CDOF1-Group-1

Containerization technologies

Paul Lacoutiere, Adrien Balderacchi and Shree Varshan Murali

Table of CONTENTS

INTRODUCTION

Microservices-based Task Management System

OUR TEAM

Paul Lacoutiere

Backend User and dockerfiles

Adrien Balderacchi

Frontend

Shree varshan Murali

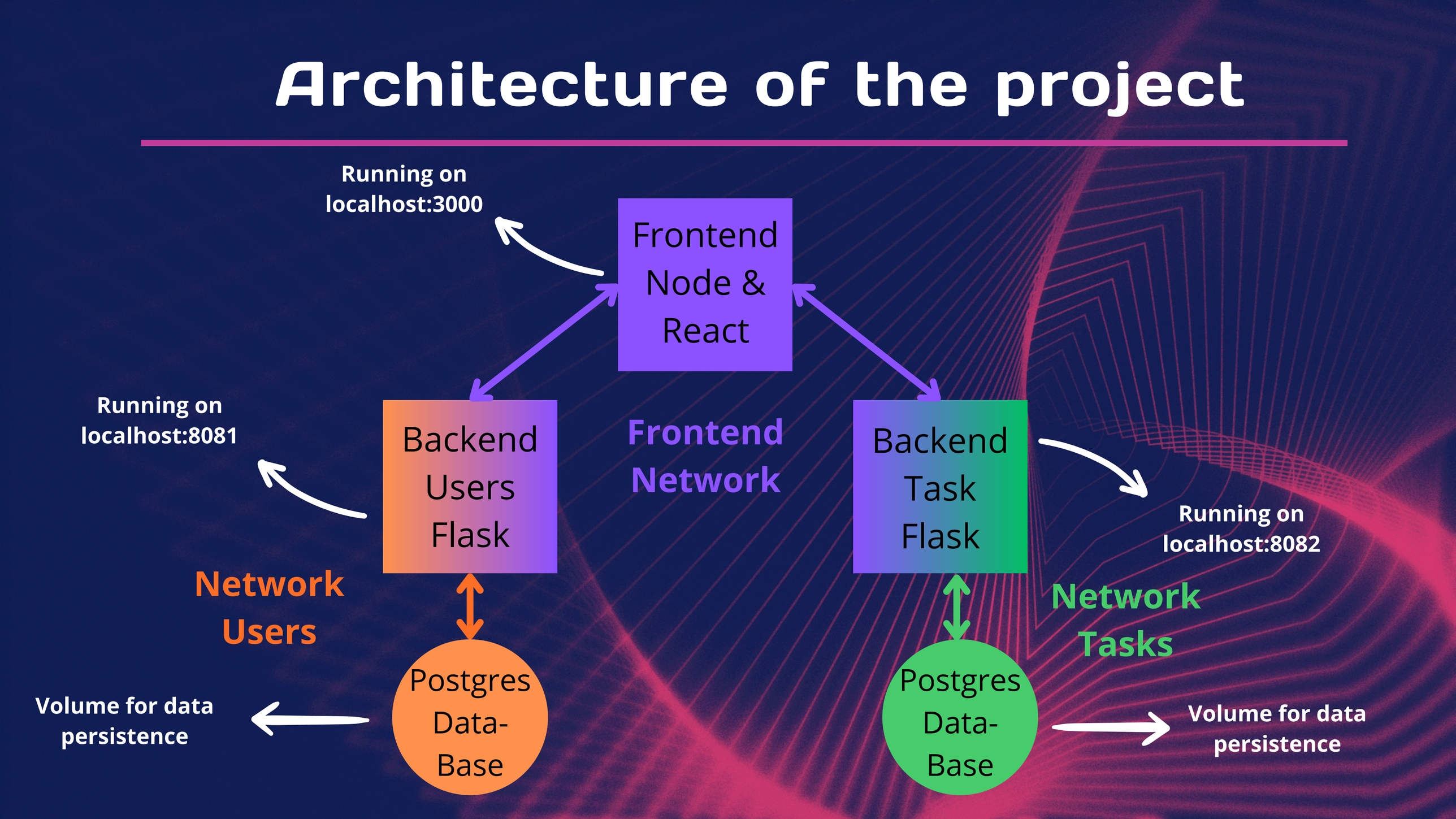
Backend Task

Coopération and communication

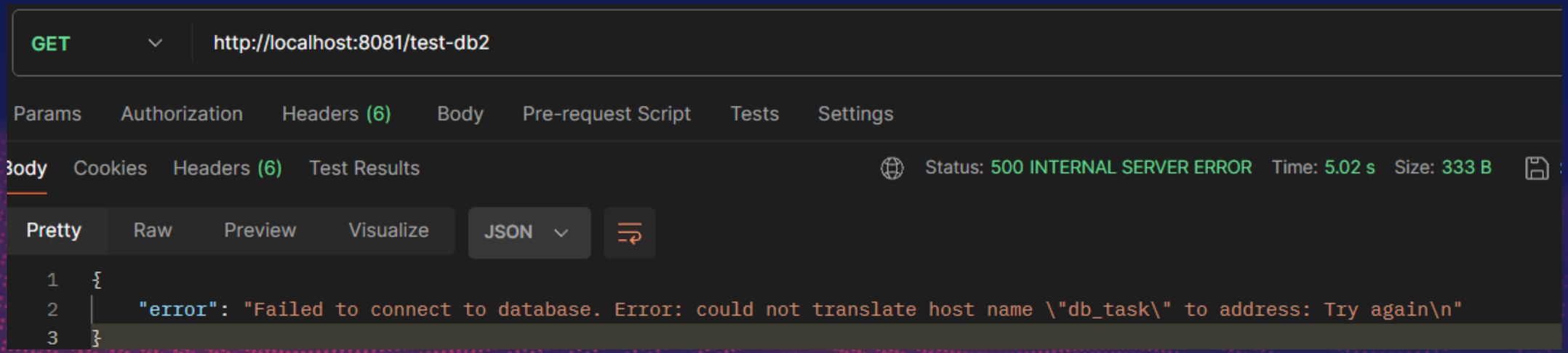
Communcation via a Whatsapp group

Coopération with GitLab, branches, merge request

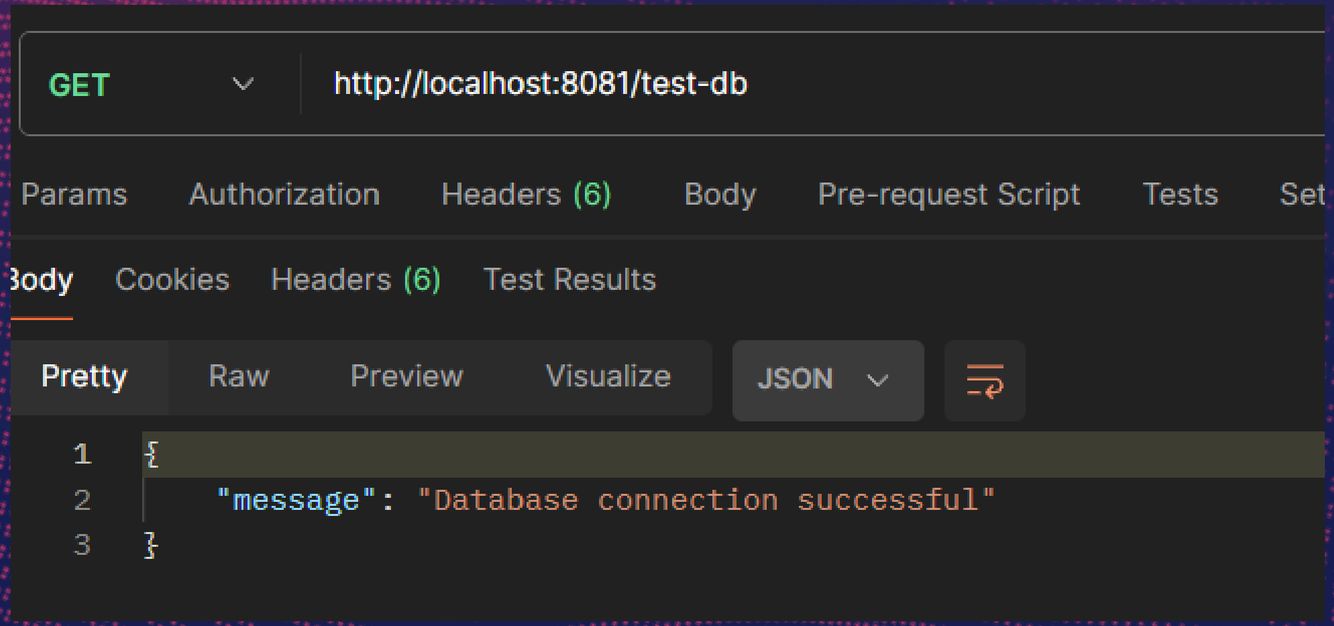
Architecture of the project



Network Isolation



The first backend does not have access to the second database



The first backend have access to his database

Our Dockerfiles
Backend user

```dockerfile
# BackendUsersApp.dockerfile
FROM python:3.9-alpine
#Créate a new user
RUN adduser -D userBackendTasks
#Set the working directory
WORKDIR /app
#Copy the current directory contents into the container at /app
COPY . /app
#Install the dependencies
RUN pip install --no-cache-dir -r requirements.txt
#Expose port 8080
EXPOSE 8080
#Use the user we previously created
USER userBackendTasks
#Set the health check
HEALTHCHECK --interval=5s --timeout=3s \
CMD curl -f http://localhost:8082/ || exit 1
#Run the application
CMD ["python", "app.py"]
```

Python flask backend with a Postgres database

```dockerfile
# Use the official PostgreSQL image
FROM postgres:14-alpine
#Switch to the root user
USER root
