (error) {              // console.log(error.response)              if (error.response.status == 400) {                  access\_token = error.response.data.access\_token                  alert(`${error.response.data.foutmelding}`)                  return              }              alert(                  `Fout bij het toevoegen van de werknemer:                  ${error.response.status} (${error.response.statusText})                  \nLog opnieuw aan om een token te verkrijgen`              )              return          }          if (response.status != 201) {              alert(                  `Fout bij het toevoegen van de werknemer: ${response.status} (${response.statusText})`              )              return          }          const json = await response.data          access\_token = json.access\_token          document.querySelector("#naam").value = ''          document.querySelector("#telefoon").value = ''          document.querySelector("#email").value = ''          // console.log('Einde voegToe', access\_token)          await laadFunctieWerknemers()      } |
| * Hier worden medewerkergegevens toegevoegd. De api vereist een token en geeft een nieuwe token terug. * De nieuwe token wordt in de variabele access\_token gezet. * Doordat AXIOS alle response codes buiten het bereik 2xx via een exception terugspeelt, moet je hier een try-catch gebruiken (eventueel ,catch()). Zorg dat je de token ook nu bewaart. |
| const verwijder = async (id) => {          console.log("Begin verwijder: ", id, access\_token)          const apiDelete = `${apiWerknemers}/${id}`          console.log("api", apiDelete)          const response = await axios.delete(apiDelete, {              headers: {                  'Content-Type': 'application/json',                  'Accept': 'application/json',                  'Authorization': 'Bearer ' + access\_token              }          })          if (response.status != 202) {              alert(  `Fout bij het verwijderen van werknemers: ${response.status} (${response.statusText})`  )          }          const json = await response.data          access\_token = json.access\_token          werknemers = await response.data  // console.log("Einde verwijder: ", id, access\_token)          await laadFunctieWerknemers()      } |
| * Hier worden medewerkergegevens verwijderd. De api vereist een token en geeft een nieuwe token terug. * De nieuwe token wordt in de variabele access\_token gezet. |
| const login = async () => {          // console.log('Begin Login')          const gebruiker = {}          gebruiker.password = document.querySelector("#wachtwoord").value          gebruiker.email = document.querySelector("#mail").value          const respons = await axios.post(  apiLogin, gebruiker, { headers: { 'Content-Type': 'application/json' } })          // console.log('respons', respons)          access\_token = await respons.data.access\_token          // console.log('access\_token: ', access\_token)          document.querySelector("#appLogin").style.visibility = "hidden"          document.querySelector("#appContent").style.visibility = "visible"          // console.log('Einde login')          await laad()      } |
| * Deze function wordt gestart als nadat de gebruiker op de loginknop drukt. De login-api heeft geen token nodig. Bij een succesvolle inlog, bevat het antwoord van de api een bearer token die in de variabele access\_token wordt gezet. * Daarna wordt ervoor gezorgd dat de HTML-div-tag waarbinnen het inlogformulier staat, gehide wordt en dat de HTML-div-tag waarbinnen de app data staat, visible wordt. |
| </script>  </html> |

## Testen: Update testplan en voer het uit

### Inleiding

Het testrapport moet worden aangepast. :

* Scenario’s op het ingelogd zijn.
* NB: Je kunt dit niet enkel testen met de front end. Alle functies moeten getest worden met zowel de FE als met Postman

### Testrapport

#### Inloggen van een gebruiker

##### Scenario: Geen backend

|  |  |
| --- | --- |
| Scenario | Backend niet beschikbaar |
| Verwacht resultaat | Melding waarschuw Service desk |
| Werkelijk resultaat | Console laat zien dat er een fout is opgetreden |
| Acties | Student lost het op! |

##### Scenario: Geen database

|  |  |
| --- | --- |
| Scenario | Backend beschikbaar  Database niet beschikbaar |
| Verwacht resultaat | Melding waarschuw Service desk |
| Werkelijk resultaat | Console laat zien dat er een fout is opgetreden |
| Acties | Student lost het op! |

##### Scenario: Gebruiker typt ingeldige naam en/of wachtwoord in

|  |  |
| --- | --- |
| Scenario | Backend beschikbaar  Database beschikbaar   1. Email leeg, password leeg 2. Email ongeldig, password leeg 3. Email leeg, password gevuld 4. Email ongeldig, password gevuld 5. Email geldig, password niet van email |
| Verwacht resultaat | Melding niet geautoriseerd |
| Werkelijk resultaat | Geen melding |
| Acties | Student lost het op! |

##### Scenario: Gebruiker correct email en bijbehorende wachtwoord in

|  |  |
| --- | --- |
| Scenario | Backend beschikbaar  Database beschikbaar  Email en password correct gevuld van geregistreerde gebruiker |
| Verwacht resultaat | Bearer token beschikbaar voor gebruiker  Overzicht werknemers met functie wordt getoond |
| Werkelijk resultaat | Zoals verwacht |
| Acties | n.v.t. |

#### Opvragen medewerkers met een functie

##### Scenario: Geen backend

|  |  |
| --- | --- |
| Scenario | Backend niet beschikbaar |
| Verwacht resultaat | Melding waarschuw Service desk |
| Werkelijk resultaat | Console laat zien dat er een fout is opgetreden |
| Acties | Student lost het op! |

