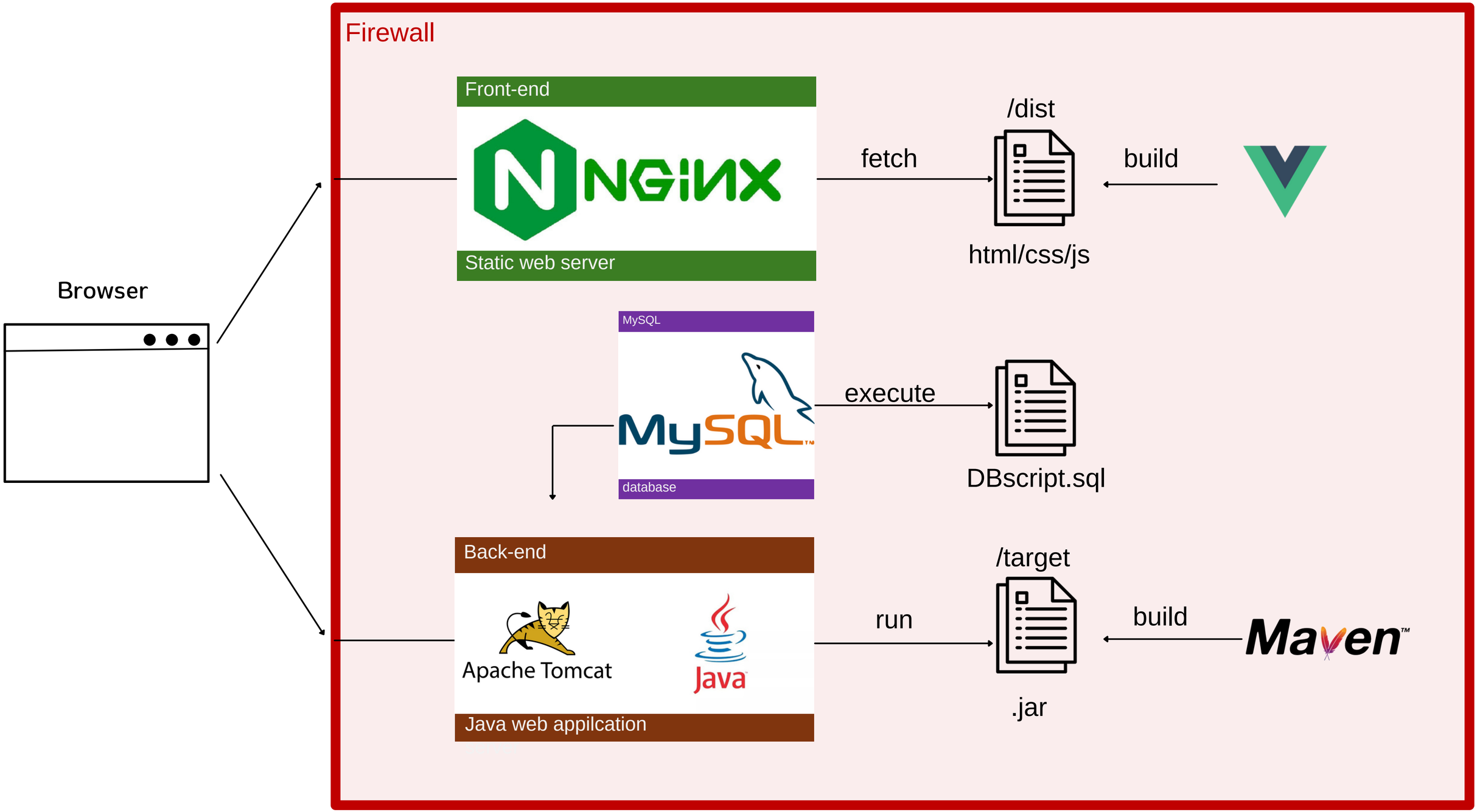
