

Toegepaste Informatica
Design & Develop with Frameworks

Paper

FreshSaucyFoods

Team: 5

Abdelwahid El Mazghari, Iejan Massart, Bram Van Pevenage, Sami Hafiz, Yarne Valck, Mohamed Amine Kallachi

Tweede bachelor – Toegepaste informatica

Vandenbogaerde Hans

Academiejahr 2020-2021

Inhoudsopgave

FreshSaucyFoods..... 1

 Inleiding..... 3

 Micro-services..... 4

 Bestelling..... 4

 Persoon..... 12

 Front..... 17

 Bestelling..... 18

 Persoon..... 26

REST-resource..... 34

Besluit 35

Zelfreflectie 36

Inleiding

Bij deze opdracht hebben we eerst de theorie over REST en micro-services geleerd. Dan moesten we deze toepassen op onze case waar we al heel het jaar mee werken.

We moesten delen uit vorige stappen aanpassen denk dan aan het front van de pagina en de het bestelling deel omzetten naar een micro-service en ten slotte de database omzetten. Maar ook nieuwe delen maken zoals de micro-service voor Persoon en het front hiervan uit werken.

Dit deel was enorm gericht op iets programmatisch maken en er waren vele nieuwe technologieën die hierbij te zien kwamen.

Micro-services

De micro-services die wij gerealiseerd hebben is eentje van Bestelling waar we in vorige delen de logica al van hadden gemaakt en dan een nieuwe namelijk Persoon.

We hebben dit gedaan met het patroon van Decompose by Subdomain door het domein van de bestelling management en het domein van Personen Management.

Bestelling

Eerst laten we hier de main zien waar we al 2 bestellingen gaan seeden zodat er al iets in onze database staat hier is niet zoveel anders aan als vroeger.

```
package be.odisee.team5.userapi;

import ...

@SpringBootApplication(exclude = { SecurityAutoConfiguration.class })
//@EnableGlobalMethodSecurity
public class UserApiApplication {

    public static void main(String[] args) { SpringApplication.run(UserApiApplication.class, args); }

    /**
     * Seeding of the database
     * @param personRepository
     * @return
     */
    @Bean
    CommandLineRunner init(BestellingRepository bestellingRepository){
        Bestelling b = new Bestelling();
        b.setTitel("Eerste bestelling");
        b.setAantallliterBesteld(10);
        b.setDatumStartproductie(LocalDate.now());
        b.setVooruitgang("In Productie");
        b.setVoorafAfgesprokenEindDatum(LocalDate.now().plusMonths(1));

        Bestelling b2 = new Bestelling();
        b.setTitel("Tweede bestelling");
        b.setAantallliterBesteld(50);
        b.setDatumStartproductie(LocalDate.now());
        b.setVooruitgang("In Productie");
        b.setVoorafAfgesprokenEindDatum(LocalDate.now().plusMonths(2));

        return (evt) -> {bestellingRepository.save(b);};
    }
}
```

Vervolgens hebben we onze controller die wel al bestond maar we moesten REST maken. Waardoor we als we op een specifieke URL gingen dan konden PUT, GET, DELETE en POST acties konden uitvoeren en de controller ook wist wat hij moest uitvoeren.

```
package be.odisee.team5.fsfopdracht2.controllers;

import ...

@RestController
@CrossOrigin(origins = "http://localhost:8888", maxAge = 3600, allowCredentials = "true")
public class BestellingRestController {

    @Autowired
    protected FreshSaucyFoodsService freshSaucyFoodsService=null; // ready for dependency injection

    @RequestMapping("/bestelling.html")
    public String bestelling() { return "bestelling"; }

    // REST GET ... breng de toestand van bestaande resources van de server naar de client (lijst van objecten)

    @RequestMapping(value={"/bestellingen"},method=RequestMethod.GET)
    public @ResponseBody
    List<Bestelling> getBestellingen(){

        List<Bestelling> bestellingen = freshSaucyFoodsService.getBestellingen();
        return bestellingen;
    }

    // REST PUT ... breng de toestand van bestaande resource van de client naar de server

    @RequestMapping(value={"/bestelling/{id}"},method=RequestMethod.PUT)
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void putPersoon(@PathVariable("id") Integer id, @RequestBody Bestelling bestelling){

        freshSaucyFoodsService.prepareEntryDataToEdit(id);
    }
}
```

```
// REST DELETE ... hiermee wordt een resource verwijderd

@RequestMapping(value={"/bestelling/{id}"},method=RequestMethod.DELETE)
@ResponseStatus(HttpStatus.NO_CONTENT)
public void deletePersoon(@PathVariable("id") Integer id){

    freshSaucyFoodsService.deleteBestelling(id);
}

// REST POST ... hiermee wordt een resource gecreeerd

@RequestMapping(value={"/bestelling"},method=RequestMethod.POST)
@ResponseStatus(HttpStatus.CREATED)
public @ResponseBody Bestelling createPersoon(@RequestBody BestellingData bestelling, HttpServletResponse response)
    throws BindException {

    freshSaucyFoodsService.processBesteling(bestelling);
    return bestelling;
}
```

