

به نام خدا

پروژه داکر ایز CRUD

لطف الله صالحی - ۴۰۰۲۲۷۷۰۵۲ - lotfullahsalehi4269@gmail.com

درس مربوطه: مبانی رایانش ابری - استاد: دکتر طاهری

برای انجام این پروژه از Ubuntu استفاده شده است (در ابتدا از MINT قرار بود استفاده بشه که بخاطر مشکلات مربوط به ماشین مجازی انجام نشد)

برای نصب docker از Documentation خود داکر استفاده شد که لینک مربوطه در زیر قرار داده شده است:

<https://docs.docker.com/desktop/setup/install/linux/ubuntu>

برای اطمینان از نصب صحیح داکر از دستور زیر استفاده میکنیم:

docker run hello-world

```
lotfi@lotfi-VMware-Virtual-Platform:~$ sudo docker run hello-world
[sudo] password for lotfi:

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

اگر چنین پیغامی دریافت کردیم به معنای نصب صحیح داکر است.

پروژه CRUD خود را ایجاد میکنیم و بعد به سراغ داکر ایز کردن آن میرویم.

ابتدا یک Dockerfile ایجاد میکنیم به این صورت:

```
Dockerfile
1  # Use an official Node.js runtime as a parent image
2  FROM node:18
3
4  # Set the working directory
5  WORKDIR /app
6
7  # Copy package.json and package-lock.json from the src folder
8  COPY package*.json .
9
10 # Install dependencies
11 RUN npm install
12
13 # Copy the rest of the application files from the src folder
14 COPY . .
15
16 # Expose the port the app runs on
17 EXPOSE 3000
18
19 # Command to run the app
20 CMD ["npm", "start"]
21
```

خط ۲: از یک ایمیج رسمی Node.js نسخه ۱۸ به عنوان پایه برای ساخت کانتینر استفاده می‌کند.

خط ۵: workdir داخل کانتینر را روی /app تنظیم می‌کند تا بقیه دستورات در این مسیر اجرا شوند.

خط ۸: فایل‌های package.json و package-lock.json را از پروژه به workdir داخل کانتینر کپی می‌کند.

خط ۱۱: دستور npm install را اجرا می‌کند تا تمام دپندنسی‌های پروژه نصب شوند.

خط ۱۴: تمام فایل‌های باقی‌مانده پروژه را از مسیر فعلی به دایرکتوری کاری داخل کانتینر کپی می‌کند.

خط ۱۷: پورت ۳۰۰۰ را برای دسترسی به اپلیکیشن در کانتینر باز می‌کند.

خط ۲۰: دستور npm start را اجرا می‌کند تا اپلیکیشن شروع به کار کند.

سپس یک فایل به نام docker-compose-app.yml به این صورت:

```
🐳 docker-compose-app.yml
1  version: '3.8'
2
3  services:
4    app:
5      build:
6        context: .
7      ports:
8        - "${PORT}:3000"
9      environment:
10       - DB_HOST=${DB_HOST}
11       - DB_USER=${DB_USER}
12       - DB_PASSWORD=${DB_PASSWORD}
13       - DB_NAME=${DB_NAME}
14      volumes:
15        - ./app
16        - /app/node_modules
17      networks:
18        - lotf-network
19
20  networks:
21    lotf-network:
22      external: true
23
```

این فایل یک سرویس به نام app تعریف می‌کند که کانتینر آن از دایرکتوری فعلی ساخته می‌شود. پورت ۳۰۰۰ داخل کانتینر به پورتهای که از مقدار فایل env برای PORT گرفته شده، مپ می‌شود. مقادیر دیتابیس DB_HOST ، DB_USER ،

DB_PASSWORD، و (DB_NAME نیز از فایل env خوانده و برای کانتینر تنظیم شده‌اند. دایرکتوری پروژه به app/در کانتینر متصل شده و پوشه node_modules از تغییرات مستثنی شده است.

سپس فایل docker-compose-db.yml به این صورت:

```
🐙 docker-compose-db.yml
1  version: '3.8'
2
3  services:
4    db:
5      image: mysql:8
6      ports:
7        - "3306:3306"
8      environment:
9        MYSQL_ROOT_PASSWORD: ${DB_PASSWORD}
10       MYSQL_DATABASE: ${DB_NAME}
11      volumes:
12        - db-data:/var/lib/mysql
13      networks:
14        - lotf-network
15
16  volumes:
17    db-data:
18
19  networks:
20    lotf-network:
21      external: true
22
```

در محیط لینوکس یک شبکه به نام lotf-network که در این پروژه بین دیتابیس و اپلیکیشن مشترک است ایجاد میکنیم:

docker network create lotf-network

حالا با این دستور فایل docker-compose-db.yml را اجرا میکنیم تا دیتابیس در کانتینر خود راه اندازی شود:

docker compose -f docker-compose-db.yml up -d