##### Scenario: Geen database

|  |  |
| --- | --- |
| Scenario | Backend beschikbaar  Database niet beschikbaar |
| Verwacht resultaat | Melding waarschuw Service desk |
| Werkelijk resultaat | Console laat zien dat er een fout is opgetreden |
| Acties | Student lost het op! |

##### Scenario: Selecteer Medewerkers met functie Engineer

|  |  |
| --- | --- |
| Scenario | Backend beschikbaar  Database beschikbaar  Selecteer de functie Engineer |
| Verwacht resultaat | Alle medewerkers met de functie Engineer: |
| Werkelijk resultaat | Zoals verwacht |
| Acties | n.v.t. |

#### Toevoegen medewerkers

##### Scenario: Geen backend

|  |  |
| --- | --- |
| Scenario | Backend niet beschikbaar |
| Verwacht resultaat | Melding waarschuw Service desk |
| Werkelijk resultaat | Console laat zien dat er een fout is opgetreden |
| Acties | Student lost het op! |

##### Scenario: Geen database

|  |  |
| --- | --- |
| Scenario | Backend beschikbaar  Database niet beschikbaar |
| Verwacht resultaat | Melding waarschuw Service desk |
| Werkelijk resultaat | Console laat zien dat er een fout is opgetreden |
| Acties | Student lost het op! |

##### Scenario: Ongeldige naam en email (Naam en email leeg)

|  |  |
| --- | --- |
| Scenario | Backend beschikbaar  Database beschikbaar  Gebruiker laat naam en email leeg. |
| Verwacht resultaat | Medewerker wordt niet toegevoegd.  Er volgt een melding. |
| Werkelijk resultaat | Medewerker wordt niet toegevoegd.  Er volgt geen melding. In plaats daarvan in de console log een melding. |
| Acties | Student lost het op! |

##### Scenario: Ongeldige naam en email (Naam leeg, en email ongeldig)

|  |  |
| --- | --- |
| Scenario | Backend beschikbaar  Database beschikbaar  Gebruiker laat naam en vult ongeldig email in. |
| Verwacht resultaat | Medewerker wordt niet toegevoegd.  Er volgt een melding. |
| Werkelijk resultaat | Medewerker wordt niet toegevoegd.  Er volgt geen melding. In plaats daarvan in de console log een melding. |
| Acties | Student lost het op! |

##### Scenario: Ongeldige naam en email (Naam leeg, en geldig email)

|  |  |
| --- | --- |
| Scenario | Backend beschikbaar  Database beschikbaar  Gebruiker laat naam en vult geldig email in. |
| Verwacht resultaat | Medewerker wordt niet toegevoegd.  Er volgt een melding. |
| Werkelijk resultaat | Medewerker wordt niet toegevoegd.  Er volgt geen melding. In plaats daarvan in de console log een melding. |
| Acties | Student lost het op! |

##### Scenario: Ongeldige naam en email (Naam geldig, en email leeg)

|  |  |
| --- | --- |
| Scenario | Backend beschikbaar  Database beschikbaar  Gebruiker vult naam in en laat email leeg. |
| Verwacht resultaat | Medewerker wordt niet toegevoegd.  Er volgt een melding. |
| Werkelijk resultaat | Medewerker wordt niet toegevoegd.  Er volgt geen melding. In plaats daarvan in de console log een melding. |
| Acties | Student lost het op! |

##### Scenario: Ongeldige naam en email (Naam geldig, en email ongeldig)

|  |  |
| --- | --- |
| Scenario | Backend beschikbaar  Database beschikbaar  Gebruiker vult naam in en vult ongeldig email in. |
| Verwacht resultaat | Medewerker wordt niet toegevoegd.  Er volgt een melding. |
| Werkelijk resultaat | Medewerker wordt niet toegevoegd.  Er volgt geen melding. In plaats daarvan in de console log een melding. |
| Acties | Student lost het op! |

##### Scenario: Ongeldige naam en email (Naam geldig, en email geldig)

|  |  |
| --- | --- |
| Scenario | Backend beschikbaar  Database beschikbaar  Gebruiker vult naam en geldig email in. |
| Verwacht resultaat | Gebruiker wordt toegevoegd.  Scherm wordt bijgewerkt. |
| Werkelijk resultaat | Volgens verwacht resultaat. |
| Acties | n.v.t. |

#### Verwijderen medewerkers

##### Scenario: Geen backend

|  |  |
| --- | --- |
| Scenario | Backend niet beschikbaar  Gebruiker clickt op x achter medewerker. |
| Verwacht resultaat | Melding contact met service desk |
| Werkelijk resultaat | Er volgt geen melding |
| Acties | Student past aan! |

##### Scenario: Geen database

|  |  |
| --- | --- |
| Scenario | Backend beschikbaar  Database niet beschikbaar  Gebruiker clickt op x achter medewerker. |
| Verwacht resultaat | Melding contact met service desk |
| Werkelijk resultaat | Er volgt geen melding |
| Acties | Student past aan! |

##### Scenario: Verwijder medewerker

|  |  |
| --- | --- |
| Scenario | Backend beschikbaar  Database beschikbaar  Gebruiker clickt op x achter medewerker. |
| Verwacht resultaat | Medewerker wordt verwijderd.  Scherm wordt bijgewerkt. |
| Werkelijk resultaat | Volgens verwacht resultaat. |
| Acties | n.v.t. |