#Set the environment variable
ENV PGDATA /data
#Set the working directory
WORKDIR /docker-entrypoint-initdb.d
# Copy the SQL initialization script into the container
COPY init.sql /docker-entrypoint-initdb.d/
```

Our Dockerfiles
Backend tasks

```
FROM postgres:14-alpine
#Switch to the root user
USER root
#Set the environment variable
ENV PGDATA /data
#Set the working directory
WORKDIR /docker-entrypoint-initdb.d
# Copy the SQL initialization script into the contai
COPY script.sql /docker-entrypoint-initdb.d/
```

Python flask backend with a Postgres database

```
# Dockerfile.app
FROM python:3.9-alpine
#Créate a new user
RUN adduser -D userBackendUsers
#Set the working directory
WORKDIR /app
#Copy the current directory contents into the container at /app
COPY . /app
#Install the dependencies
RUN pip install --no-cache-dir -r requirements.txt
#Expose port 8080
EXPOSE 8080
#Use the user we previously created
USER userBackendUsers
#Set the health check
HEALTHCHECK --interval=5s --timeout=3s \
CMD curl -f http://localhost:8081/ || exit 1
#Run the application
CMD ["python", "app.py"]
```

Our Dockerfiles
Frontend

```
# Use official Node.js image as base
FROM node:14-alpine
#Set a user
USER node
# Set working directory
WORKDIR /app
# Copy package.json and package-lock.json to work directory
COPY package*.json ./
# Install dependencies
RUN npm install --silent
# Copy the rest of the application code
COPY . ./
# Expose port 3000
EXPOSE 3000
