


Mikele Lemmens (202291269)

-  Web Services: GITHUB URL
 - <https://github.com/Web-IV/2324-webservices-MikeleLemmens1.git>

Logingegevens

De onderstaande gegevens worden gebruikt om in te loggen als gezinslid in de app

- Gebruikersnaam/e-mailadres: mikele.lemmens@hotmail.com
- Wachtwoord: '12345678'

DATABANK Credentials:

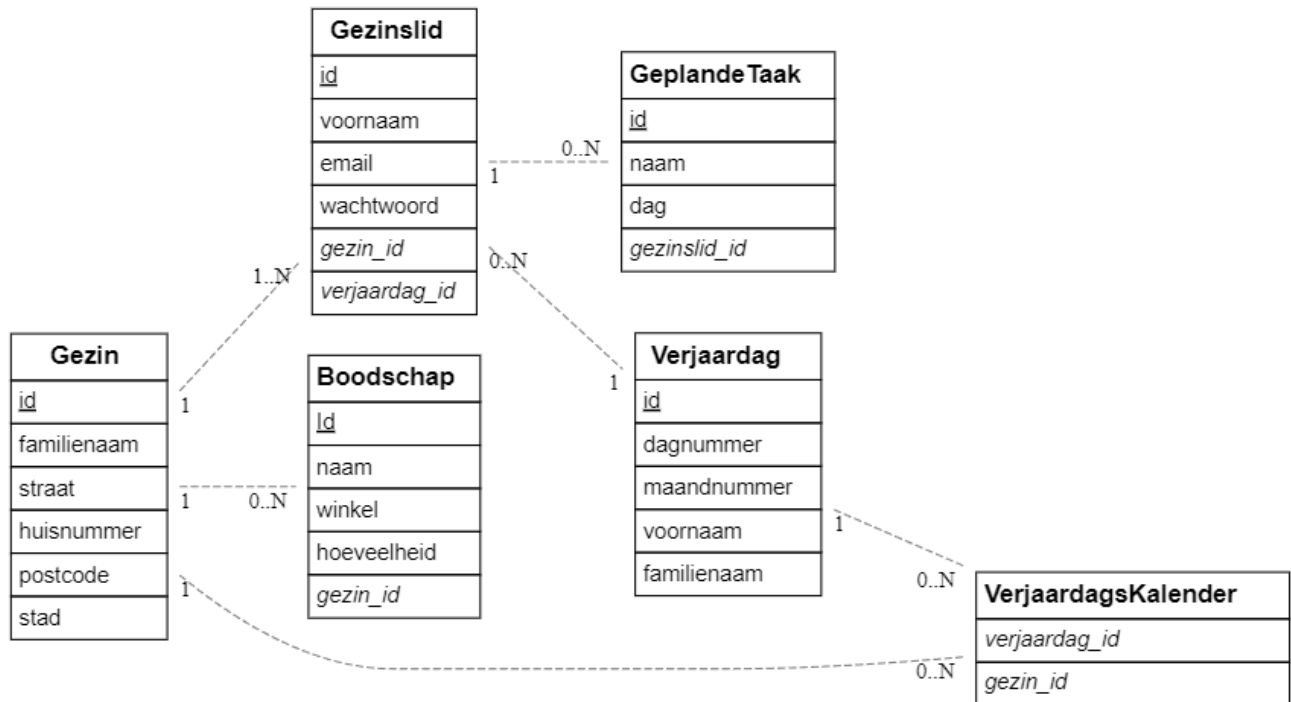
- Databank: 291269ml
- Gebruikersnaam: 291269ml
- Wachtwoord: SnKQ1eNSvgYaCkzOFFFU
- Host: vichogent.be
- Poort: 40043

Projectbeschrijving

Een gezin moet met bijzonder veel zaken rekening houden. Iedere dag lijkt er een waslijst aan taken en herinneringen te zijn die snel overweldigend kunnen worden. Het bijhouden van zulke zaken in een gebruiksvriendelijk overzicht kan al een deel van het werk overnemen. Ik maak de API die kan worden gebruikt om dit mogelijk te maken.

