Manual Técnico

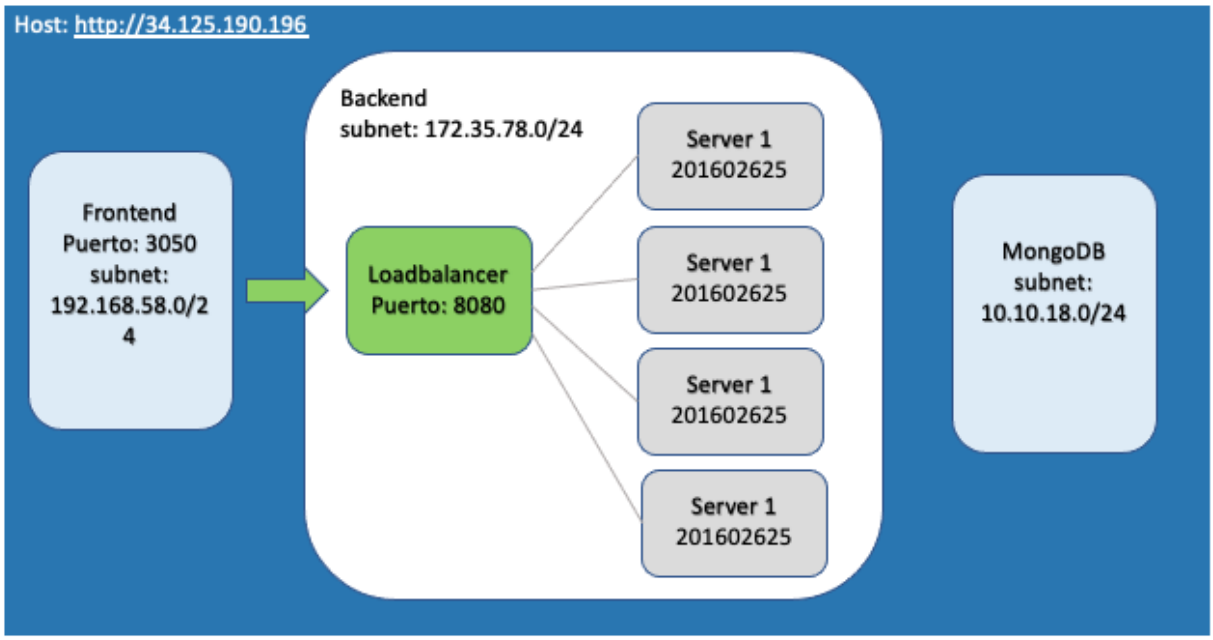
Práctica 2

LABORATORIO DE REDES DE COMPUTADORAS 2

GRUPO 18

Marlon Abraham Fuentes Zárate	199911132
Oscar Alfredo Llamas Lemus	201602625
Nery Antonio Alvizures	201602986
Nery Eduardo Herrera Cottom	2012602870

Arquitectura



Frontend

Se utilizó el framework react, y se realizó conteinerización mediante docker.

Archivo Dockerfile:

REDES2_2S2021_GRUPO18 > Practica2 > frontend > Dockerfile >

```
1  FROM node:alpine
2  WORKDIR /app
3  COPY package.json ./
4  COPY package-lock.json ./
5  COPY ./ ./
6  RUN npm i
7  CMD ["npm", "run", "start"]
```

También se configuró un servidor nginx para la publicación:

REDES2_2S2021_GRUPO18 > Practica2 > nginx-frontend > Dockerfile >

```
FROM nginx
COPY ./default.conf /etc/nginx/conf.d/default.conf
```

Archivo .conf

REDES2_2S2021_GRUPO18 > Practica2 > nginx-frontend > default.conf

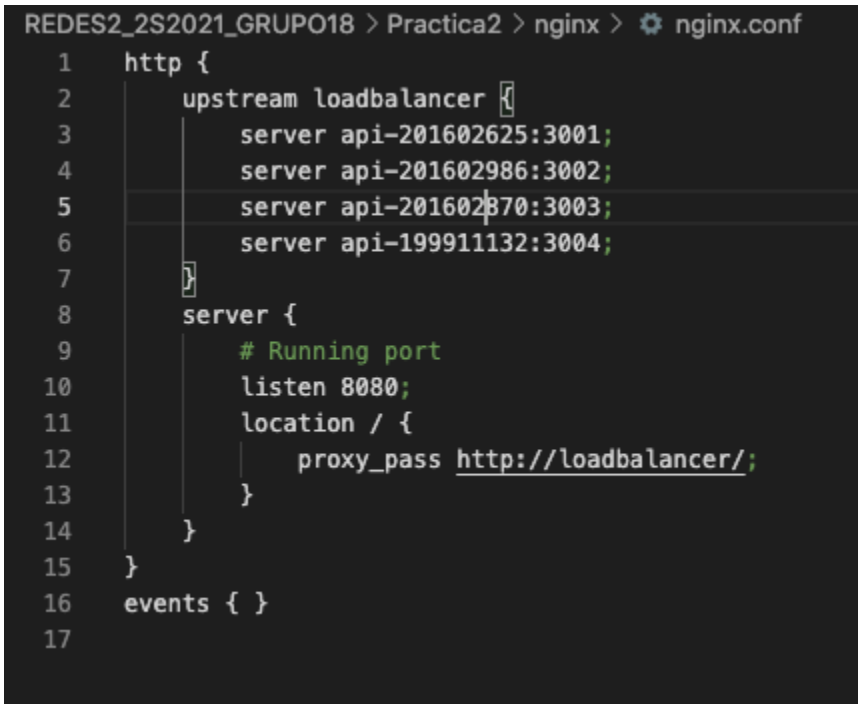
```
upstream client {
    server client:3000;
}

upstream loadbalancer {
    server loadbalancer:8080;
}

server {
    listen 80;

    location / {
        proxy_pass http://client;
    }

    location /sockjs-node {
        proxy_pass http://client;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";
    }

    location /loadbalancer {
        rewrite /loadbalancer/(.*) /$1 break;
        proxy_pass http://loadbalancer;
    }
}
```