# start app
CMD ["npm", "start"]
```

Docker compose

```yaml
services:
  db_users:
    build:
      context: ./BackendUsers/Database
      dockerfile: BackendUsersDb.dockerfile
    env_file:
      - BackendUsers/Database/.env
    volumes:
      - db-vol-users:/data
    networks:
      - network-db-users
    container_name: db_users

  app_users:
    build:
      context: ./BackendUsers/Backend
      dockerfile: BackendUsersApp.dockerfile
    ports:
      - "8081:8080"
    depends_on:
      - db_users
    networks:
      - network-db-users
      - network-frontend
    container_name: app_users
```

```yaml
db_task:
  build:
    context: ./BackendTasks/Database
    dockerfile: BackendTasksDb.dockerfile
  env_file:
      - BackendTasks/Database/.env
  volumes:
    - db-vol-task:/data
  networks:
    - network-db-task
  container_name: db_task

app_task:
  build:
    context: ./BackendTasks/Backend
    dockerfile: BackendTasksApp.dockerfile
  ports:
    - "8082:8080"
  depends_on:
    - db_task
  networks:
    - network-db-task
    - network-frontend
  container_name: app_task
```

```yaml
  app_frontend:
    build:
      context: ./front-react
      dockerfile: Dockerfile
    ports:
      - "3000:3000"
    networks:
      - network-frontend
    container_name: app_frontend
    depends_on:
      - app_users
      - app_task

networks:
  network-db-users:
  network-db-task:
  network-frontend:

volumes:
  db-vol-users:
  db-vol-task:
```

