RIGS ADMIN APP

S3-DB04 Groep 2

Inhoud

Inleiding

In dit document wordt beschreven hoe de structuur van het RIGS software systeem in elk zit en dan specifiek gericht op de admin applicatie. Ook wordt beschreven hoe je het admin project moet builden en runnen. Daarnaast wordt ook uitgelegd hoe je te werk moet gaan om het project te deployen via docker.

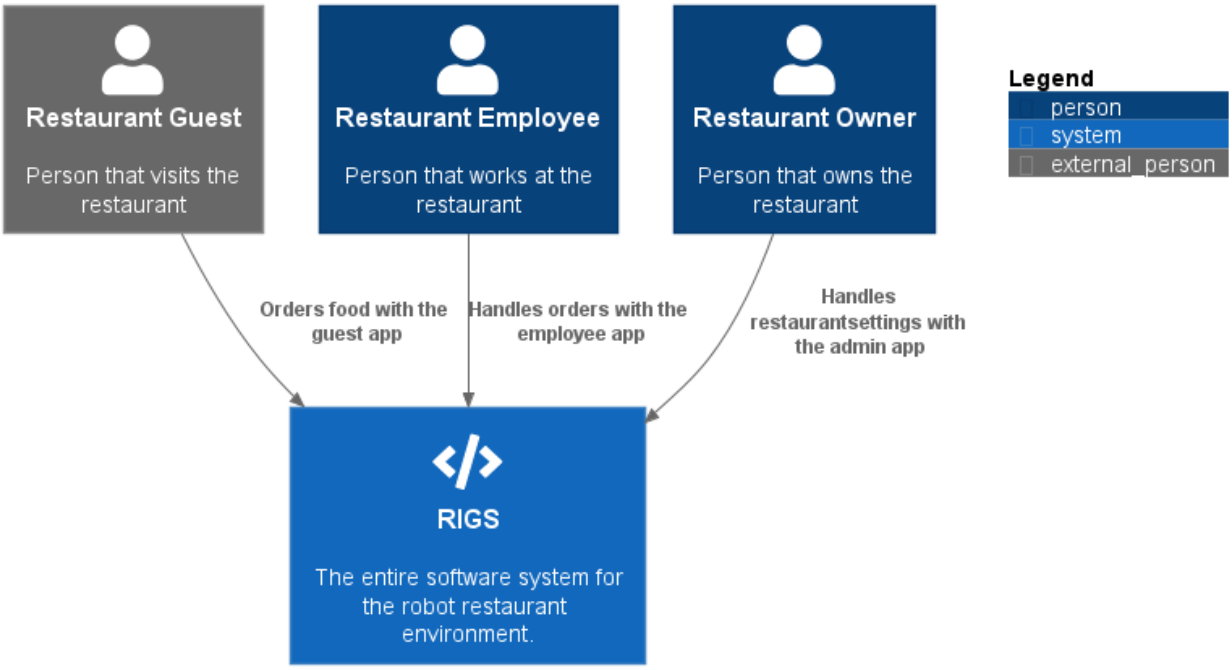
Om toegang te krijgen tot dit project, klik je hier.

Voor een demo van onze website, klik je hier.