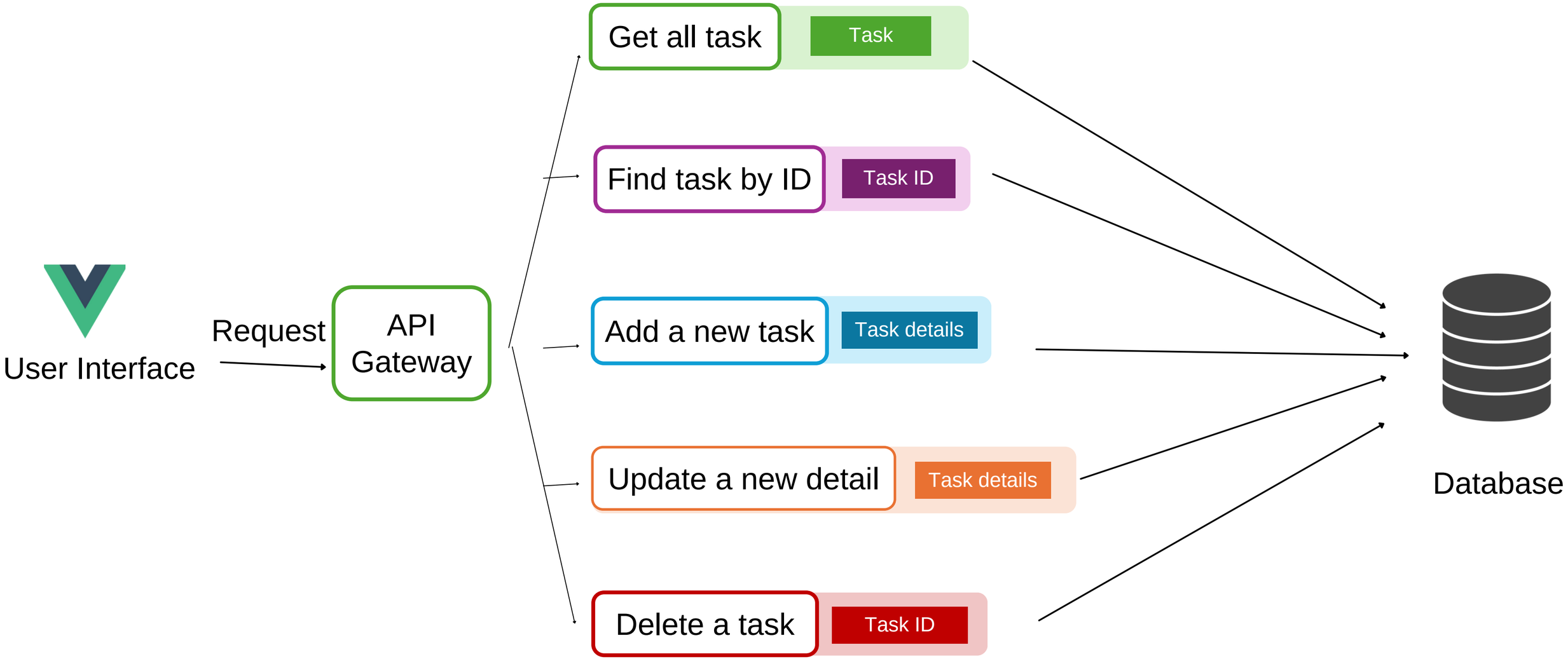