## Lever applicatie op

Demonstratie en uitleg code

## Overzicht gebruikte Laravel commando's

|  |  |
| --- | --- |
| Maken laravel project ApiPersoneel | laravel new ApiPersoneel |
| Maken database personeel | XAMPP: database apipersoneel maken  Laravel Env bevay al de juiste databasenaam |
| Maken migrations | php artisan make:migration functies  php artisan make:migration werknemers |
| Uitvoeren migration | php artisan migrate  php artisan migrate:rollback |
| Maken seeder | php artisan make:seeder FunctiesSeeder  php artisan make:seeder WerknemersSeeder |
| Alle seeders runnen | php artisan db:seed  php artisan migrate:fresh --seed |
| FeatureTest maken | php artisan make:test PersoneelTest |
| Testen uitvoeren | php artisan test |

## Maak Front-End applicatie (Extra: Flutter)

### Beschrijving

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Bij start van de applicatie wordt het volgende scherm getoond, waarop de gebruiker kan inloggen. | Nadat de gebruiker is ingelogd, krijgt hij toegang tot de applicatie: | Via de bottomnavigatie kan hij tussen het homescreen en index screen kiezen.  Op het index screen kan de gebruiker:   1. Een functie kiezen, waarna alle werkemers met die functie worden getoond 2. Naar het toevoegscherm gaan (+ button) 3. Naar het wijzigscherm gaan (potlood) 4. Een werknemer verwijderen (prullenbak) |
|  |  | De applicatie is ook beschikbaar voor de volgende platforms:   * Android (mobile app) * Web (website) * Windows (desktop) |
| Toevoegscherm | Wijzigscherm |  |

### Code

#### AndroidManifest.xml

|  |
| --- |
| AndroidManifest.xml |
|  |
| Folderstructuur |
|  |

#### Assets

|  |
| --- |
| Assets |
| pubspec.yaml |
|  |
| employees.jpg |
| * Zoek een geschikte afbeelding |

#### Components

|  |
| --- |
| Components  Herbruikbare componenten (optimalisatie) die zowel bij toevoegen en bij wijzigen worden gebruikt.  Betreft TextFormFields (naam, email, datum sinds) |
| email\_text\_form\_field.dart |
| import 'package:flutter/material.dart';  class EmailTextFormField extends StatelessWidget {  final TextEditingController controller;   const EmailTextFormField({  Key? key,  required this.controller,  }) : super(key: key);   @override  Widget build(BuildContext context) {  return TextFormField(  controller: controller,  decoration: const InputDecoration(  labelText: 'email',  border: OutlineInputBorder(),  ),  validator: (value) {  if (value == null || value.isEmpty) {  return 'Vul email in';  }   *// Onderstaande code via internet gevonden:* const pattern = r"(?:[a-z0-9!#$%&'\*+/=?^\_`{|}~-]+(?:\.[a-z0-9!#$%&'"  r'\*+/=?^\_`{|}~-]+)\*|"(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21\x23-\x5b\x5d-'  r'\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])\*")@(?:(?:[a-z0-9](?:[a-z0-9-]\*'  r'[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]\*[a-z0-9])?|\[(?:(?:(2(5[0-5]|[0-4]'  r'[0-9])|1[0-9][0-9]|[1-9]?[0-9]))\.){3}(?:(2(5[0-5]|[0-4][0-9])|1[0-9]'  r'[0-9]|[1-9]?[0-9])|[a-z0-9-]\*[a-z0-9]:(?:[\x01-\x08\x0b\x0c\x0e-\x1f\'  r'x21-\x5a\x53-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])+)\])';  final regex = RegExp(pattern);  return regex.hasMatch(value)  ? null  : 'Vul een geldig emailadres in';  },  );  } } |
| naam\_text\_form\_field.dart |
| import 'package:flutter/material.dart';  class NaamTextFormField extends StatelessWidget {  final TextEditingController controller;   const NaamTextFormField({  Key? key,  required this.controller,  }) : super(key: key);   @override  Widget build(BuildContext context) {  return TextFormField(  controller: controller,  decoration: const InputDecoration(  labelText: 'naam',  border: OutlineInputBorder(),  ),  validator: (value) {  if (value == null || value.isEmpty) {  return 'Vul naam in';  }  return null;  },  );  } } |
| sinds\_text\_form\_field.dart |
| import 'package:flutter/material.dart';  class SindsTextFormField extends StatelessWidget {  final TextEditingController controller;   const SindsTextFormField({  Key? key,  required this.controller,  }) : super(key: key);   @override  Widget build(BuildContext context) {  return TextFormField(  controller: controller,  decoration: const InputDecoration(  labelText: 'Datum sinds',  border: OutlineInputBorder(),  ),  validator: (value) {  if (value == null || value.isEmpty) {  return 'Vul datum sinds in';  }  if (value.length != 10) {  return 'Vul een geldige datum in JJJJ-MM-DD';  }  try {  var parsedDate = DateTime.*parse*(value);  }  *// on FormatException catch (formatError) {* on FormatException {  return 'Vul een geldige datum in JJJJ-MM-DD';  }  return null;  },  );  } } |

#### Models

|  |
| --- |
| Models:bevat classen voor Functie en Werknemer |
| functie.dart |
| class Functie {  final int id;  final String naam;   Functie({  required this.id,  required this.naam,  });   *// Onderstaande is nodig i.v.m. gebruik DropdownButton* @override  bool operator ==(Object other) {  if (other is Functie) {  return other.id == id;  }  else {  return false;  }  }   @override  int get hashCode {  return Object.*hash*(id, naam);  } |
| werknemer.dart |
| class Werknemer {  final int id;  final String naam;  final int functieId;  final String? telefoon;  final String email;  final String? sinds;   Werknemer({  required this.id,  required this.naam,  required this.functieId,  this.telefoon,  required this.email,  this.sinds,  });  } |