Ten laatste is er nog de klasse Bestelling, de Repository, de Service en de ServiceImpl maar deze zijn bijna niet veranderd sinds vorige opdracht.

```
package be.odisee.team5.userapi.domain;

import ...

/**
 * @author bramv
 * @version 1.0
 * @created 09-Mar-2021 3:56:55 PM
 */
@Data
@Entity
@Table()
@XmlRootElement(name="Bestelling")
public class Bestelling implements Serializable {

    @Column
    private int aantalLiterBesteld;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id=0L;

    @Column
    private LocalDate datumStartproductie;
    @Column
    private String duurProductie;
    @Column
    private String status;
    @Column
    private LocalDate voorafAfgesprokenEindDatum;
    @Column
    private String vooruitgang;
    @Column
    private String titel;

    public Bestelling(){}

    @ManyToOne
    public Persoon klant;

    public void finalize() throws Throwable { }
    public void addProefstaaltje(String beschrijving, String titel){ }
```

```

//Detail van een bestelling kunnen opvragen
@XmlElement(name = "bestellingDetail")
public String bestellingDetail() {
    String bestellingDetail = "Titel Bestelling: " + titel + "\n" + "Aantal liter besteld: " + aantalLiterBesteld;
    return bestellingDetail;
}

@XmlElement(name = "duurProductie")
public String getDuurProductie() { return duurProductie; }
@XmlElement(name = "voorafAfgesprokenEindDatum")
public LocalDate getEindDate() { return voorafAfgesprokenEindDatum; }

@XmlElement(name = "id")
public long getId() { return id; }
@XmlElement(name = "titel")
public String getTitel(){return titel;}
@XmlElement(name = "aantalLiterBesteld")
public int getLiterBesteld() { return aantalLiterBesteld; }

@XmlElement(name = "status")
public String getStatus() { return status; }

@XmlElement(name = "vooruitgang")
public String getVooruitgang() { return vooruitgang; }

public void setDuurProducte(String duur) { duurProductie = duur; }

public void setEindDatum(LocalDate eindDatum) { voorafAfgesprokenEindDatum = eindDatum; }

public void setLiterBesteld(int literBesteld) { aantalLiterBesteld = literBesteld; }

public void setStatus(String s) { status = s; }

public void setVooruitgang(String vooruit) { vooruitgang = vooruit; }

```

```

package be.odisee.team5.fsfopdracht2.dao;

import ...

public interface BestellingRepository extends CrudRepository<Bestelling, Long> {
    /**
     * Find bestelling by its unique number
     */
    public Bestelling getById(long id);

    public Bestelling findFirstByOrderByIdDesc();

    public Bestelling findById(long id);
}

```

```

package be.odisee.team5.fsfopdracht2.service;

import ...

public interface FreshSaucyFoodsService {

    public Object getObjectives();

    public BestellingData prepareNewBestellingData();

    public String processBesteling(BestellingData bestellingData);

    public List<Bestelling> getBestellingen();

    public void deleteBestelling(long id);

    public BestellingData prepareEntryDataToEdit(long id);

    public String processBestellingInplannen(BestellingData bestellingData);

    public String getAuthenticatedFullname();
}

```



```

package be.odisee.team5.fsfopdracht2.service;

import ...

@Slf4j
@Service
public class FreshSaucyFoodsServiceImpl implements FreshSaucyFoodsService {

    @Autowired
    private BestellingRepository bestellingRepository;

    @Autowired
    private PersonRepository personRepository;

    @Override
    public Object getObjectives() { return null; }

    @Override
    public BestellingData prepareNewBestellingData() {
        Bestelling lastBestelling = bestellingRepository.findFirstByOrderByIdDesc();
        return prepareBestellingData(lastBestelling);
    }

    private BestellingData prepareBestellingData(Bestelling bestelling){
        BestellingData bestellingData = new BestellingData();
        bestellingData.setTitel(bestelling.getTitel());
        bestellingData.setGewensteLeverdatum(bestelling.getVoorafAfgesprokenEindDatum().toString());
        bestellingData.setAantalLiter(bestelling.getAantalLiterBesteld());
        bestellingData.setVooruitgang(bestelling.getVooruitgang());
        try {
            bestellingData.setStartProductieDate(bestelling.getDatumStartproductie().toString());
        } catch (Exception e){

        }
        bestellingData.setId(bestelling.getId());
        return bestellingData;
    }
}