Firewall
Front-end
NGINX
Static web server
/dist
fetch
build
html/css/js
Browser
MySQL
MySQL
database
execute
DBscript.sql
Back-end
Apache Tomcat
Java
Java web appilcation
/target
run
build
Maven
.jar

API specifications

Method	Path	Description
GET	/task	To get all tasks
GET	/task/{taskId}	Find task by ID of task
POST	/task	To add a new task
PATCH	/task/{taskId}	To update a new detail in a task
DELETE	/task/{taskId}	To delete a task

Software architecture : Service-Based



Client

Server

1. Request (Json)

Example message

```
POST /api/v1/users HTTP/1.1
Host: example.com
Content-Type: application/json
Authorization:Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3O
{
    "name": "Alice",
    "email": "alice@example.com",
    "password": "secret123"
}
```

2.Response(Json)

Example message

```
HTTP/1.1 201 Created
Content-Type: application/json

{
"id": "1234567890",
"name": "Alice",
"email": "alice@example.com",
"createdAt": "2023-05-25T10:15:30Z"
}
```

- **GET**: Retrieves a representation of a resource.
- **POST**: Submits data to be processed by the server, often used for form submissions.
- **PUT**: Updates or replaces a resource with the provided data.
- **DELETE**: Removes a specified resource.

Protocol: TCP

Frontend

Stores(pinia)

task.js

status.js

toast.js

theme.js

limitTask.js

filter.js

Composables

task-api.js

status-api.js

limit-api.js

Composables
provides several benefits

- Reusability
- Separation of Concerns
- Maintainability
- Testability
- Consistency
- Enhanced Code Readability

*เปลี่ยนจาก libs มาใช้ composables เนื่องจากต้องการเรียกใช้ toast เมื่อมีการ call api เกิดขึ้น -> เปลี่ยนรูปแบบการ export เป็น closure function

App.vue

Fetch data when mounted(tasks, statuses, limitTask)

{path: '/task'}

{path: '/status'}

TaskView.vue

TaskTable.vue

ConfirmDelete.vue (['mode', 'object', 'number'])

(['closeModal'])

StatusSetting.vue

(['close'])

FilterModal.vue

(['close'] , ['applyFilter'])

{path: '/task'}

{path: '/:taskId'}

{path: '/:taskId/edit'}

{path: '/add'}

TaskModal.vue

StatusView.vue

StatusTable.vue

ConfirmDelete.vue (['mode', 'object', 'number'])

(['closeModal'])

StatusSetting.vue

(['close'])

{path: '/status}

{path: '/:statusId'}

{path: '/:statusId/edit'}

{path: '/add'}

StatusModal.vue

Backend

Entity

LimitTask

- id (Integer)
- limit (Boolean)
- limitMaximumTask (Integer)

TaskV2

- id (Integer)
- title(String)
- description (Integer)
- assignees (String)
- createdOn (Date)
- updatedOn (Date)
- Status (Status)

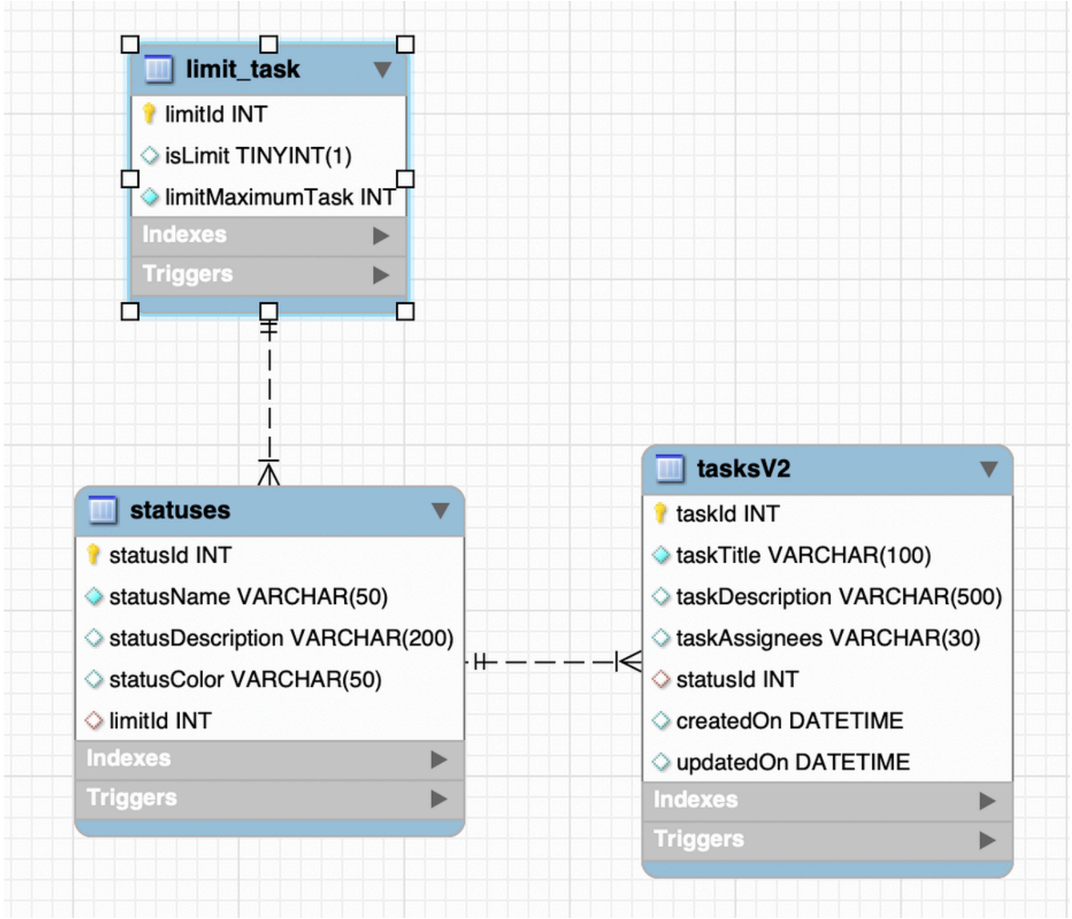
Status

- id (Integer)
- name (String)
- description (String)
- status (color)
- noOfTasks (List)