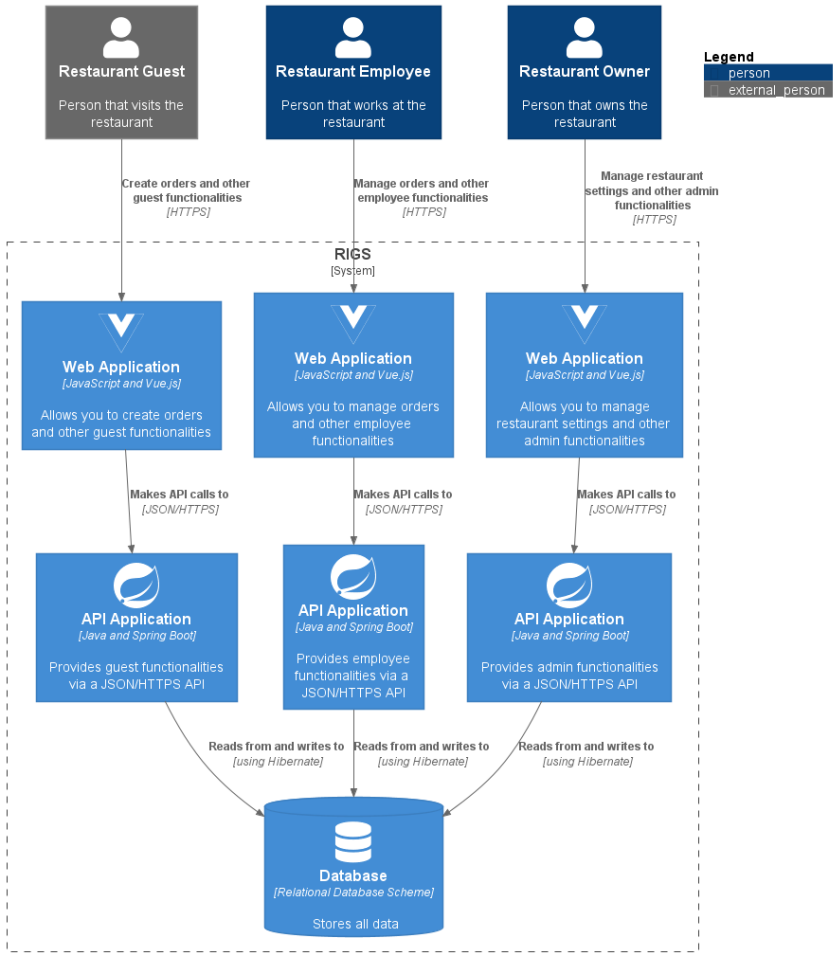
Systeem Structuur

Om de structuur van het software systeem weer te geven hebben we gebruik gemaakt van het C4 model. Dit model laat de structuur van het systeem op 4 verschillende lagen zien. Elke laag gaat steeds iets dieper, tot laag 4 dat is de code.

Context Diagram



Container Diagram



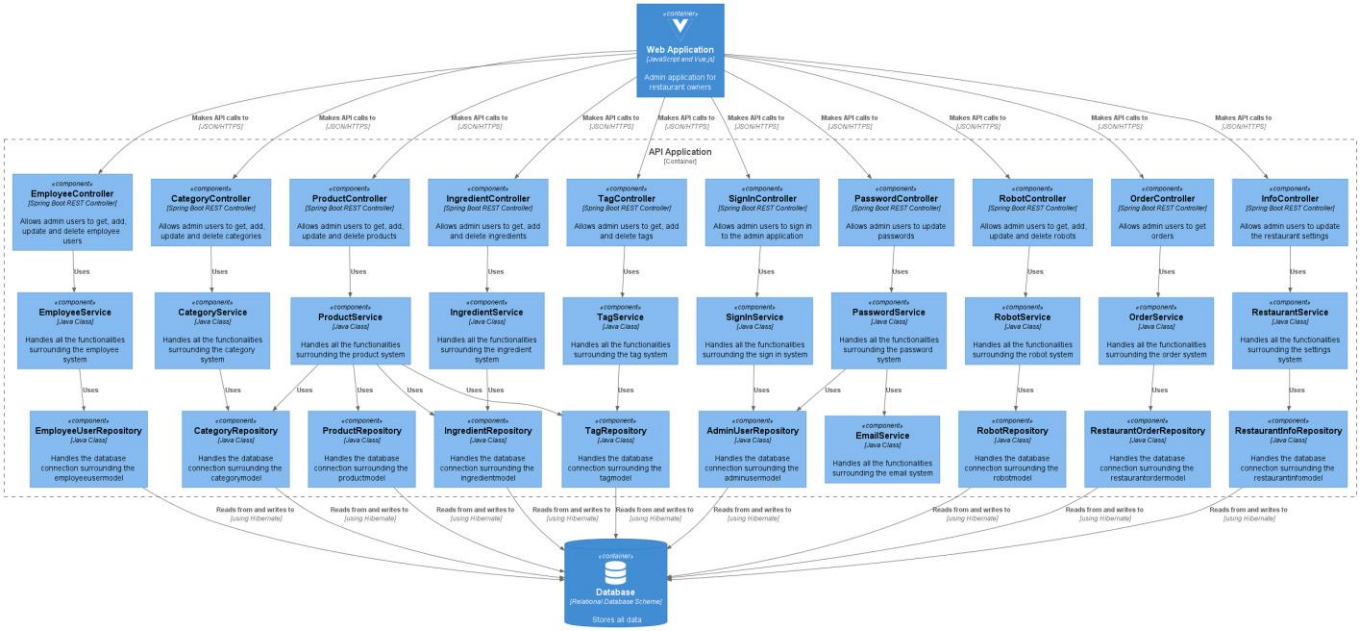
Admin applicatie: Hier stelt onder andere de restauranteigenaar het menu in.

Klant applicatie: Hier besteld de klant onder andere zijn eten

Medewerker applicatie: Hier komen onder andere de bestellingen van de klant binnen zodat de keuken kan beginnen met de bereiding.

Deze drie projecten communiceren doormiddel van een gedeelde database. Hiervoor is een database library gemaakt om conflicten te voorkomen.

Component Diagram



Dit is een diagram van de API. De API is opgedeeld in drie verschillende lagen:

1. Controller laag – Hier komen de HTTP requests binnen
2. Service laag – Hier in zit de logic van de applicatie
3. Repository laag – Hier in communiceert de applicatie met de database

Frontend Configuratie

Voor het starten van de frontend moeten eerst wat configuratie properties worden veranderd, ga hiervoor naar **.env.development** en **.env.production** in de root folder van de frontend applicatie.