Een gezin is de groep van gebruikers. Het bestaat uit **gezinsleden** (dit zullen de uiteindelijke gebruikers worden), en heeft een lijst met **geplande taken**, een **boodschappenlijst** en een **verjaardagskalender**. Hier nog extra info per entiteit:

- Een gezin heeft een *familienaam*, adres (bestaande uit *straat*, *huisnummer*, *postcode* en *stad*), een of meerdere gezinsleden, een verjaardagskalender, boodschappenlijst en een *id*.
- Ieder gezinslid heeft een *id*, *voornaam*, *email*, een *wachtwoord* en een *verjaardagsld*. Ieder gezinslid is een geregistreerde gebruiker.
- Een geplande taak is een taak die toebehoort aan een gezinslid en dient te worden uitgevoerd op een bepaalde dag. De gezinsplanner kan een overzicht geven van alle taken die op een bepaalde dag of week zijn ingepland. Er is een *taakId*, *naam*, *dag* en *gezinslidId* (de uitvoerder).
- Een boodschap wordt gekenmerkt door zijn *id*, en heeft verder een *naam* (of beschrijving), *winkel* en *hoeveelheid*. De winkel en hoeveelheid zijn optioneel. Iedere boodschap heeft een *gezinId*, deze stelt de verwijzing voor naar het boodschappenlijstje van het gezin waartoe de boodschap behoort.
- De verjaardagen worden verzameld in een verjaardagskalender en bevatten een *id*, *dagnummer*, *maandnummer*, *voornaam* en *familienaam*. Ook de verjaardagen van de gezinsleden worden hierin opgenomen, maar de meerderheid van de verjaardagen zullen van mensen zijn die niet tot het gezin behoren.



API calls

Gezinsleden

- GET /api/gezinsleden: alle gezinsleden ophalen
- GET /api/gezinsleden/:id: gezinslid ophalen op id
- POST /api/gezinsleden/login: inloggen als gezinslid
- POST /api/gezinsleden/register: registreren als gezinslid
- PUT /api/gezinsleden/:id: gezinslid updaten
- DELETE /api/gezinsleden/:id: gezinslid verwijderen

Gezinnen

- GET /api/gezinnen: alle gezinnen ophalen
- GET /api/gezinnen/id: gezin ophalen op id
- POST /api/gezinnen/: nieuw gezin maken
- PUT /api/gezinnen/:id: gezin wijzigen
- DELETE /api/gezinnen/:id: gezin verwijderen

Boodschappen

- GET /api/boodschappen: alle boodschappen ophalen
- GET /api/boodschappen/:id: Boodschap ophalen op id
- GET /api/boodschappen?winkel=WINKEL&gezin=GEZIN_ID: Boodschappen van een gezin ophalen op winkel
- POST /api/boodschappen: Boodschap maken
- PUT /api/boodschappen/:id: Boodschap wijzigen
- DELETE /api/boodschappen/:id: Boodschap verwijderen

GeplandeTaak

- `GET /api/geplande_taken`: alle geplande taken ophalen, m.u.v. diegene die in het verleden liggen.
- `GET /api/geplande_taken?dag=YYYY-MM-DD`: alle geplande taken van een bepaalde dag ophalen
- `GET /api/geplande_taken/:id`: alle geplande taken van een bepaald gezinslid ophalen
- `POST /api/geplande_taken`: Taak maken
- `PUT /api/geplande_taken/:id`: Taak wijzigen
- `DELETE /api/geplande_taken/:id`: Taak verwijderen

Verjaardag

- `GET /api/verjaardagen`: alle verjaardagen ophalen
- `GET /api/verjaardagen/:id`: verjaardag ophalen op id
- `POST /api/verjaardagen`: verjaardag maken
- `PUT /api/verjaardagen/:id`: verjaardag wijzigen
- `DELETE /api/verjaardagen/:id`: verjaardag verwijderen