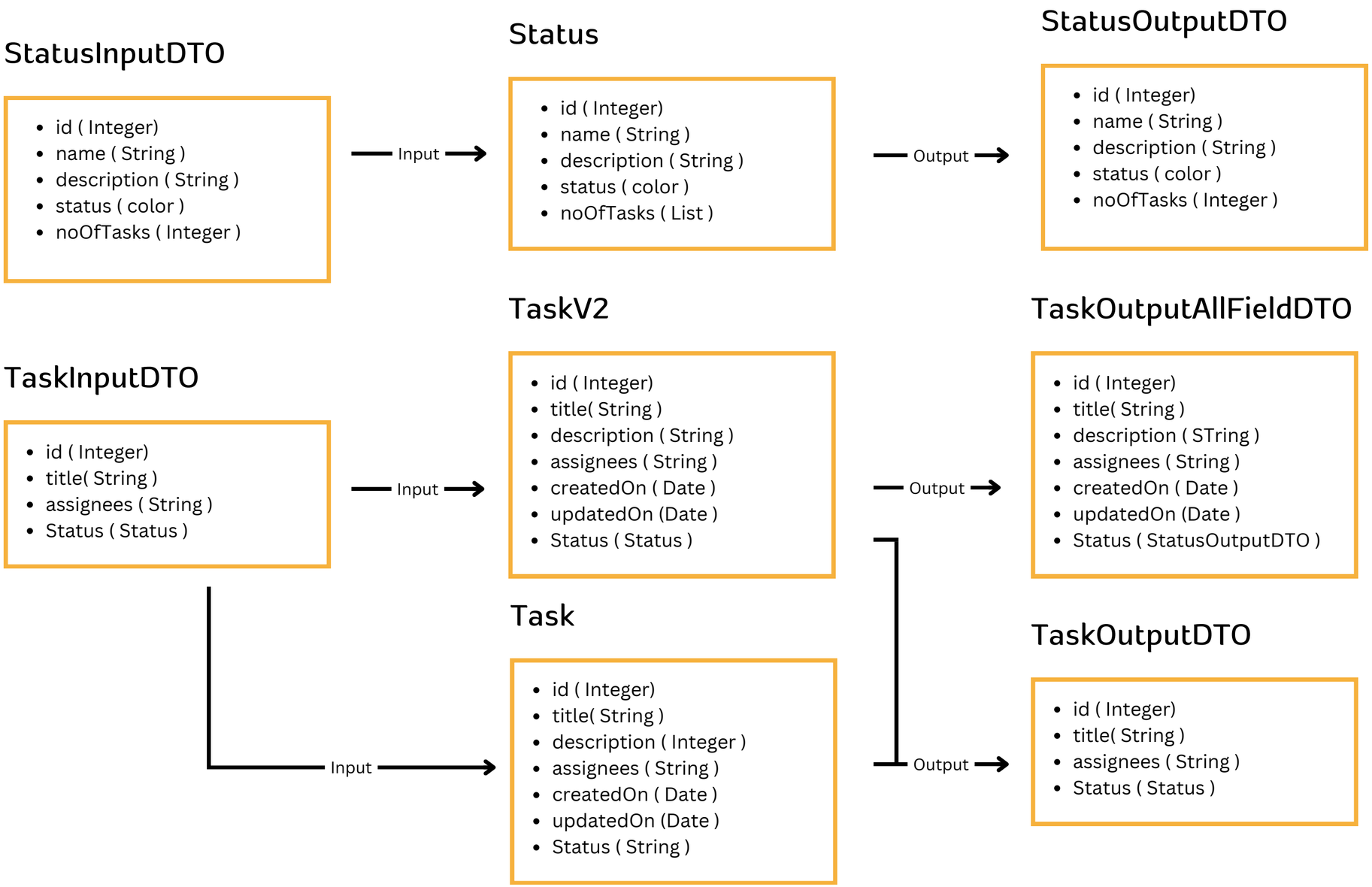
Task

- id (Integer)
- title(String)
- description (Integer)
- assignees (String)
- createdOn (Date)
- updatedOn (Date)
- Status (String)