#### Pages

|  |
| --- |
| Pages:bevat de pages (schermen) die worden gebruikt |
| login\_page.dart |
| import 'package:flutter/material.dart'; import 'package:personeels\_app/services/authentication\_services.dart';  class LoginPage extends StatefulWidget {  const LoginPage({Key? key, required this.setSignedIn}) : super(key: key);   final void Function(bool signedIn) setSignedIn;   @override  State<LoginPage> createState() => \_LoginPageState(); }  class \_LoginPageState extends State<LoginPage> {  final \_formKey = GlobalKey<FormState>();  final \_emailTextController = TextEditingController();  final \_passwordTextController = TextEditingController();   @override  void dispose() {  \_emailTextController.dispose();  \_passwordTextController.dispose();  super.dispose();  }   @override  Widget build(BuildContext context) {  return Scaffold(  body: Form(  key: \_formKey,  child: ListView(  children: [  *// email* TextFormField(  controller: \_emailTextController,  textInputAction: TextInputAction.next,  decoration: const InputDecoration(  border: OutlineInputBorder(), hintText: 'email adres'),  validator: (value) {  if (value == null || value.isEmpty) {  return 'Vul emailadres in';  }  return null;  },  ),  *// password* TextFormField(  controller: \_passwordTextController,  textInputAction: TextInputAction.next,  obscureText: true,  decoration: const InputDecoration(  border: OutlineInputBorder(), hintText: 'password'),  validator: (value) {  if (value == null || value.isEmpty) {  return 'Vul password in';  }  return null;  },  ),  *// submit button* ElevatedButton(  onPressed: () async {  if (\_formKey.currentState!.validate()) {  try {  final result = await AuthenticationServices.*login*(  \_emailTextController.text,  \_passwordTextController.text,  );  widget.setSignedIn(result);  } catch (e) {  widget.setSignedIn(false);  }  }  },  child: const Text('Inloggen'),  ),  ],  ),  ),  );  } } |
| main\_page.dart |
| import 'package:flutter/material.dart'; import 'package:personeels\_app/pages/werknemers\_home\_page.dart'; import 'package:personeels\_app/pages/werknemers\_index\_page.dart';  class MainPage extends StatefulWidget {  const MainPage({Key? key, required this.setSignedIn}) : super(key: key);   final void Function(bool signedIn) setSignedIn;   @override  State<MainPage> createState() => \_MainPageState(); }  class \_MainPageState extends State<MainPage> {  @override  Widget build(BuildContext context) {  return DefaultTabController(  length: 2,  child: Scaffold(  bottomNavigationBar: Container(  color: Colors.*blue*,  child: const TabBar(  tabs: [  Tab(icon: Icon(Icons.*info\_outline*)),  Tab(icon: Icon(Icons.*people*)),  ],  ),  ),  body: TabBarView(  children: [  const WerknemersHomePage(),  WerknemersIndexPage(setSignedIn: widget.setSignedIn),  ],  ),  ),  );  } } |
| werknemers\_create\_page.dart |
| import 'package:flutter/material.dart'; import 'package:personeels\_app/components/email\_text\_form\_field.dart'; import 'package:personeels\_app/components/naam\_text\_form\_field.dart'; import 'package:personeels\_app/components/sinds\_text\_form\_field.dart'; import 'package:personeels\_app/models/functie.dart'; import 'package:personeels\_app/services/werknemers\_services.dart';  class WerknemersCreatePage extends StatefulWidget {  final Functie functie;   const WerknemersCreatePage({Key? key, required this.functie})  : super(key: key);   @override  State<WerknemersCreatePage> createState() => \_WerknemersCreatePageState(); }  class \_WerknemersCreatePageState extends State<WerknemersCreatePage> {  final \_formKey = GlobalKey<FormState>();   final \_naamController = TextEditingController();  final \_telefoonController = TextEditingController();  final \_emailController = TextEditingController();  final \_sindsController = TextEditingController();   @override  void dispose() {  \_naamController.dispose();  \_telefoonController.dispose();  \_emailController.dispose();  \_sindsController.dispose();  super.dispose();  }   @override  Widget build(BuildContext context) {  return Scaffold(  appBar: AppBar(  title: Text('Werknemers - Toevoegen - ${widget.functie.naam}'),  ),  body: SingleChildScrollView(  child: Padding(  padding: const EdgeInsets.all(8.0),  child: Form(  key: \_formKey,  child: Column(  children: [  *// Functie* Row(children: [  Text(  widget.functie.naam,  style: TextStyle(fontWeight: FontWeight.*bold*, fontSize: 16),  )  ]),   *//naam* const SizedBox(height: 20),  NaamTextFormField(controller: \_naamController),   *// email* const SizedBox(height: 20),  EmailTextFormField(controller: \_emailController),   *// sinds* const SizedBox(height: 20),  SindsTextFormField(controller: \_sindsController),   *// Buttons Save en Cancel* const SizedBox(height: 40),  Row(  mainAxisAlignment: MainAxisAlignment.center,  children: [  ElevatedButton(  child: const Text('Toevoegen'),  onPressed: () async {  if (\_formKey.currentState!.validate()) {  await WerknemersServices.*create*(  naam: \_naamController.text,  functieId: widget.functie.id,  telefoon: \_telefoonController.text,  email: \_emailController.text,  sinds: \_sindsController.text);  Navigator.*of*(context).pop();  }  },  ),  const SizedBox(width: 20),  ElevatedButton(  child: const Text('Annuleer'),  onPressed: () {  Navigator.*of*(context).pop();  },  )  ],  )  ],  ),  ),  ),  ),  );  } } |