Behaalde minimumvereisten

Web Services

- **data laag**
 - ☒ voldoende complex (meer dan één tabel, 2 een-op-veel of veel-op-veel relaties)
 - ☒ één module beheert de connectie + connectie wordt gesloten bij sluiten server
 - ☒ heeft migraties - indien van toepassing
 - ☒ heeft seeds
- **repository laag**
 - ☒ definieert één repository per entiteit (niet voor tussentabellen) - indien van toepassing
 - ☒ mapt OO-rijke data naar relationele tabellen en vice versa - indien van toepassing
- **servicelaag met een zekere complexiteit**
 - ☒ bevat alle domeinlogica
 - ☒ bevat geen SQL-queries of databank-gerelateerde code
- **REST-laag**
 - ☒ meerdere routes met invoervalidatie
 - ☒ degelijke foutboodschappen
 - ☒ volgt de conventies van een RESTful API
 - ☒ bevat geen domeinlogica
 - ☒ geen API calls voor entiteiten die geen zin hebben zonder hun ouder (bvb tussentabellen)
 - ☒ degelijke autorisatie/authenticatie op alle routes
- **algemeen**
 - ☒ er is een minimum aan logging voorzien
 - ☒ een aantal niet-triviale integratietesten (min. 1 controller $\geq 80\%$ coverage)
 - ☐ minstens één extra technologie

- ☒ maakt gebruik van de laatste ES-features (async/await, object destructuring, spread operator...)
- ☒ duidelijke en volledige README.md
- ☒ volledig en tijdig ingediend dossier en voldoende commits

Projectstructuur

Ik heb de structuur gevolgd zoals in de voorbeeldapplicatie. Omdat ik wat tijd nodig had om alle onderdelen te begrijpen heb ik steeds de voorbeeldapplicatie gebruikt in de les om de getoonde zaken te kunnen reproduceren. Bij het implementeren in mijn eigen project heb ik bv andere domeinlogica toegevoegd, gebruik gemaakt van een tussentabel en alle endpoints voorzien van alle CRUD-operaties en testen.

Ik heb mijn repo opgesplitst in 2 belangrijke branches, de MAIN-branch en de Auth-branch. De main branch omvat alle endpoints, testen en deployment, maar alles omtrent autorisatie en authenticatie heb ik opgesplitst omdat ik de testen niet werkende kreeg. Ik heb de users voorzien als gezinsleden die een belangrijk onderdeel zijn van mijn database, en door de testen goed te proberen krijgen ben ik op een dwaalspoor terecht gekomen. Ik heb alle endpoints voorzien van authenticatie en autorisatie, maar krijg de testen niet werkende.

Extra technologie

Ik heb vooral ingezet op het goed krijgen van mijn endpoints en testen. Doordat ik enerzijds veel testen heb geschreven, alsook verloren geraakt ben in de autorisatie ben ik er niet aan toe gekomen om extra technologieën te voorzien. Ik ben wel enige tijd genomen om de documentatie te lezen van Auth0 en de cookbook mapper package, maar doordat mijn applicatie met autorisatie niet lijkt te werken ben ik niet meer aan de implementatie toe geraakt.

Testresultaten

In de main-branch zitten alle belangrijke testen (maar zonder autorisatie). Hiermee heb ik kunnen verzekeren dat de logica goed zit en dat de statuscodes en teruggegeven objecten de juiste zijn voor alle scenario's:

```
PS C:\Users\Mikel\Web Services Project\2324-webservices-MikeleLemmens1> yarn test
yarn run v1.22.19
$ env-cmd -f .env.test jest --runInBand
PASS __tests__/rest/gezinslid.spec.js
PASS __tests__/rest/boodschap.spec.js
PASS __tests__/rest/verjaardag.spec.js
PASS __tests__/rest/geplandeTaak.spec.js
PASS __tests__/rest/gezin.spec.js
PASS __tests__/rest/health.spec.js

Test Suites: 6 passed, 6 total
Tests: 100 passed, 100 total
Snapshots: 0 total
Time: 4.818 s, estimated 5 s
```

Gekende bugs

De grootste bugs zitten hem in bv. het opzoeken van een gezin. Aanvankelijk wou ik een id kunnen meegeven en alle gezinsleden tonen van dat gezin, maar ik had er geen rekening mee gehouden dat dit in autorisatie mijn logica zou verdraaien. Omdat getByld ook gebruikt werd in POST en PUT heb ik deze logica

vereenvoudigd, wat oorspronkelijk niet de bedoeling was. Dit geldt ook voor geplande taken (get /:id had alle geplande taken van een gezinslid moeten geven).

Ik hoop dat deze oplevering voldoende is om te kunnen deelnemen aan het examen. Mijn applicatie werkt niet, maar ik heb er altijd voor gezorgd dat ik zo goed mogelijk begreep wat ik overnam en implementeerde. Ik heb redelijk veel zaken ontdekt bij het debuggen, en heb het gevoel over de nodige kennis te beschikken om een kwalitatief product te kunnen opleveren.