Our Pipeline

```yaml
stages:
  - kics
  - build&deploy

kics-scan:
  stage: kics
  image:
    name: checkmarx/kics:latest
    entrypoint: [""]
  script:
    - kics scan --no-progress -p ${PWD}/BackendTasks/Database/BackendTasksDb.dockerfile -o ${PWD}/kics-results --fail-on "high" --report-formats json --output-name tasks-db
    - kics scan --no-progress -p ${PWD}/BackendTasks/Backend/BackendTasksApp.dockerfile -o ${PWD}/kics-results --fail-on "high" --report-formats json --output-name tasks-app
    - kics scan --no-progress -p ${PWD}/BackendUsers/Database/BackendUsersDb.dockerfile -o ${PWD}/kics-results --fail-on "high" --report-formats json --output-name users-db
    - kics scan --no-progress -p ${PWD}/BackendUsers/Backend/BackendUsersApp.dockerfile -o ${PWD}/kics-results --fail-on "high" --report-formats json --output-name users-app
    - kics scan --no-progress -p ${PWD}/front-react/Dockerfile -o ${PWD}/kics-results --fail-on "high" --report-formats json --output-name frontend
  artifacts:
    name: kics-results
    paths:
      - kics-results
```

Two stages:
Kics
Build and deploy

```yaml
build:
  stage: build&deploy
  image:
    name: gcr.io/kaniko-project/executor:v1.14.0-debug
    entrypoint:
    - ''
  script:
  - /kaniko/executor --context "${CI_PROJECT_DIR}/BackendUsers/Database" --dockerfile
    "${CI_PROJECT_DIR}/BackendUsers/Database/BackendUsersDb.dockerfile" --destination
    "${CI_REGISTRY_USER}/backend_users_db:${CI_COMMIT_TAG}"
  - /kaniko/executor --context "${CI_PROJECT_DIR}/BackendUsers/Backend" --dockerfile
    "${CI_PROJECT_DIR}/BackendUsers/Backend/BackendUsersApp.dockerfile" --destination
    "${CI_REGISTRY_USER}/backend_users_app:${CI_COMMIT_TAG}"
  - /kaniko/executor --context "${CI_PROJECT_DIR}/BackendTasks/Database" --dockerfile
    "${CI_PROJECT_DIR}/BackendTasks/Database/BackendTasksDb.dockerfile" --destination
    "${CI_REGISTRY_USER}/backend_tasks_db:${CI_COMMIT_TAG}"
  - /kaniko/executor --context "${CI_PROJECT_DIR}/BackendTasks/Backend" --dockerfile
    "${CI_PROJECT_DIR}/BackendTasks/Backend/BackendTasksApp.dockerfile" --destination
    "${CI_REGISTRY_USER}/backend_tasks_app:${CI_COMMIT_TAG}"
  - /kaniko/executor --context "${CI_PROJECT_DIR}/front-react" --dockerfile "${CI_PROJECT_DIR}/front-react/Dockerfile"
    --destination "${CI_REGISTRY_USER}/frontend:${CI_COMMIT_TAG}"
  - echo "{\"auths\":{\"${CI_REGISTRY}\":{\"auth\":\"$(printf "%s:%s""${CI_REGISTRY_USER}"
    "${CI_REGISTRY_PASSWORD}" | base64 | tr -d'\n')\"}}}" > /kaniko/.docker/config.json
  rules:
  - if: "$CI_COMMIT_TAG"
```

TECHNICAL AND SOFT SKILLS CHALLENGES

Front-end

Communication & Coordination

Docker

DÉMONSTRATION

CONCLUSION

Thank you for your attention