| werknemers\_home\_page.dart |
| import 'package:flutter/material.dart';  class WerknemersHomePage extends StatelessWidget {  const WerknemersHomePage({Key? key}) : super(key: key);   @override  Widget build(BuildContext context) {  return Container(  decoration: const BoxDecoration(  image: DecorationImage(  image: AssetImage(  'lib/assets/employees.jpg',  ),  fit: BoxFit.fill  ),  ),  );  } } |
| werknemers\_index\_page.dart |
| import 'package:flutter/material.dart'; import 'package:personeels\_app/models/functie.dart'; import 'package:personeels\_app/models/werknemer.dart'; import 'package:personeels\_app/pages/werknemers\_create\_page.dart'; import 'package:personeels\_app/pages/werknemers\_update\_page.dart'; import 'package:personeels\_app/services/functies\_services.dart'; import 'package:personeels\_app/services/werknemers\_services.dart';  class WerknemersIndexPage extends StatefulWidget {  const WerknemersIndexPage({Key? key, required this.setSignedIn})  : super(key: key);   final void Function(bool signedIn) setSignedIn;   @override  State<WerknemersIndexPage> createState() => \_WerknemersIndexPageState(); }  class \_WerknemersIndexPageState extends State<WerknemersIndexPage> {  Future<List<Werknemer>>? \_werknemers;  Functie? \_selectedFunctie;   void \_refreshIndex(Functie selectedFunctie) {  setState(() {  *// Werk combobox selected item bij* \_selectedFunctie = selectedFunctie;  *// Werk list werknemersbij* \_werknemers =  WerknemersServices.*getAllByFunctie*(functieId: \_selectedFunctie!.id);  });  }   @override  Widget build(BuildContext context) {  return Scaffold(  appBar: AppBar(  title: const Text('Werknemers Index'),  actions: [\_logout()],  ),  floatingActionButton: \_btnCreateWerknemer(),  body: Column(  children: [  *// Combobox met functies* \_cmbFuncties(),  *// Lijst met werknemers* const SizedBox(height: 20),  Expanded(child: \_lstWerknemers()),  ],  ),  );  }   IconButton \_logout() {  return IconButton(  onPressed: () {  widget.setSignedIn(false);  },  icon: const Icon(Icons.*logout*),  );  }   FloatingActionButton \_btnCreateWerknemer() {  return FloatingActionButton(  onPressed: () async {  Functie selectedFunctie = \_selectedFunctie!;   await Navigator.*of*(context).push(MaterialPageRoute(  builder: (context) =>  WerknemersCreatePage(functie: selectedFunctie),  ));  \_refreshIndex(selectedFunctie);  },  child: const Icon(  Icons.*add*,  ),  );  }   \_btnUpdateWerknemer({required Werknemer werknemer}) {  return IconButton(  onPressed: () async {  Functie selectedFunctie = \_selectedFunctie!;   await Navigator.*of*(context).push(MaterialPageRoute(  builder: (context) => WerknemersUpdatePage(werknemer: werknemer),  ));  \_refreshIndex(selectedFunctie);  },  icon: const Icon(  Icons.*edit*,  color: Colors.*red*,  ),  );  }   \_btnDeleteWerknemer({required Werknemer werknemer}) {  return IconButton(  onPressed: () async {  await WerknemersServices.*delete*(werknemerId: werknemer.id);  \_refreshIndex(\_selectedFunctie!);  },  icon: const Icon(  Icons.*delete*,  color: Colors.*red*,  ),  );  }   FutureBuilder<List<Functie>> \_cmbFuncties() {  return FutureBuilder<List<Functie>>(  future: FunctieServices.*getAll*(),  builder: (context, snapshot) {  if (snapshot.hasError) {  return Center(child: Text(snapshot.error.toString()));  }   if (!snapshot.hasData) {  return const Center(child: CircularProgressIndicator());  }   List<Functie> functies = snapshot.data!;   return DropdownButton<Functie>(  value: \_selectedFunctie ?? functies.first,  icon: const Icon(Icons.*arrow\_downward*),  underline: Container(height: 2, color: Colors.*blue*),  onChanged: (Functie? value) {  *// This is called when the user selects an item.* \_refreshIndex(value!);  },  items: functies.map<DropdownMenuItem<Functie>>((Functie value) {  return DropdownMenuItem<Functie>(  value: value,  child: Text(value.naam),  );  }).toList(),  );  },  );  }   Widget \_lstWerknemers() {  return \_werknemers == null  ? const Center(child: CircularProgressIndicator())  : FutureBuilder(  future: \_werknemers,  builder: (context, snapshot) {  if (snapshot.hasError) {  return Center(child: Text(snapshot.error.toString()));  }  if (snapshot.hasData == false) {  return const Center(child: CircularProgressIndicator());  }  return ListView.builder(  itemCount: snapshot.data!.length,  itemBuilder: (context, index) {  return ListTile(  title: Text(snapshot.data![index].naam),  subtitle: Text(snapshot.data![index].email),  leading:  \_btnUpdateWerknemer(werknemer: snapshot.data![index]),  trailing:  \_btnDeleteWerknemer(werknemer: snapshot.data![index]),  );  },  );  },  );  } } |