Relationnal

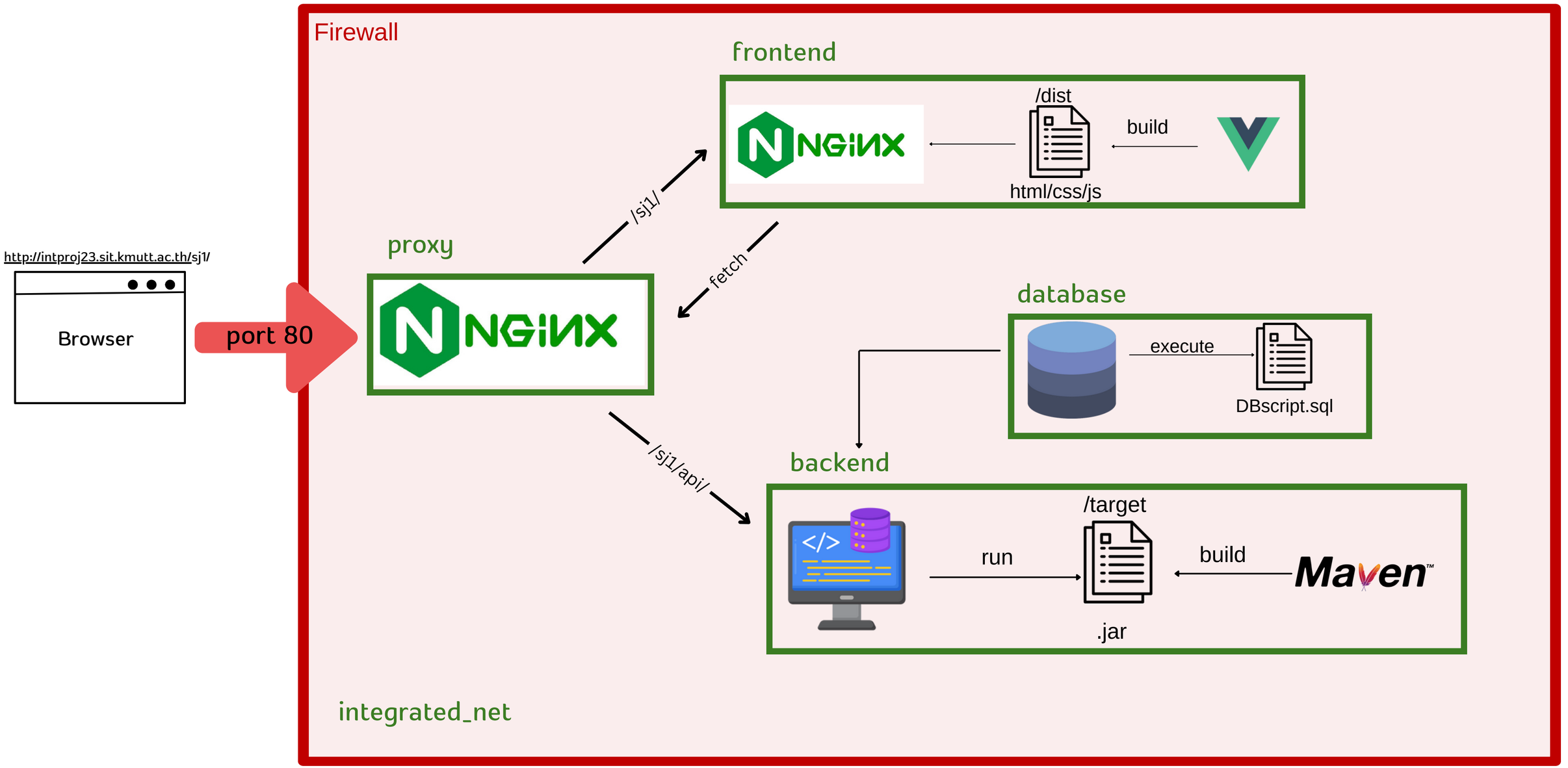


DTO


StatusInputDTO
id (Integer)
name (String)
description (String)
status (color)
noOfTasks (Integer)
Input
Status
id (Integer)
name (String)
description (String)
status (color)
noOfTasks (List)
Output
StatusOutputDTO
id (Integer)
name (String)
description (String)
status (color)
noOfTasks (Integer)
TaskInputDTO
id (Integer)
title(String)
assignees (String)
Status (Status)
Input
TaskV2
id (Integer)
title(String)
description (String)
assignees (String)
createdOn (Date)
updatedOn (Date)
Status (Status)
Output
TaskOutputAllFieldDTO
id (Integer)
title(String)
description (STring)
assignees (String)
createdOn (Date)
updatedOn (Date)
Status (StatusOutputDTO)
Input
Task
id (Integer)
title(String)
description (Integer)
assignees (String)
createdOn (Date)
updatedOn (Date)
Status (String)
Output
TaskOutputDTO
id (Integer)
title(String)
assignees (String)
Status (Status)

