Exercise One:

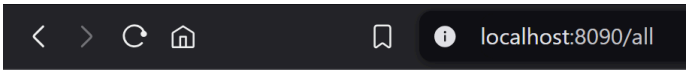
Building the image of the web app:

```
$ docker build -t web-app .
```

Running the container on port 8090:

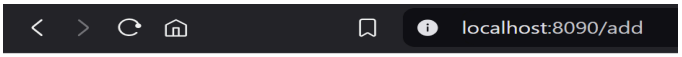
```
$ docker run -d -p 8090:5000 --name web-app web-app
```

Results:



Student Information Form

Student ID	First Name	Last Name	Module Code



Student Information Form

Student ID:

First Name:

Last Name:

Module Code:

Submit

Exercise Two:

Create a network to connect the database with API correctly:

```
$ docker network create webAppNetwork
```

Create the database and connect it to the our network:

```
$ docker run --name database --network webAppNetwork  -p 5432:5432 -e POSTGRES_DB=student -e POSTGRES_PASSWORD=password
-e POSTGRES_USER=postgres -d postgres
```

Build the image of the API app:

```
$ docker build -t api-app .
```

Building the container of the API app and connecting it with same network of the database:

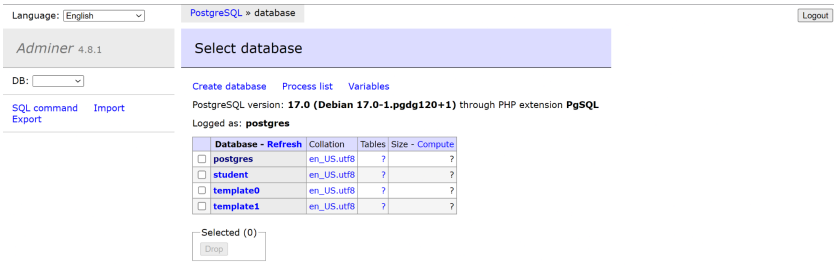
```
$ docker run -d --network webAppNetwork -p 8080:8080 --name api-app api-app
```

Exercise Three:

Running the database adminer container on the same netwrok of the database:

```
$ docker run --network webAppNetwork -p 8091:8080 adminer
```

Result after filling the form:



Exercise Four:

In this exercise we gonna add the docker compose file to build all of our container in one composer.
to run the docker compose:

```
$ docker compose up
```

the content of the docker compose documented here:

```yaml
# Docker Compose configuration version 3.9
version: "3.9"

services:
  # PostgreSQL Database Service
  database:
    image: postgres                 # Uses official PostgreSQL image
    restart: always                 # Automatically restarts if container stops
    environment:
      POSTGRES_DB: student          # Creates database named 'student'
      POSTGRES_USER: postgres       # Sets database user to 'postgres'
      POSTGRES_PASSWORD: password   # Sets database password to 'password'
    ports:
      - "5433:5432"                 # Maps host port 5433 to container port 5432
    networks:
      - backend                     # Connects to backend network
    volumes:
      - postgres_data:/var/lib/postgresql/data  # Persists database data
```

```yaml
# Database Management UI Service
adminer:
  image: adminer                 # Uses official Adminer image
  restart: always                # Automatically restarts if container stops
  ports:
    - "8091:8080"                # Maps host port 8091 to container port 8080
  networks:
    - backend                    # Connects to backend network

# API Service
api:
  build: ./api                   # Builds from Dockerfile in ./api directory
  restart: always                # Automatically restarts if container stops
  depends_on:
    - database                   # Ensures database starts first
  networks:
    - frontend                   # Connects to frontend network
    - backend                    # Connects to backend network
  volumes:
    - ./api:/app                 # Mounts local ./api directory to /app in container
  ports:
    - "8080:8080"               # Maps host port 8080 to container port 8080
```

```yaml
  # Web Application Service
  web-app:
    build: ./web-app             # Builds from Dockerfile in ./web-app directory
    restart: always              # Automatically restarts if container stops
    depends_on:
      - api                      # Ensures api starts first
      - database                 # Ensures database starts first
    networks:
      - frontend                 # Connects to frontend network
    ports:
      - "8090:5000"             # Maps host port 8090 to container port 5000

# Network Configurations
networks:
  backend:                       # Internal network for database communications
  frontend:                      # Internal network for web-facing services

# Volume Configurations
volumes:
  postgres_data:                 # Named volume for persisting database data
```