| werknemers\_update\_page.dart |
| import 'package:flutter/material.dart'; import 'package:personeels\_app/components/email\_text\_form\_field.dart'; import 'package:personeels\_app/components/naam\_text\_form\_field.dart'; import 'package:personeels\_app/components/sinds\_text\_form\_field.dart'; import 'package:personeels\_app/models/functie.dart'; import 'package:personeels\_app/models/werknemer.dart'; import 'package:personeels\_app/services/functies\_services.dart'; import 'package:personeels\_app/services/werknemers\_services.dart';  class WerknemersUpdatePage extends StatefulWidget {  final Werknemer werknemer;   const WerknemersUpdatePage({  Key? key,  required this.werknemer,  }) : super(key: key);   @override  State<WerknemersUpdatePage> createState() => \_WerknemersUpdatePageState(); }  class \_WerknemersUpdatePageState extends State<WerknemersUpdatePage> {  Functie? \_selectedFunctie;   final \_formKey = GlobalKey<FormState>();   final \_naamController = TextEditingController();  final \_telefoonController = TextEditingController();  final \_emailController = TextEditingController();  final \_sindsController = TextEditingController();   void \_refreshIndex(Functie selectedFunctie) {  setState(() {  *// Werk combobox selected item bij* \_selectedFunctie = selectedFunctie;  });  }   @override  void initState() {  \_naamController.text = widget.werknemer.naam;  \_telefoonController.text = widget.werknemer.telefoon ?? 'Geen';  \_emailController.text = widget.werknemer.email;  \_sindsController.text = widget.werknemer.sinds ?? 'Al heel lang';  super.initState();  }   @override  void dispose() {  \_naamController.dispose();  \_telefoonController.dispose();  \_emailController.dispose();  \_sindsController.dispose();  super.dispose();  }   @override  Widget build(BuildContext context) {  return Scaffold(  appBar: AppBar(  title: const Text('Werknemers - Wijzigen}'),  ),  body: SingleChildScrollView(  child: Padding(  padding: const EdgeInsets.all(8.0),  child: Form(  key: \_formKey,  child: Column(  children: [  *// Functie  // Combobox met functies* Row(children: [\_cmbFuncties()]),   *//naam* const SizedBox(height: 20),  NaamTextFormField(controller: \_naamController),   *// email* const SizedBox(height: 20),  EmailTextFormField(controller: \_emailController),   *// sinds* const SizedBox(height: 20),  SindsTextFormField(controller: \_sindsController),   *// Buttons Save en Cancel* const SizedBox(height: 40),  Row(  mainAxisAlignment: MainAxisAlignment.center,  children: [  ElevatedButton(  child: const Text('Bewaren'),  onPressed: () async {  if (\_formKey.currentState!.validate()) {  await WerknemersServices.*update*(  werknemerId: widget.werknemer.id,  naam: \_naamController.text,  functieId: \_selectedFunctie!.id,  telefoon: \_telefoonController.text,  email: \_emailController.text,  sinds: \_sindsController.text);  *// Check of de context nog beschikbaar is* if (context.mounted){  Navigator.*of*(context).pop();  }  }  },  ),  const SizedBox(width: 20),  ElevatedButton(  child: const Text('Annuleer'),  onPressed: () {  Navigator.*of*(context).pop();  },  )  ],  )  ],  ),  ),  ),  ),  );  }   FutureBuilder<List<Functie>> \_cmbFuncties() {  return FutureBuilder<List<Functie>>(  future: FunctieServices.*getAll*(),  builder: (context, snapshot) {  if (snapshot.hasError) {  return Center(child: Text(snapshot.error.toString()));  }   if (!snapshot.hasData) {  return const Center(child: CircularProgressIndicator());  }   List<Functie> functies = snapshot.data!;   *// zoek de functie van de medewerker en zet deze in \_selectedFunctie  // NB: Als er al een functie geselecteerd was, moet de waarde  // ongewijzigd blijven, omdat de gebruiker anders geen andere  // functie kan kiezen.* \_selectedFunctie ??= functies  .firstWhere((element) => element.id == widget.werknemer.functieId);   return DropdownButton<Functie>(  value: \_selectedFunctie,  icon: const Icon(Icons.*arrow\_downward*),  underline: Container(height: 2, color: Colors.*blue*),  onChanged: (Functie? value) {  *// This is called when the user selects an item.* \_refreshIndex(value!);  },  items: functies.map<DropdownMenuItem<Functie>>((Functie value) {  return DropdownMenuItem<Functie>(  value: value,  child: Text(value.naam),  );  }).toList(),  );  },  );  } } |