Integration

Infrastructure



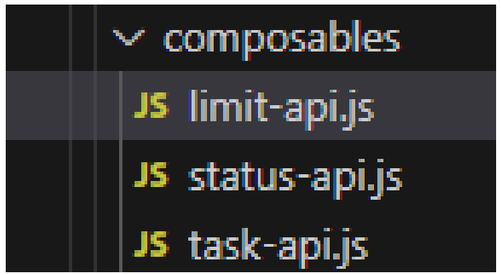
Frontend

.env.production

integrated-1-frontend > $ .env.production

```
# VITE_BASE_URL=http://ip23sj1.sit.kmutt.ac.th:8080/v2
VITE_BASE_URL=/sj1/api/v2
VITE_BASE=/sj1
```

Fetch API



Composables
provides several benefits

- Reusability
- Separation of Concerns
- Maintainability
- Testability
- Consistency
- Enhanced Code Readability

Dockerfile (ของ frontend)

```
FROM node:20.10.0 as build
COPY . .
RUN rm -rf node_modules/
RUN npm install
RUN npm run build

FROM nginx:alpine as run
COPY --from=build dist /usr/share/nginx/html
COPY nginx.conf /etc/nginx/nginx.conf

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

Backend

Dockerfile (ของ backend)

```
FROM maven:3.8.3-openjdk-17 AS build
COPY pom.xml .
COPY src ./src
RUN mvn clean package -DskipTests

FROM openjdk:17-jdk-alpine AS run
COPY --from=build target/*.jar app.jar
EXPOSE 8080
CMD ["java", "-jar", "app.jar"]
```

Database

Dockerfile (Database)

```
FROM mysql:latest
ENV MYSQL_USER=dev
ENV MYSQL_PASSWORD=Mysql@sit123
ENV MYSQL_ROOT_PASSWORD=Mysql@sit123
ENV MYSQL_DATABASE=integrated
ENV MYSQL_CHARSET=utf8mb4
ENV MYSQL_COLLATION=utf8mb4_unicode_ci
EXPOSE 3306
COPY scripts/ /docker-entrypoint-initdb.d/
COPY dbscripts/ /dbscripts/

COPY my.cnf /etc/my.cnf
```

Proxy

Dockerfile (Proxy)

```
FROM nginx:alpine as run
COPY default.conf /etc/nginx/conf.d/default.conf

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

defualt.conf

```
server {
    listen       80 default_server;
    server_name  ip23sj1.sit.kmutt.ac.th;

    location / {
        proxy_pass http://frontend/;
    }
    location /api/ {
        proxy_pass http://backend:8080/;
    }
    location /sj1/ {
        proxy_pass http://frontend/;
    }
    location /sj1/api/ {
        proxy_pass http://backend:8080/;
    }
}
```

Compose.yaml

service (database)

```yaml
database:
  build: ./integrated-1-database
  container_name: database
  restart: unless-stopped
  networks:
    - integrated_net
  volumes:
    - myapp:/var/lib/mysql
  healthcheck:
    test: ["CMD", "mysqladmin", "ping", "-h", "localhost"]
    interval: 5s
    timeout: 5s
    retries: 5
```

service (frontend)

```yaml
frontend:
  build: ./integrated-1-frontend
  container_name: frontend
  restart: unless-stopped
  networks:
    - integrated_net
  depends_on:
    - backend
```

service (backend)

```yaml
backend:
  build: ./integrated-1-backend
  container_name: backend
  restart: unless-stopped
  networks:
    - integrated_net
  depends_on:
    - database
  environment:
    - DB_HOST=database
    - DB_PORT=3306
    - DB_NAME=integrated
    - DB_USER=dev
    - DB_PASS=Mysql@sit123
```

networks & volumes

```yaml
networks:
  integrated_net:

volumes:
  myapp:
```

service (proxy)

```yaml
proxy:
  build: ./integrated-1-proxy
  container_name: proxy
  restart: unless-stopped
  ports:
    - 80:80
  networks:
    - integrated_net
  depends_on:
    - frontend
    - backend
    - database
```