```

```

@Override
public String processBestelling(BestellingData bestellingData) {
    Bestelling bestelling;
    if (bestellingData.getId() == 0) {
        bestelling = new Bestelling();
        bestelling.setStatus("Aangemaakt");
        bestelling.setVooruitgang("Aangemaakt");
    }
    else {
        bestelling = bestellingRepository.findById( bestellingData.getId() );
        bestelling.setVooruitgang(bestellingData.getVooruitgang());
        try {
            bestelling.setDatumStartproductie(LocalDate.parse(bestellingData.getStartProductieDate(), DateTimeFormatter.ofPattern("yyyy-dd-MM")));
        }
        catch (Throwable e){
            bestelling.setDatumStartproductie(LocalDate.parse(bestellingData.getStartProductieDate(), DateTimeFormatter.ofPattern("yyyy-MM-dd")));
        }
    }
    bestelling.setStatus(bestellingData.getVooruitgang());
    bestelling.setAantalLiterBesteld(bestellingData.getAantalLiter());
    try{
        bestelling.setVoorafAfgesprokenEindDatum(LocalDate.parse(bestellingData.getGewensteLeverdatum(), DateTimeFormatter.ofPattern("yyyy-dd-MM")));
    }
    catch (Throwable e){
        bestelling.setVoorafAfgesprokenEindDatum(LocalDate.parse(bestellingData.getGewensteLeverdatum(), DateTimeFormatter.ofPattern("yyyy-MM-dd")));
    }
    bestelling.setTitel(bestellingData.getTitel());
    bestellingRepository.save(bestelling);
    return "bestelling:" + bestelling.getTitel();
}

@Override
public List<Bestelling> getBestellingen() {
    return (List<Bestelling>) bestellingRepository.findAll();
}

```

```

@Override
public void deleteBestelling(long id) {
    Bestelling bestelling = bestellingRepository.findById(id);
    bestellingRepository.delete(bestelling);
}

@Override
public BestellingData prepareEntryDataToEdit(long id) {

    Bestelling bestelling = bestellingRepository.findById(id);
    BestellingData bestellingDataData = prepareBestellingData(bestelling);
    bestellingDataData.setId(id);
    return bestellingDataData;
}

@Override
public String processBestellingInplannen(BestellingData bestellingData) {
    Bestelling bestelling = bestellingRepository.findById(bestellingData.getId());
    bestelling.setStatus("In Gepland");
    bestelling.setVooruitgang("In Gepland");
    bestelling.setDatumStartproductie(LocalDate.parse(bestellingData.getStartProductieDate(), DateTimeFormatter.ofPattern("yyyy-MM-dd")));
    bestellingRepository.save(bestelling);
    return "bestelling"+bestelling.getTitel();
}

private String getAuthenticatedUsername() {

    Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
    String currentPrincipalName = authentication.getName();
    return currentPrincipalName;
}

private Persoon findAuthenticatedPersoon() {

    String email = getAuthenticatedUsername();
    return personRepository.findPersoonByEmailadress(email);
}

@Override
public String getAuthenticatedFullname() {

    Persoon theUser = findAuthenticatedPersoon();
    return theUser.getVoornaam() + ' ' + theUser.getFamiliennaam();
}

```

Persoon

Hieronder gaan we de micro-service code van Persoon uitleggen welke helemaal nieuw is. Hier zijn we ook eerst al 2 personen gaan seeden zodat er al iets in onze database staat.

```
package be.odisee.team5.userapi;

import ...

@SpringBootApplication(exclude = { SecurityAutoConfiguration.class })
//@EnableGlobalMethodSecurity
public class UserApiApplication {

    public static void main(String[] args) {
        SpringApplication.run(UserApiApplication.class, args);
    }

    /**
     * Seeding of the database
     * @param personRepository
     * @return
     */
    @Bean
    CommandLineRunner init(PersonRepository personRepository){
        Persoon p = new Persoon();
        p.setRole("ROLE_USER");
        p.setNaam("Jef");
        p.setFamiliennaam("Lokers");
        p.setPassword("12345");
        p.setEmailadress("jef.lokers@hotmail.com");

        Persoon p2 = new Persoon();
        p2.setRole("ROLE_ADMIN");
        p2.setNaam("Tom");
        p2.setFamiliennaam("Berens");
        p2.setPassword("1");
        p2.setEmailadress("tom.berens@hotmail.com");

        return (evt) -> {personRepository.save(p);personRepository.save(p2);};
    }
}
```

Vervolgens de Controller die we ook REST hebben moeten maken natuurlijk hebben we ons kunnen baseren op deze van bestelling om de controller te maken.

```
package be.odisee.team5.userapi.controller;

import ...

@RestController
@CrossOrigin(origins = "http://localhost:8888", maxAge = 3600, allowCredentials = "true")
public class PersoonController {

    @Autowired
    PersoonService persoonService;

    @RequestMapping (path = "deletePersoon/{id}", method = RequestMethod.DELETE)
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void delete(@PathVariable("id") Integer id) { persoonService.deletePersoon(id); }

    @RequestMapping (path = "persoon/{id}", method = RequestMethod.GET)
    Persoon persoon (@PathVariable("id") Integer id) { return persoonService.getPersoonDetailsById(id); }

    @RequestMapping (path = "updatePersoon/{id}", method = RequestMethod.POST)
    Persoon persoon (@RequestBody Persoon persoon){
        persoonService.processPersoon(persoon);
        return persoon;
    }

    @RequestMapping (path = "personen", method = RequestMethod.GET)
    List<Persoon> personen () { return persoonService.getAllPersonen(); }

    @RequestMapping (path = "createPersoon", method = RequestMethod.POST)
    @ResponseStatus(HttpStatus.CREATED)
    public void createPersoon(@RequestBody Persoon persoon) { persoonService.processPersoon(persoon); }
}
```

