Overdrachtsdocument RIGS-Employee App:

GitHub Repositories:

De GitHub-repositories kunnen gevonden worden op onze GitHub organisatie:

https://github.com/horeca-robot/

Hierin zitten onze repositories:

- Front-end
- Back-end
- Server Setup
- Web socket server

Inhoudsopgave

Achtergrondinformatie:

Tijdens de ontwikkelingsperiode voor het RIGS-project zijn er met 3 verschillende groepen gewerkt aan één grote applicatie. In totaal zijn er 3 losse applicaties. Er is een applicatie die functionaliteit levert voor klanten van een restaurant, een applicatie die functionaliteit levert voor medewerkers van een restaurant, en een applicatie die functionaliteit levert voor het onderhouden van de informatie die het restaurant toont. Dit document beschrijft de functionaliteit en het ontwerp van de medewerkers applicatie.

De medewerkers applicatie biedt de volgende functionaliteit aan: Het inzien, aanpassen en printen van bestellingen waar nodig, het aansturen van de robots, het inzien van de locaties van de robot, het ontvangen van klantverzoekjes, en het inzien van producten.

Requirements

De medewerkers app heeft de volgende requirements:

- Employees (when logged into an administrator account) must be able to add items to, edit or remove from the menu.
- An employee can see a (schematic) map with the tables in the restaurant.
    - On this map the locations and routes of the robots are displayed.
- An employee must be able to summon the robot(s) when food needs to be delivered to a table.
- The robot must have an emergency stop
- The app must show certain information about the robot:
    - Battery percentage.
    - Location in the restaurant.
    - Robot Status (Busy, available, charging, etc.)
    - Should be able to show multiple robots
- There are separate roles for accounts, regular employee, or administrator.
- There must be an easy overview of all the orders that are placed.
    - Oldest and uncompleted first (can be changed with filters)
- Payment status for every order (paid or unpaid)
- Employees need to be able to see if a customer wants service (at a specific table).
- Needs to be able to export all orders and transactions from any period.
- It needs to be easy for the user to switch between map and order overview mode.
    - Both views need all the possibilities for checking (and changing) orders.
- Orders can be split up into multiple part-orders when the amount of food does not fit onto the in one go.
- House style (Logo, colors, and information of the restaurant) can be configured at any time.

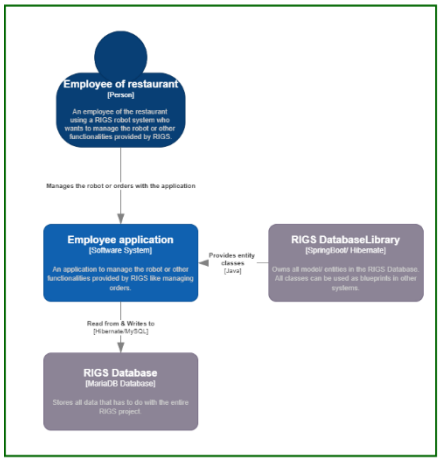
C3 model applicatie

Het onderstaande diagram schetst een beeld van de applicatie architectuur. In het contextdiagram kan worden gezien uit welke componenten de applicatie bestaat. In het midden staat onze bartender app. Deze app maakt gebruik van een algemene database library. Deze library wordt gebruikt om voor alle teams dezelfde datastructuur te garanderen. De applicatie maakt ook nog gebruik van een relationele MariaDB database om de data op te slaan. Er is gekozen voor een relationele database omdat onze data vele relaties bevat. Ook is er specifiek voor MariaDB gekozen omdat dat open source is, en wij daarom ons geen zorgen hoeven te maken over licenses.

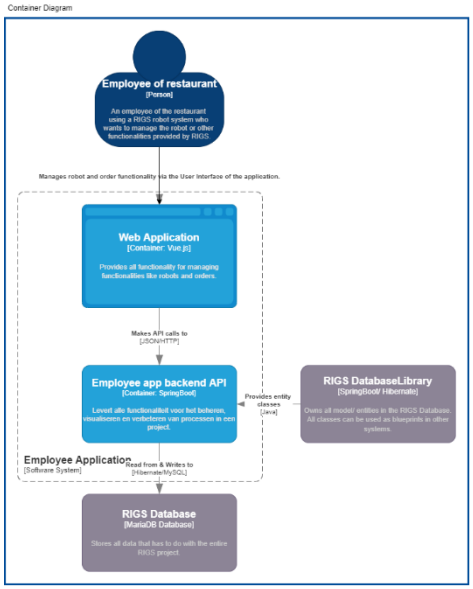
In het container diagram kan gezien worden uit welke onderdelen onze applicatie bestaat. Wij maken gebruik van een frontend webapplicatie, en een backend java applicatie. Voor de frontend is gekozen om Vue.js te gebruiken. Deze keuze is gemaakt omdat dit framework het maken van dashboard georiënteerde applicaties makkelijker maakt. Vue.js gebruikt een two-way-binding systeem waardoor wij data direct aan objecten kunnen koppelen. Verder is het ontwikkelen van een complexe frontend applicatie zeer makkelijk in dit framework omdat het een overzichtelijke structuur heeft. Voor de backend is gekozen om spring-boot te gebruiken. Wij hebben hiervoor gekozen omdat dit ons een aantal helpvolle tools heeft gegeven voor tijdens het ontwikkelen. Zo maken wij veel gebruik van de dependency injection functionaliteit en de automatische JPA-database configuratie. Door gebruik te maken van deze technieken is het aanpassen en uitbreiden van deze applicatie vrij gemakkelijk.