Balanceador de Carga

Se configuró un servidor nginx el cual redirige el tráfico a los 4 servidores en la misma proporción. Para la containerización se utiliza docker.

Archivo Dockerfile:

Archivo .conf



Servidores

Para la realización de los servidores se utilizó node js - express. Para la containerización se utiliza docker.

REDES2_2S2021_GRUPO18 > Practica2 > API > Dockerfile

```
FROM node:12

ARG API_PORT

ENV API_PORT $API_PORT

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE ${API_PORT}

CMD ["npm", "run", "start"]

```

Las rutas creadas y disponibles son:

```
router.post('/create/reporte', controller.createReporte);
router.get('/get/reporte', controller.getAllReportes);
router.get('/get/reporte/:carnet', controller.getReporte);
router.get('/get/reporte/id/:id', controller.getReporteID);
```

Docker-Compose

Para la configuración de la red de cada uno de los contenedores y para ejecución de las imágenes se utilizó una archivo docker-compose.yml que tiene las siguiente configuración:

Se crearon las redes

- front-end_network
- service-network
- db-network

todas configuradas en modo bridge

```
versión: "3.5"

services:

 client:
   stdin_open: true
   environment:
     - CHOKIDAR_USEPOLLING=true
   build:
     dockerfile: Dockerfile
     context: ./frontend
   #volumes:
   #  - /app/node_modules
   #  - ./frontend:/app
   networks:
     - service_network
     - frontend_network

 nginx:
   depends_on:
     - loadbalancer
     - client
   restart: always
   build:
     dockerfile: Dockerfile
     context: ./nginx-frontend
   ports:
     - "3050:80"
   networks:
     - service_network
     - frontend_network
```

```
loadbalancer:
  build: ./nginx
  ports:
    - "8080"
  restart: always
  depends_on:
    - api-201602625
    - api-201602986
    - api-201602870
    - api-199911132
  networks:
    - service_network
    - frontend_network

mongoservice:
  image : mongo:latest
  container_name: mongoservice
  volumes:
    - /home/data/db:/data/db
  ports:
    - "27017"
  restart: unless-stopped
  networks:
    - db_network
    - service_network

api-201602625:
  container_name: api-201602625
  build:
    context: ./API
    dockerfile: Dockerfile
    args:
      - API_PORT=3001
  ports:
   - "3001"
  links:
    - mongoservice
  depends_on:
    - mongoservice
  restart: unless-stopped
  environment:
    - FIRMA=201602625
    - SERVER_HOSTNAME=0.0.0.0
    - SERVER_PORT=3001
    - MONGO_HOST=mongoservice
    - MONGO_PORT=27017
    - db_retry_time=2000
  networks:
    - service_network
    - db_network

api-201602986:
  container_name: api-201602986
  build:
    context: ./API
    dockerfile: Dockerfile
    args:
      - API_PORT=3002
  ports:
   - "3002"
```

```
  links:
    - mongoservice
  depends_on:
    - mongoservice
  restart: unless-stopped
  environment:
    - FIRMA=201602986
    - SERVER_HOSTNAME=0.0.0.0
    - SERVER_PORT=3002
    - MONGO_HOST=mongoservice
    - MONGO_PORT=27017
    - db_retry_time=2000
  networks:
    - db_network
    - service_network

api-201602870:
  container_name: api-201602870
  build:
    context: ./API
    dockerfile: Dockerfile
    args:
      - API_PORT=3003
  ports:
   - "3003"
  links:
    - mongoservice
  depends_on:
    - mongoservice
  restart: unless-stopped
  environment:
    - FIRMA=201602870
    - SERVER_HOSTNAME=0.0.0.0
    - SERVER_PORT=3003
    - MONGO_HOST=mongoservice
    - MONGO_PORT=27017
    - db_retry_time=2000
  networks:
    - db_network
    - service_network

api-199911132:
  container_name: api-199911132
  build:
    context: ./API
    dockerfile: Dockerfile
    args:
      - API_PORT=3004
  ports:
   - "3004"
  links:
    - mongoservice
  depends_on:
    - mongoservice
  restart: unless-stopped
  environment:
    - FIRMA=199911132
    - SERVER_HOSTNAME=0.0.0.0
    - SERVER_PORT=3004
    - MONGO_HOST=mongoservice
```

```
      - MONGO_PORT=27017
      - db_retry_time=2000
    networks:
      - db_network
      - service_network

networks:
  db_network:
    driver: bridge
    ipam:
      driver: default
      config:
       - subnet: 10.10.18.0/24

 service_network:
    driver: bridge
    ipam:
      driver: default
      config:
       - subnet: 172.35.78.0/24

 frontend_network:
    driver: bridge
    ipam:
      driver: default
      config:
       - subnet: 192.168.58.0/24
```