#### Services

|  |
| --- |
| authentication\_services.dart |
| import 'dart:convert'; import 'package:http/http.dart' as http; import 'package:personeels\_app/services/platform\_services.dart';  class AuthenticationServices {  *// Chrome, Edge, Windows, IOS* static final String *\_baseApi* = PlatformServices.*isAndroid* ? 'http://10.0.2.2:8000/api'  : 'http://127.0.0.1:8000/api';   static String *\_bearerToken* = '';   static String *getBearerToken*() {  return *\_bearerToken*;  }   static void *setBearerToken*(String bearerToken) {  *\_bearerToken* = bearerToken;  }   *// api/register/* static Future<bool> *register*(  String email, String password, String name) async {  final response = await http.post(  Uri.*parse*('$*\_baseApi*/register'),  headers: <String, String>{  'Content-Type': 'application/json; charset=UTF-8',  },  body: jsonEncode(<String, String>{  'name': name,  'email': email,  'password': password,  'password\_confirmation': password  }),  );   return response.statusCode == 200;  }   *// api/login/* static Future<bool> *login*(String email, String password) async {  final response = await http.post(  Uri.*parse*('$*\_baseApi*/login'),  headers: <String, String>{  'Content-Type': 'application/json; charset=UTF-8',  },  body: jsonEncode(<String, String>{'email': email, 'password': password}),  );   if (response.statusCode == 200) {  final result = jsonDecode(response.body);  *\_bearerToken* = result['access\_token'];  }   return response.statusCode == 200;  }   *// api/logout/* static Future<bool> *logout*() async {  final response = await http.post(  Uri.*parse*('$*\_baseApi*/logout'),  headers: <String, String>{  'Content-Type': 'application/json; charset=UTF-8',  'Authorization': 'Bearer $*\_bearerToken*'  },  );   return response.statusCode == 200;  } } |
| functies\_services.dart |
| import 'dart:convert'; import 'package:http/http.dart' as http; import 'package:personeels\_app/models/functie.dart'; import 'package:personeels\_app/services/platform\_services.dart';  class FunctieServices { *// Chrome, Edge, Windows, IOS* static final String *\_baseApi* = PlatformServices.*isAndroid* ? 'http://10.0.2.2:8000/api'  : 'http://127.0.0.1:8000/api';   *// api/register/* static Future<List<Functie>> *getAll*() async {  *// print('Start Functies getAll Bearer ${AuthenticationServices.getBearerToken()}');* final response = await http.get(Uri.*parse*('$*\_baseApi*/functies'));   if (response.statusCode != 200) {  Exception('Fout bij ophalen functies ${response.statusCode}');  }   List<Functie> functies = [];  final json = jsonDecode(response.body);  *// print('Json Functies getAll $json');* for (int i = 0; i < json['data'].length; i++) {  functies.add(  Functie(id: json['data'][i]['id'], naam: json['data'][i]['naam']));  }  *// print('Eind Functies getAll Bearer ${AuthenticationServices.getBearerToken()}');* return functies;  } } |
| platform\_services.dart |
| import 'dart:io' show Platform; import 'package:flutter/foundation.dart' show kIsWeb;  class PlatformServices {  static bool get *isMobile* {  if (kIsWeb) {  return false;  } else {  return Platform.*isIOS* || Platform.*isAndroid*;  }  }   static bool get *isDesktop* {  if (kIsWeb) {  return false;  } else {  return Platform.*isLinux* ||  Platform.*isFuchsia* ||  Platform.*isWindows* ||  Platform.*isMacOS*;  }  }   static bool get *isAndroid* {  if (kIsWeb) {  return false;  } else {  return Platform.*isAndroid*;  }  } } |
| werknemers\_services.dart |
| import 'dart:convert'; import 'package:http/http.dart' as http; import 'package:personeels\_app/models/werknemer.dart'; import 'package:personeels\_app/services/authentication\_services.dart'; import 'package:personeels\_app/services/platform\_services.dart';  class WerknemersServices { *// Chrome, Edge, Windows, IOS* static final String *\_baseApi* = PlatformServices.*isAndroid* ? 'http://10.0.2.2:8000/api'  : 'http://127.0.0.1:8000/api';   static Future<List<Werknemer>> *getAll*() async {  *// print('Start Werknemers GetAll Bearer ${AuthenticationServices.getBearerToken()}');* final response = await http.get(  Uri.*parse*('$*\_baseApi*/werknemers'),  headers: <String, String>{  'Content-Type': 'application/json; charset=UTF-8',  'Authorization': 'Bearer ${AuthenticationServices.*getBearerToken*()}'  },  );   if (response.statusCode != 200) {  Exception('Fout bij ophalen werknemers ${response.statusCode}');  }   List<Werknemer> werknemers = [];  final json = jsonDecode(response.body);  *// print('Json Werknemers GetAll $json');* for (int i = 0; i < json['data'].length; i++) {  werknemers.add(Werknemer(  id: json['data'][i]['id'],  naam: json['data'][i]['naam'],  functieId: json['data'][i]['functie\_id'],  telefoon: json['data'][i]['telefoon'],  email: json['data'][i]['email'],  sinds: json['data'][i]['sinds']));  }  AuthenticationServices.*setBearerToken*(json['access\_token']);  *// print('Eind Werknemers GetAll Bearer ${AuthenticationServices.getBearerToken()}');* return werknemers;  }   static Future<List<Werknemer>> *getAllByFunctie*({  required int functieId,  }) async {  *// print('Start Werknemers getAllByFunctie Bearer ${AuthenticationServices.getBearerToken()}');* final response = await http.get(  Uri.*parse*('$*\_baseApi*/functies/$functieId/werknemers'),  headers: <String, String>{  'Content-Type': 'application/json; charset=UTF-8',  'Authorization': 'Bearer ${AuthenticationServices.