Hierna hebben we de klasse Persoon

```
@Entity
@Data
@Table(name = "PERSONEN")
public class Persoon {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int persoonId;

    @Column(unique = true)
    private String emailadress;
    @Column
    private String familienaam;
    @Column
    private String password;
    @Column
    private int status;
    @Column
    private String voornaam;
    @OneToMany
    private List<Bestelling> bestellingen;
    @Column
    private String role;

    public Persoon(){

    }

    public long getId() { return persoonId; }

    public void setId(int id) { this.persoonId = id; }

    public String getNaam() { return voornaam; }

    public void setNaam(String voornaam) { this.voornaam = voornaam; }

    public double getSaldo() { return status; }

    public void setSaldo(int status) { this.status = status; }

    public void finalize() throws Throwable {

    }
```

Vervolgens de Repository en de service welke niet zo enorm groot zijn.

```
package be.odisee.team5.userapi.dao;

import ...

public interface PersoonRepository extends CrudRepository<Persoon, Integer> {
    public Persoon findPersoonByEmailadress(String emailAddress);
    public Persoon findById(int id);
}
```

```
package be.odisee.team5.userapi.service;

import ...

public interface PersoonService {
    public Persoon getPersoonDetailsById(int id);
    public int processPersoon(Persoon persoon);
    public void deletePersoon(int id);
    public List<Persoon> getAllPersonen();
}
```

En ten laatste de ServiceImpl die we tegenover deze van bestelling hebben kunnen afslanken door de nieuwe technieken te gebruiken maar hier wordt alles gebruikt om de logica waar te maken.

```
@Service
public class PersoonServiceImpl implements PersoonService{

    @Autowired
    PersoonRepository persoonRepository;

    @Override
    public Persoon getPersoonDetailsById(int id) {
        Persoon persoon = persoonRepository.findById(id);
        return persoon;
    }

    @Override
    public int processPersoon(Persoon persoon) {

        if(persoon.getPersoonId() == 0){
            //create new persoon
            persoonRepository.save(persoon);
        }
        else {
            //update an persoon
            Persoon persoon1 = persoonRepository.findById(persoon.getPersoonId());

            //mapping of persoon objecten
            persoon1.setEmailadress(persoon.getEmailadress());
            persoon1.setFamiliennaam(persoon.getFamiliennaam());
            persoon1.setPassword(persoon.getPassword());
            persoon1.setVoornaam(persoon.getVoornaam());
            persoon1.setRole(persoon.getRole());

            persoonRepository.save(persoon1);
        }
        return persoon.getPersoonId();
    }

    @Override
    public void deletePersoon(int id) { persoonRepository.deleteById(id); }

    @Override
    public List<Persoon> getAllPersonen() { return (List<Persoon>) persoonRepository.findAll(); }
}
```


Front

Hieronder zullen we de nieuwe fronts laten zien die we gemaakt hebben dit is gebeurd met de nieuwe technologie van VUE. We hebben een front-page dan een pagina voor bestelling, persoon en ten slotte nog een About.

Hieronder ziet u onze front-page of homepage die gebaseerd is op vorige oefening maar natuurlijk wel drastisch is veranderd in code.



```
<template>
  <div class="home">
    <h2>Fresh Saucy Foods</h2>
    
    <p>Welkom bij Fresh Saucy Foods!!</p>
  </div>
</template>

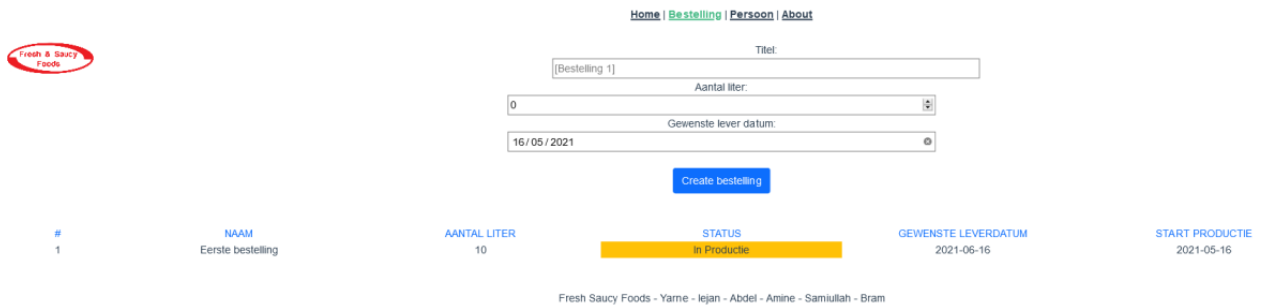
<script>
export default {
  name: "Home"
}
</script>

<style scoped>

</style>
```

Bestelling

Hieronder ziet u onze bestelling front de logica hiervan is niet veranderd maar de look en code erachter wel.