VUE_APP_API_BASE_URL moet worden aangepast naar de base URL van de backend.

Setup & Run

Voor de setup en uitvoering van de applicatie zijn er twee commands nodig:

1. **npm install**
2. **npm run serve**

Backend Configuratie

Voor het starten van de backend moeten eerst wat configuratie properties worden veranderd, ga hiervoor naar **src/main/resources/application.properties**

admin.email en **admin.password** kunnen veranderd worden voor de default admin account.

spring.datasource properties moeten aangepast worden voor de connectie met de database. Bij **url** moet een connectie string worden toegevoegd naar een database en onder **username** en **password** de daarbij horende credentials.

VUE_APP_API_BASE_URL moet worden aangepast naar de base URL van de backend.

Build & Run

Voor het builden van de applicatie gebruik je dit command: **gradle build**

Dit bouwt een .jar file, deze komt te staan in de **src/build/libs/** folder. Om deze .jar te runnen gebruik dit command: **java -jar <JAR NAME>.jar**

API Documenatie

Voor de documentatie omtrent de API endpoint, klik hier.

Docker Deployment

Front-end Dockerfile

In de front-end applicatie is een Dockfile met de configuratie om de applicatie in een docker container te draaien. Belangrijk hierbij is dat in de Dockerfile staat dat de applicatie op port 8081 zal gaan luisteren op inkomende verzoeken als er in de package.json bestand een andere port wordt aangegeven dan moet diezelfde poort ook in de Dockerfile komen te staan anders zal de applicatie niet bereikbaar zijn.

Voor de rest zorgt de Dockerfile er voor dat de applicatie gebouwd wordt voor productie.

Om de een docker image te maken voor het front-end moet je eerst Docker installeren, dat kan hier.

Als docker geïnstalleerd is voer je de volgende commando uit in een terminal/CMD in de root folder van het project waar de Dockerfile zich bevindt.

Commando: docker build -t rigs-front-end .

De naam rigs-front-end kan veranderd woorden naar iets anders

LET OP DE PUNT OP HET EINDE IS HEEL BELANGRIJK ANDERS WORD DE IMAGE NIET GEMAAKT!

Backend Dockerfile

In de backend applicatie is een Dockfile met de configuratie om de applicatie in een docker container te draaien. Belangrijk hierbij is dat in de Dockerfile staat dat de applicatie op port 8080 zal gaan luisteren op inkomende verzoeken als een andere port wordt aangegeven dan moet diezelfde poort ook in de Dockerfile komen te staan anders zal de applicatie niet bereikbaar zijn.

Voor de rest zorgt de Dockerfile er voor dat de applicatie gebouwd wordt voor productie.

Om de een docker image te maken voor het front-end moet je eerst Docker installeren, dat kan hier.

Als docker geïnstalleerd is voer je de volgende commando uit in een terminal/CMD in de root folder van het project waar de Dockerfile zich bevindt.

Commando: docker build -t rigs-backend .

De naam rigs-backend kan veranderd woorden naar iets anders

LET OP DE PUNT OP HET EINDE IS HEEL BELANGRIJK ANDERS WORD DE IMAGE NIET GEMAAKT!

DockerCompose

Om beide applicaties live te zetten en om er voor te zorgen dat ze met elkaar kunnen communiceren is er gebruik gemaakt van een docker-compose.yml bestand in de root van het project. Niet in de backend of front-end project maar waar beide projecten zitten. De docker-compose.yml zorgt ervoor dat beide applicaties worden gebouwd door op beide docker build -t [naam] . uit te voeren.

Download docker-compose

In de docker-compose.yml wordt er gebruik gemaakt van een reverse proxy zodat beide applicaties hun eigen host hebben maar toch met elkaar kunnen communiceren. Om de docker-compose.yml uit te voeren moet je de volgende commando uitvoeren in een terminal/CMD. Commando: docker-compose up --build beide applicaties worden gebouwd en gestart.

In de docker-compose die wij hebben opgezet wordt en nog gebruik gemaakt van een lokale mariadb database container. Als de applicatie gebruikt gaat worden in een lokale omgeving, zal de database connectie string in de API properties aangepast moeten worden naar die container.

Niet Geïmplementeerde Requirements

1. Generen van map naar .yaml file.
2. Opslaan van de map in de database en naar de robot sturen.
3. Categorieën volgorde aan kunnen passen.

Belangrijke Info

Om afbeeldingen op te slaan in de database maken wij gebruik van base64. Een base64 image kan een erg lange string zijn en daarom moet er iets aangepast worden in de database instellingen om zo'n lange string toe te staan. In de configuratie van de MySQL server verander **max_allowed_packet** naar **10M** om langere strings toe te laten.