In het component diagram is de architectuur van onze backend applicatie te zien. Wij maken gebruik van de volgende flow: Controller <--> Service <--> Repository. De controller ontvangt een HTTP-request en kan hier wat data uit halen die daarna naar de service doorgestuurd kan worden. Een service ontvangt data en kan hier validatie op doen, of kan deze data aanpassen. De service gebruikt daarna een repository om aanpassingen door te sturen naar de database. Wij hebben voor deze flow gekozen omdat ieder elke soort actie zijn gedeelte krijgt. De controller hoeft alleen maar data te verzamelen en terug te sturen. Al het belangrijke werk gebeurt in de services. Deze services kunnen daarom ook allemaal netjes ge-unit test worden om zo de werking van de applicatie te garanderen. De repositories hoeven alleen data op te slaan. Doordat al deze soort acties afgezonderd van elkaar zijn zou het mogelijk moeten zijn om dit in de toekomst allemaal op aparte servers te runnen, dit zou gedaan kunnen worden om een hoger aantal requests te kunnen ontvangen.

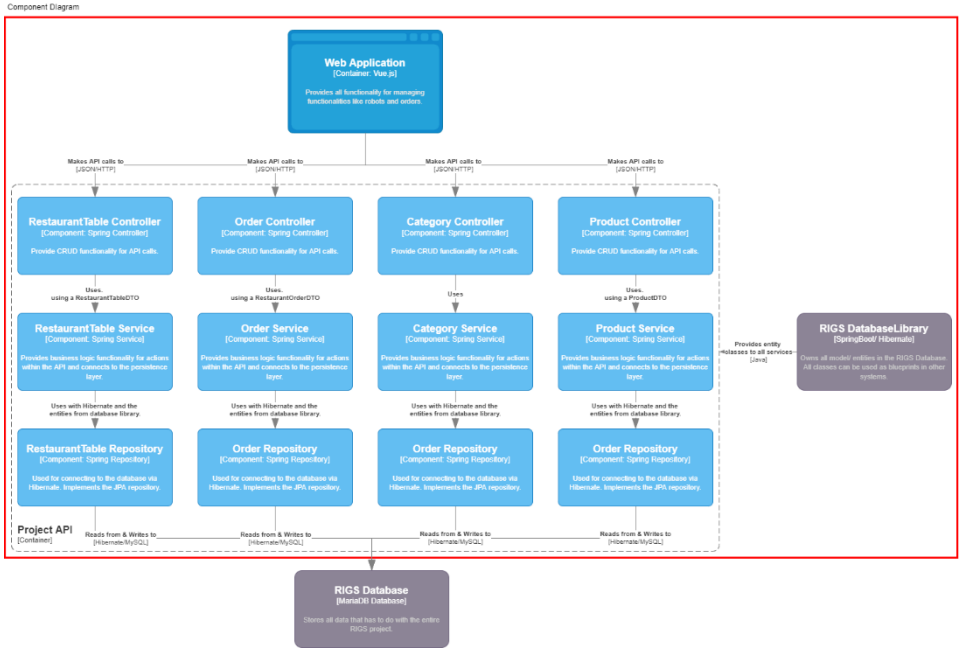
System Context Diagram



Container Diagram

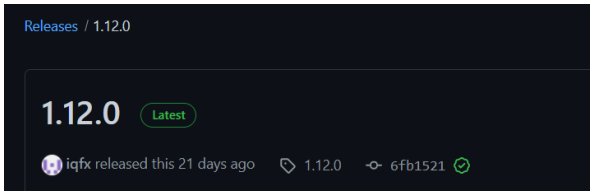


Component Diagram



Database en ERD:

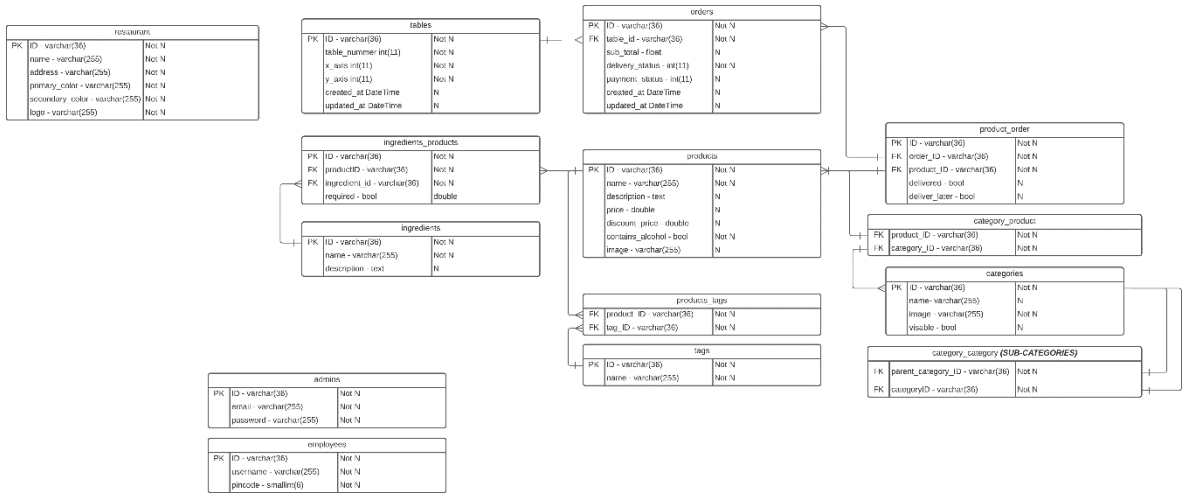
Omdat de 3 verschillende apps met elkaar verbinden via een centrale database hebben we hiervoor een library opgesteld waarin alle entiteiten en repositories staan die nodig zijn om de data in de database op te halen. Deze is op te halen via Jitpack als maven dependency. Dit ziet er dan als volgt uit in de backend, met de versie die op GitHub gereleased is:



Deze wordt dan als volgt als dependency opgehaald binnen de backend (API) van onze applicatie:

```
<dependency>
    <groupId>com.github.horeca-robot</groupId>
    <artifactId>database-library</artifactId>
    <version>1.12.0</version>
</dependency>
```

Voor deze database library hebben we ook een ERD-diagram opgesteld, deze beschrijft de structuur van de database met alle entiteiten en relaties:



Voor betere leesbaarheid kan deze afbeelding ook online gevonden worden: [Link]

Websocket

Om communicatie tussen de robot, of tussen verschillende apps, een realiteit te maken hebben we ervoor gekozen om gebruik te maken van een websocket. Deze socket is geschreven in PHP en vereist dus ook dat er een apache of nginx configuratie beschikbaar is. Verder vereist de deployment dat er gebruik gemaakt wordt van Supervisor. Meer informatie over de deployment kan gevonden worden in deze guide. Als laatste moet er vanuit de apache of nginx configuratie een reverse proxy gemaakt worden om al het verkeer naar de desbetreffende port te sturen. Ook moet de URL herschreven worden dat alles terug naar het index.php bestand geroute wordt. De nginx configuratie van onze deployment ziet er zo uit:

```
<VirtualHost *:80>
    ServerAdmin YOUR_EMAIL_HERE
    ServerName YOUR_DOMAIN_HERE
    ServerAlias YOUR_DOMAIN_HERE
    DocumentRoot YOUR_ROOT_HERE/public

    <Directory YOUR_ROOT_HERE/public>
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>

    RewriteEngine On
    RewriteCond %{REQUEST_URI}  ^/ [NC]
    RewriteCond %{HTTP:Upgrade} =websocket [NC]
    RewriteRule /(.*) ws://localhost:8080/$1 [P,L]

    ProxyPass / http://localhost:8080/
    ProxyPassReverse / http://localhost:8080/

    ErrorLog ${APACHE_LOG_DIR}/YOUR_DOMAIN_HERE_error.log
    CustomLog ${APACHE_LOG_DIR}/YOUR_DOMAIN_HERE_access.log combined
</VirtualHost>
```

Nog te doen

Nog niet alle features zijn in de applicatie verwerkt. Het grootste ding dat nog moet gebeuren is de integratie met een websocket. Deze websocket bestaat al wel en kan gevonden worden in de GitHub organization onder de naam “robot-communication”.

Daarnaast is het implementeren van de verschillende categorieën en ingrediënten van de producten nog niet af, we hebben hier geen tijd voor gehad. Dit kan worden opgelost door de producten die worden weergegeven via de bijhorende categorie op te halen.