Home | **Bestelling** | Persoon | About

Titel: [Bestelling 1]

Aantal liter: 0

Gewenste lever datum: 16/05/2021

Create bestelling

#	NAAM	AANTAL LITER	STATUS	GEWENSTE LEVERDATUM	START PRODUCTIE
1	Eerste bestelling	10	In Productie	2021-06-16	2021-05-16

Fresh Saucy Foods - Yarne - Iejan - Abdel - Amine - Samiullah - Bram

In de logica wordt verwezen naar de Component MainForm waar eigenlijk het meeste instaat.

```
<template>

  <div id="bestelling">
    <div id="image">
      
    </div>
    <main-form/>
  </div>
</template>

<script>

import MainForm from '@components/MainForm.vue'

export default {
  name: "Bestelling",
  components: {
    MainForm
  }
}
</script>

<style scoped>

</style>
```

Met eerst het template gedeelte wat eigenlijk zorgt voor de html van de pagina dus al de tekst buttons, etc.

```
<template>
<div id="bestellingForm">
  <input type="hidden" id="id" name="id" value="0"/>

  <div class="form-group">
    <label for="title">Titel: </label><br><input id="title" name="title" type="text" style="width: 100%;" v-model="bestellingData.titel" placeholder="[Bestelling 1]" /><br>
    <span class="validationError" ></span>
  </div>

  <div class="form-group">
    <label for="aantalLiter">Aantal liter: </label><br><input id="aantalLiter" name="aantalLiter" type="number" style="width: 100%;" v-model="bestellingData.aantalLiterBesteld" /><br>
    <span class="validationError" ></span>
  </div>

  <div class="form-group">
    <label for="leverDatum">Gewenste lever datum: </label><br><input id="leverDatum" name="leverDatum" type="date" style="width: 100%;" v-model="bestellingData.voorafAfgesprokenEindDatum" /><br>
    <span class="validationError" ></span>
  </div>

  <div v-if="this.$route.params.id != null" >
    <label for="vooruitgang">Vooruitgang </label><br><input id="vooruitgang" name="vooruitgang" type="text" style="width: 100%;" v-model="bestellingData.vooruitgang"/>
  </div>

  <div v-if="this.$route.params.id != null" class="form-group">
    <label for="startProductie">Start productie datum </label><br><input id="startProductie" name="startProductie" type="date" style="width: 100%;" v-model="bestellingData.datumStartproductie" />
  </div>

  <div v-if="this.$route.params.id == null" class="form-group my-4">
    <button v-on:click="submitForm" class="btn btn-primary btn-md" name="submit">Create bestelling</button>

  </div>

  <div v-if="this.$route.params.id != null" class="form-group my-4">
    <button v-on:click="submitForm" class="btn btn-primary btn-md" name="submit">Update bestelling</button>
    <button v-on:click="deleteBestelling" class="btn btn-danger btn-md" name="delete">Delete bestelling</button>
    <button v-on:click="cancel" class="btn btn-warning btn-md" name="cancel">Cancel</button>

  </div>

</bestellingen-table> </bestellingen-table>
</div>
</template>
```

Hierna het script deel om de logica van de buttons te laten werken als er op wordt geklikt en zo je verwezen wordt naar de juiste URL.

In het eerste deel wordt er bij data() ook verwezen naar BestellingTable component dit is namelijk de tabel waar al onze bestellingen worden weergegeven.

```
<script>
import ...

export default {
  name: "MainForm",
  components : {
    BestellingenTable
  },
  data(){
    return {
      "bestellingData": {
        "id": 0,
        "titel": "",
        "aantalLiterBesteld" : 0,
        "voorafAfgesprokenEindDatum" : moment().format( date: "YYYY-MM-DD") ,
        "vooruitgang" : "",
        "datumStartproductie": moment().format( date: "YYYY-MM-DD"),
      },
      "message" : "Maak aub een bestelling aan",
      "componentKey" : 0,
    };
  },
}
```