*getBearerToken*()}'  },  );   if (response.statusCode != 200) {  Exception(  'Fout bij ophalen werknemers bij functie ${response.statusCode}');  }   List<Werknemer> werknemers = [];  final json = jsonDecode(response.body);  *// print('Json Werknemers getAllByFunctie $json');* for (int i = 0; i < json['data'].length; i++) {  werknemers.add(Werknemer(  id: json['data'][i]['id'],  naam: json['data'][i]['naam'],  functieId: json['data'][i]['functie\_id'],  telefoon: json['data'][i]['telefoon'],  email: json['data'][i]['email'],  sinds: json['data'][i]['sinds']));  }  AuthenticationServices.*setBearerToken*(json['access\_token']);   *// print('Eind Werknemers getAllByFunctie Bearer ${AuthenticationServices.getBearerToken()}');* return werknemers;  }   static Future<Werknemer> *create*({  required String naam,  required int functieId,  required String telefoon,  required String email,  required String sinds,  }) async {  *// print('Start Werknemers create Bearer '  // '${AuthenticationServices.getBearerToken()}');* final response = await http.post(  Uri.*parse*('$*\_baseApi*/werknemers'),  headers: <String, String>{  'Content-Type': 'application/json; charset=UTF-8',  'Authorization': 'Bearer ${AuthenticationServices.*getBearerToken*()}'  },  body: jsonEncode({  "naam": naam,  "functie\_id": functieId,  "telefoon": telefoon,  "email": email,  "sinds": sinds  }),  );   *// print(response.statusCode);* if (response.statusCode != 201) {  Exception('Fout bij verwijderen werknemers ${response.statusCode}');  }   final json = jsonDecode(response.body);  *// print('Json Werknemers create $json');* final werknemer = Werknemer(  id: json['data']['id'],  naam: json['data']['naam'],  functieId: json['data']['functie\_id'],  telefoon: json['data']['telefoon'],  email: json['data']['email'],  sinds: json['data']['sinds']);  AuthenticationServices.*setBearerToken*(json['access\_token']);  *// print('Eind Werknemers create Bearer '  // '${AuthenticationServices.getBearerToken()}');* return werknemer;  }   static Future<Werknemer> *update*({  required int werknemerId,  required String naam,  required int functieId,  required String telefoon,  required String email,  required String sinds,  }) async {  *// print('Start Werknemers update Bearer '  // '${AuthenticationServices.getBearerToken()}');* final response = await http.patch(  Uri.*parse*('$*\_baseApi*/werknemers/$werknemerId'),  headers: <String, String>{  'Content-Type': 'application/json; charset=UTF-8',  'Authorization': 'Bearer ${AuthenticationServices.*getBearerToken*()}'  },  body: jsonEncode({  'id' : werknemerId,  "naam": naam,  "functie\_id": functieId,  "telefoon": telefoon,  "email": email,  "sinds": sinds  }),  );   *// print(response.statusCode);* if (response.statusCode != 200) {  Exception('Fout bij verwijderen werknemers ${response.statusCode}');  }   final json = jsonDecode(response.body);   *// print('Json Werknemers update $json');* final werknemer = Werknemer(  *// id: json['data']['id'], NB: id is in laravel geen fillable* id: werknemerId,  naam: json['data']['naam'],  functieId: json['data']['functie\_id'],  telefoon: json['data']['telefoon'],  email: json['data']['email'],  sinds: json['data']['sinds']);   AuthenticationServices.*setBearerToken*(json['access\_token']);  *// print('Eind Werknemers update Bearer '  // '${AuthenticationServices.getBearerToken()}');* return werknemer;  }   static Future<void> *delete*({  required int werknemerId,  }) async {  *// print('Start Werknemers delete Bearer ${AuthenticationServices.getBearerToken()}');* final response = await http.delete(  Uri.*parse*('$*\_baseApi*/werknemers/$werknemerId'),  headers: <String, String>{  'Content-Type': 'application/json; charset=UTF-8',  'Authorization': 'Bearer ${AuthenticationServices.*getBearerToken*()}'  },  );   *// print(response.statusCode);* if (response.statusCode != 202) {  Exception('Fout bij verwijderen werknemers ${response.statusCode}');  }   final json = jsonDecode(response.body);  *// print('Json Werknemers Delete $json');* AuthenticationServices.*setBearerToken*(json['access\_token']);  *// print('Eind Werknemers GetAll Bearer ${AuthenticationServices.getBearerToken()}');* } } |

# Password Eisen

# Exception handling Database not available:

|  |
| --- |
| App\Exceptions\Handler |
| <?php  namespace App\Exceptions;  use Illuminate\Database\QueryException;  use Illuminate\Foundation\Exceptions\Handler as ExceptionHandler;  use Throwable;  class Handler extends ExceptionHandler  {      public function render($request, Throwable $exception)      {          if ($exception instanceof QueryException) {              if ($exception->getCode() == 200) {                  $content = [                      'success' => false,                      'data'    => null,                      'foutmelding' => 'Neem contact op met de service desk en meld de volgende code: ' . strval($exception->getCode())                  ];                  return response()->json($content, 500);              }          }          return parent::render($request, $exception);      }  …  } |
| NB: Laravel zal altijd zelf zorgen voor een exception. Zorg er wel voor dat je debug in de .env file op false hebt staan. Doe je dat niet, krijgt de gebruiker veel te veel informatie, wat serieuze beveiligingsissues met zich meebrengt. |