Hier ziet u BestellingData met bijhorend script om de werking ervan uit te voeren.

```
<template>
<div v-if="bestellingen.length !== 0" class="table-striped well my-5">
  <div class="row">
    <div class="col-md-1 text-uppercase text-primary text-left font-weight-bold">#</div>
    <div class="col-md-2 text-uppercase text-primary text-left">Naam</div>
    <div class="col-md-2 text-uppercase text-primary text-left">Aantal liter</div>
    <div class="col-md-2 text-left text-primary text-uppercase">Status</div>
    <div class="col-md-2 text-left text-primary text-uppercase">Gewenste leverdatum</div>
    <div class="col-md-2 text-left text-primary text-uppercase">Start productie</div>
  </div>
  <div v-for="bestelling in bestellingen" class="row" v-bind:key="bestelling.id"
    v-on:click="showBestellingDetails(bestelling.id)" >
    <div class="col-md-1 text-left" >
      {{bestelling.id}}
    </div>
    <div class="col-md-2 text-left titeltoegevoegd" >
      {{bestelling.titel}}
    </div>
    <div class="col-md-2 text-left" >
      {{bestelling.aantalLiterBesteld}}
    </div>
    <div v-if="bestelling.vooruitgang === 'Klaar'" class="col-md-2 text-left bg-success" >
      {{bestelling.vooruitgang}}
    </div>
    <div v-if="bestelling.vooruitgang === 'Aangemaakt'" class="col-md-2 text-left bg-info" >
      {{bestelling.vooruitgang}}
    </div>
    <div v-if="bestelling.vooruitgang === 'Weiger'" class="col-md-2 text-left bg-danger" >
      {{bestelling.vooruitgang}}
    </div>
    <div v-if="bestelling.vooruitgang === 'In Productie'" class="col-md-2 text-left bg-warning" >
      {{bestelling.vooruitgang}}
    </div>
    <div v-else class="col-md-2 text-left" >
      {{bestelling.vooruitgang}}
    </div>
    <div class="col-md-2 text-left" >
      {{bestelling.eindDate}}
    </div>
    <div class="col-md-2 text-left" >
      {{bestelling.datumStartproductie}}
    </div>
  </div>
</div>
```

```

<script>
import axios from "axios";

async function getBestellingen(){
  const url = 'http://localhost:8080/bestellingen';
  let result = (await axios.get(url , {withCredentials : true})).data;
  return result;
}

export default {
  name: "BestellingenTable",

  data () {
    return {
      "bestellingen" : [],
    }
  },
  created() {
    var result = getBestellingen();
    result.then((res) => this.bestellingen = res)
  },
  methods: {
    showBestellingDetails(bestellingId) {
      window.location.href = "/bestelling/"+bestellingId;
    }
  },
}
</script>

<style scoped>

</style>

```

Vervolgens de rest van het script van de MainForm.

```
created() {  
  let url;  
  if(this.$route.params.id != null){  
    url = 'http://localhost:8080/bestellingen';  
    this.message = "Wijzig, verwijder of plan een bestelling in - of annuleer";  
  }  
  else {  
    url = 'http://localhost:8080/bestellingen';  
  }  
  
  axios.get(url , {withCredentials : true})  
    .then((response) => {  
      if(url !== 'http://localhost:8080/bestellingen'){  
        this.persoonData = response.data;  
      }  
      console.log(this.persoonData);  
      if(response.status === 204){  
        this.newForm();  
      }  
    })  
    .catch((error) => {  
      if (error.response.status === 403) {  
        // entry forbidden for user  
        this.message = "bestelling met id "+this.$route.params.entryId+" is niet gevonden";  
        this.newForm();  
      } else {  
        console.log(error.message);  
      }  
    })  
  },  
  method: 'get'  
}
```

```

methods : {
  //update en creëren in een
  submitForm : function () {
    let url = '';

    if(this.$route.params.id != null){
      url = "http://localhost:8080/bestelling/" + this.$route.params.id;
    }
    else {
      url = "http://localhost:8080/createBestelling";
    }
    const headers = {
      withCredentials: true
    };
    console.log(this.bestellingData);
    axios.post(url, this.bestellingData, headers)
      .then( (response) => {
        this.message = response.data;
        this.componentKey += 1;

        if (this.bestellingData.id !== 0) {
          this.bestellingData.id = 0; // klaar voor nieuwe entry nu
          this.newForm();
        }
      })
      .catch(function (error) {
        console.log(error)
      });
  },
}

```



```

deleteBestelling : function () {
  const url = "http://localhost:8080/bestelling/" + this.$route.params.id;
  const headers = {
    withCredentials: true
  };

  axios.delete(url, this.bestellingData, headers)
    .then( (response) => {
      this.message = response.data;
      this.componentKey += 1;
      this.newForm();
    })
    .catch(function (error) {
      if (error.response.status === 403) {
        this.message = "bestelling met id "+this.$route.params.id+" is verboden";

        this.newForm();
      } else {
        console.log(error.message);
      }
    });
},
cancel : function(){
  window.location.href = "/bestelling";
},
newForm: function () {
  // redirect for newEntryData but after some delay
  setTimeout( handler: () => this.$router.push({ name : ''}, () => { this.$router.go() } ), timeout: 1200);
}
}

```


/script>

style scoped>

/style>

Persoon

Vervolgens het front van ons nieuw deel namelijk Persoon we hebben hierop ons weer kunnen baseren op Bestellingen waardoor de lay-out redelijk hetzelfde is wat we ook wouden omdat dit zo overzichtelijk en duidelijk is wat je moet doen.



[Home](#) | [Bestelling](#) | [Persoon](#) | [About](#)

Voor naam:


Familie naam

Email adres:

Create persoon

#	VOORNAAM	FAMILIE NAAM	EMAIL ADRES
1	Jef	Lokers	jef.lokers@hotmail.com
2	Tom	Berens	tom.berens@hotmail.com

Fresh Saucy Foods - Yarne - Iejan - Abdel - Amine - Samiullah - Bram



[Home](#) | [Bestelling](#) | [Persoon](#) | [About](#)

Voor naam:

Familie naam

Email adres:

Update persoon Delete persoon Cancel

#	VOORNAAM	FAMILIE NAAM	EMAIL ADRES
1	Jef	Lokers	jef.lokers@hotmail.com
2	Tom	Berens	tom.berens@hotmail.com

Fresh Saucy Foods - Yarne - Iejan - Abdel - Amine - Samiullah - Bram

Hierin ziet u dat er weer wordt verwezen naar PersonenForm zoals bij bestelling naar MainForm.

```
<template>

  <div id="persoon">
    <div id="image">
      
    </div>
    <personen-form/>
  </div>
</template>

<script>
import PersonenForm from "../../components/PersonenForm";
export default {
  name: "Persoon",
  components: {PersonenForm}
}
</script>

<style scoped>

</style>
```

Deze PersonenForm is hetzelfde principe als bestelling. Met hieronder eerste het template gedeelte.

```
<template>
<div id="bestellingForm">
  <input type="hidden" id="id" name="id" value="0"/>

  <div class="form-group">
    <label for="title">Voor naam: </label><br><input id="title" name="voornaam" type="text" style="..." v-model="persoonData.voornaam" placeholder="[Jef]" /><br>
    <span class="validationError" ></span>
  </div>
  <div class="form-group">
    <label for="aantalLiter">Familie naam</label><br><input id="aantalLiter" name="familienaam" type="text" style="..." v-model="persoonData.familienaam" placeholder="[Lokers]" /><br>
    <span class="validationError" ></span>
  </div>
  <div class="form-group">
    <label for="leverDatum">Email adres: </label><br><input id="leverDatum" name="emailadress" type="email" style="..." v-model="persoonData.emailadress" placeholder="[jef.lokers@test.com]" />
    <span class="validationError" ></span>
  </div>

  <div v-if="this.$route.params.persoonId === undefined" class="form-group my-4">
    <button v-on:click="submitForm" class="btn btn-primary btn-md" name="submit">Create persoon</button>
  </div>
  <div v-if="this.$route.params.persoonId !== undefined" class="form-group my-4">
    <button v-on:click="submitForm" class="btn btn-primary btn-md" name="submit">Update persoon</button>
    <button v-on:click="deletePersoon" class="btn btn-danger btn-md" name="delete">Delete persoon</button>
    <button v-on:click="cancel" class="btn btn-warning btn-md" name="cancel">Cancel</button>
  </div>

  <personen-tabel></personen-tabel>
</div>
</template>
```

Vervolgens het script gedeelte beginnend met weer een verwijzing naar PersonenTabel die zoals bij bestelling verwijst naar een tabel van waar de personen in komen.

```
<script>
import ...
export default {
  name: "PersonenForm",
  components: {PersonenTabel},
  data(){
    return {"persoonData" : {
      "persoonId": 0,
      "emailadress" : "",
      "familienaam" : "",
      "password" : "",
      "status" : "",
      "voornaam" : "",
      "role" : ""
    }},
    "message" : "Maak aub een persoon aan",
    "componentKey" : 0,
  };
},
```

Hieronder dan PersonenTabel zoals eerder gezegd sterk lijkend op BestellingTabel.

```
<template>
  <div v-if="personen.length !== 0" class="table-striped well my-5">
    <div class="row">
      <div class="col-md-1 text-uppercase text-primary text-left font-weight-bold">#</div>
      <div class="col-md-2 text-uppercase text-primary text-left">Voornaam</div>
      <div class="col-md-2 text-uppercase text-primary text-left">Familie naam</div>
      <div class="col-md-2 text-left text-primary text-uppercase">Email adres</div>
    </div>
    <div v-for="persoon in personen" class="row" v-bind:key="persoon.persoonId"
      v-on:click="showPersoonDetails(persoon.persoonId)" >
      <div class="col-md-1 text-left" >
        {{persoon.persoonId}}
      </div>
      <div class="col-md-2 text-left titeltoegevoegd" >
        {{persoon.voornaam}}
      </div>
      <div class="col-md-2 text-left" >
        {{persoon.familienaam}}
      </div>
      <div class="col-md-2 text-left" >
        {{persoon.emailadress}}
      </div>
    </div>
  </div>
</template>
```

En dan zijn bijhorend script.

```
<script>
import axios from "axios";

async function getPersonen(){
  const url = 'http://localhost:8081/personen';
  let result = (await axios.get(url , {withCredentials : true})).data;

  return result;
}

export default {
  name: "PersonenTabel",
  data () {
    return {
      "personen" : [],
    }
  },
  created() {
    var result = getPersonen();
    result.then((res) => this.personen = res)
    console.log(this.personen);
  },
  methods: {
    showPersoonDetails(persoonId) {
      window.location.href = "/persoon/" + persoonId;
    }
  },
}
]

</script>

<style scoped>

</style>
```

Vervolgens weer de vervolging van het script van PersonenForm om de werking van alle buttons en de redirects te laten gebeuren.

```
created() {
  let url;
  if(this.$route.params.persoonId !== null){
    url = 'http://localhost:8081/persoon/' + this.$route.params.persoonId;
    this.message = "Wijzig, verwijder persoon of annuleer";
  }
  else {
    url = 'http://localhost:8081/personen';
  }

  console.log(url)

  axios.get(url , {withCredentials : true})
    .then((response) => {
      if(url !== 'http://localhost:8081/personen'){
        this.persoonData = response.data;
      }
      if(response.status === 204){
        this.newForm();
      }
    })
    .catch((error) => {
      if (error.response.status === 403) {
        // entry forbidden for user
        this.message = "persoon met id "+this.$route.params.persoonId+" is niet gevonden";
        this.newForm();
      } else {
        console.log(error.message);
      }
    })
  },
```

```

methods : {
  //update en creëren in een
  submitForm : function () {
    let url = "";
    if(this.$route.params.persoonId != null){
      url = "http://localhost:8081/updatePersoon/" + this.$route.params.persoonId;
    }
    else {
      url = "http://localhost:8081/createPersoon"
    }
    const headers = {
      withCredentials: true
    };

    console.log(url);

    axios.post(url, this.persoonData, headers)
      .then( (response) => {
        this.message = response.data;
        this.componentKey += 1;

        if (this.persoonData.persoonId !== 0) {
          this.persoonData.persoonId = 0; // klaar voor nieuwe entry nu
          this.newForm();
        }
      })
      .catch(function (error) {
        console.log(error)
      });
    window.location.href = "/persoon";
  },

```



```

deletePersoon : function () {
  const url = "http://localhost:8081/deletePersoon/" + this.$route.params.persoonId;
  const headers = {
    withCredentials: true
  };

  axios.delete(url, this.persoonData, headers)
    .then( (response) => {
      this.message = response.data;
      this.componentKey += 1;
      this.newForm();
    })
    .catch(function (error) {
      if (error.response.status === 403) {
        this.message = "persoon met id "+this.$route.params.persoonId+" is verboden";
        this.newForm();
      } else {
        console.log(error.message);
      }
    });
  window.location.href = "/persoon";
},
cancel : function(){
  window.location.href = "/persoon";
},
newForm: function () {
  // redirect for newEntryData but after some delay
  setTimeout( handler: () => this.$router.push({ name : '' }, () => { this.$router.go() } ), timeout: 1200);
}
}
}
</script>

<style scoped>

</style>

```

REST-resource

Het aanbieden van een REST resource is niet gebeurd bij ons we waren wel tot de idee fase gekomen met team Quatra.

Het idee dat hieruit volgde is dat we klant bij hen zouden zijn en dus ook olie zouden kunnen bestellen. Maar dit is jammer genoeg niet verder uitgewerkt dus er is ook geen resource van een ander team die geconsumeerd is.

Besluit

Ons besluit over deze oefening is dat de theorie en het nut ervan zeker interessant waren om mee te nemen.

Maar dat het samenvallen met vele andere opdrachten en het vele programmeren dat aan deze opdracht verbonden was voor vele onder ons moeilijk was en we een beetje het analyse aspect misten.

Zelfreflectie

Iejan: Ik vind persoonlijk de theorie achter deze opdracht een van de interessantste en nuttigste dat we dit jaar gezien hebben.

De opdracht zelf was redelijk moeilijk omdat er veel nieuwe technologieën te zien waren. Ook waren er veel deadlines en lopende opdrachten van andere vakken waardoor deze opdracht niet 100% af geraakt is.

Deze opdracht vind ik ook een heel goede training voor de geïntegreerde opdracht succesvol te kunnen afwerken.

Amine: Ik vind dat deze opdracht redelijk goed ging. Al vond ik wel dat ik minder bijdrage heb geleverd dan gewoonlijk. Al bij al ging het goed en is deze opdracht tot een goed eind gebracht.

Nu rest er ons alleen de geïntegreerde opdracht nog, en ik denk wel dat die ook goed gaat lukken en kunnen we dit allemaal tot een goed einde brengen.

Abdelwahid: Ik vond deze opdracht stukken vlotter dan de vorige opdrachten waardoor ik denk dat we ook meer zullen scoren.

Voor een groot deel van de laatste weken voelde ik me niet goed, maar heb mijn time-off proberen te minimaliseren. De leerstof inhalen ging niet zo simpel als ik dacht maar heb toch mijn best gedaan. Hoe dan ook heb ik veel bijgeleerd!

Bram: Persoonlijk was het een leuke opdracht dat zeer relevant is met hoe het er in de bedrijf wereld aan toegaat. Door de 3 meetings niet meer te verplichten was er redelijk weinig contact tussen ons in de groep. Het was ook jammer dat bij een paar mensen intellij niet wou werken.

Yarne: Deze opdracht was best uitdagend omdat we zelf veel moesten programmeren en allerlei nieuwe technieken moesten toepassen. In het algemeen vond ik het wel tof of samen met de andere te kunnen nadenken en programmeren.

Opdracht 1 en 2 gingen goed en vlot maar tijdens opdracht drie merkte ik wel dat we het nogal druk hadden door andere vakken waardoor we niet alles zoals we wouden hebben kunnen afwerken.

Sami: Ik vond deze opdracht een van de interessantere die we al gehad hebben alleen waren er veel andere opdrachten van andere vakken waardoor ik me niet 100% kon inzetten voor deze opdracht al heb ik geprobeerd zoveel mogelijk mijn steentje bij te dragen en ik denk dat dit redelijk gelukt is.