```
lotfi@lotfi-VMware-Virtual-Platform:~/crud-docker$ sudo docker compose -f docker-compose-db.yml up -d
[sudo] password for lotfi:
WARN[0000] /home/lotfi/crud-docker/docker-compose-db.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
WARN[0000] Found orphan containers ([crud-docker-app-1]) for this project. If you removed or renamed this service in your compose file, you can run this command with the --remove-orphans flag to clean it up.
[+] Running 1/1
 ✓ Container crud-docker-db-1 Started                                0.4s
lotfi@lotfi-VMware-Virtual-Platform:~/crud-docker$
```

و سپس فایل docker-compose-app.yml که شامل ساخت ایمج و اجرای کانتینر اپلیکیشن میشود:

docker compose -f docker-compose-app.yml up --build

```
lotfi@lotfi-VMware-Virtual-Platform:~/crud-docker$ sudo docker compose -f docker-compose-app.yml up --build
WARN[0000] /home/lotfi/crud-docker/docker-compose-app.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Building 0.0s (0/1)
[+] Running 0/1
[+] Building 3.0s (11/11) FINISHED
=> [app internal] load build definition from Dockerfile
=> => transferring dockerfile: 435B
=> [app internal] load metadata for docker.io/library/node:18
=> [app internal] load .dockerignore
=> => transferring context: 2B
=> [app internal] load build context
=> => transferring context: 2.38kB
=> [app 1/5] FROM docker.io/library/node:18@sha256:7f31a1eb14c61719b0bb0eaa029318cc33851f71d3578cc422b398f8096977c5
=> CACHED [app 2/5] WORKDIR /app
=> CACHED [app 3/5] COPY package*.json .
=> CACHED [app 4/5] RUN npm install
=> CACHED [app 5/5] COPY . .
=> [app] exporting to image
=> => exporting layers
=> => writing image sha256:2e5848ad55ecf58dad214e98c3371aa58771e76cfcea26deabe56e10bbdb3ce0
=> => naming to docker.io/library/crud-docker-app
=> [app] resolving provenance for metadata file
[+] Running 2/2d orphan containers ([crud-docker-db-1]) for this project. If you removed or renamed this service in your compose file, you can run this command with the --remove-orphans flag to clean it up.
 ✓ Container crud-docker-app-1 Recreated
Attaching to app-1
app-1 |
app-1 | > lotfi@1.0.0 start
app-1 | > node server.js
app-1 |
app-1 | Server running at http://localhost:3000
app-1 | Connected to MySQL
[+] Service app Built
Enable Watch
```

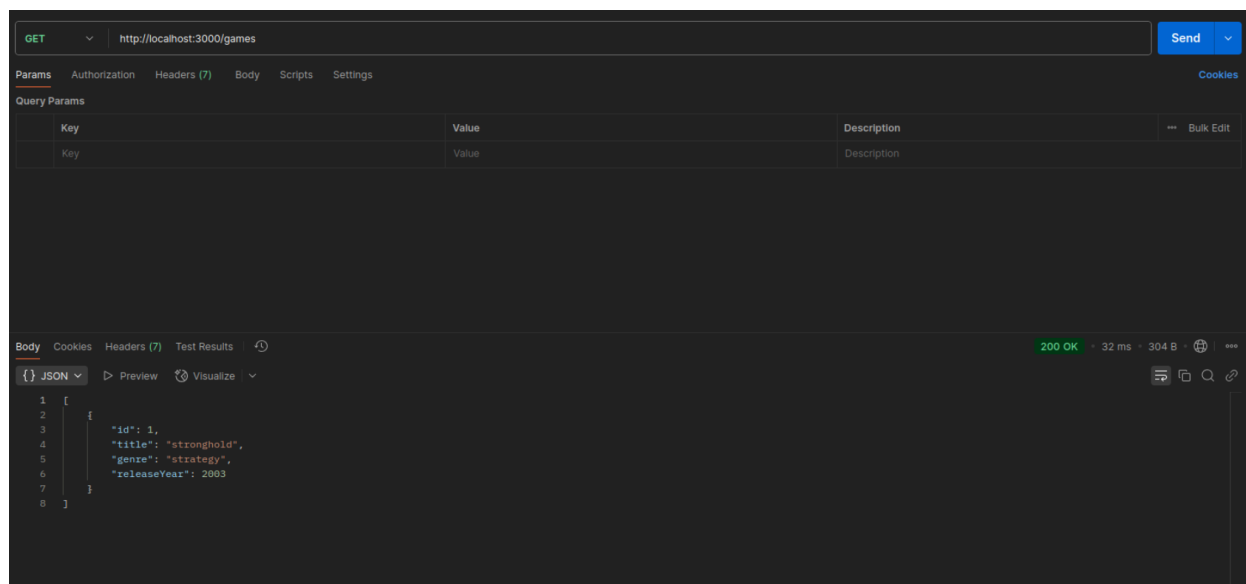
بعد از اجرای کامل به صورت بالا خواهد بود.

همانطور که در تصویر مشاهده میکنیم نوشته شده است:

Server running at <http://localhost:3000>

با وارد کردن این آدرس در Postman آن را تست میکنیم: (دقت داشته باشید که اگر بخواهیم از Postman تحت وب استفاده کنیم حتما باید postman agent را راه اندازی کنیم)

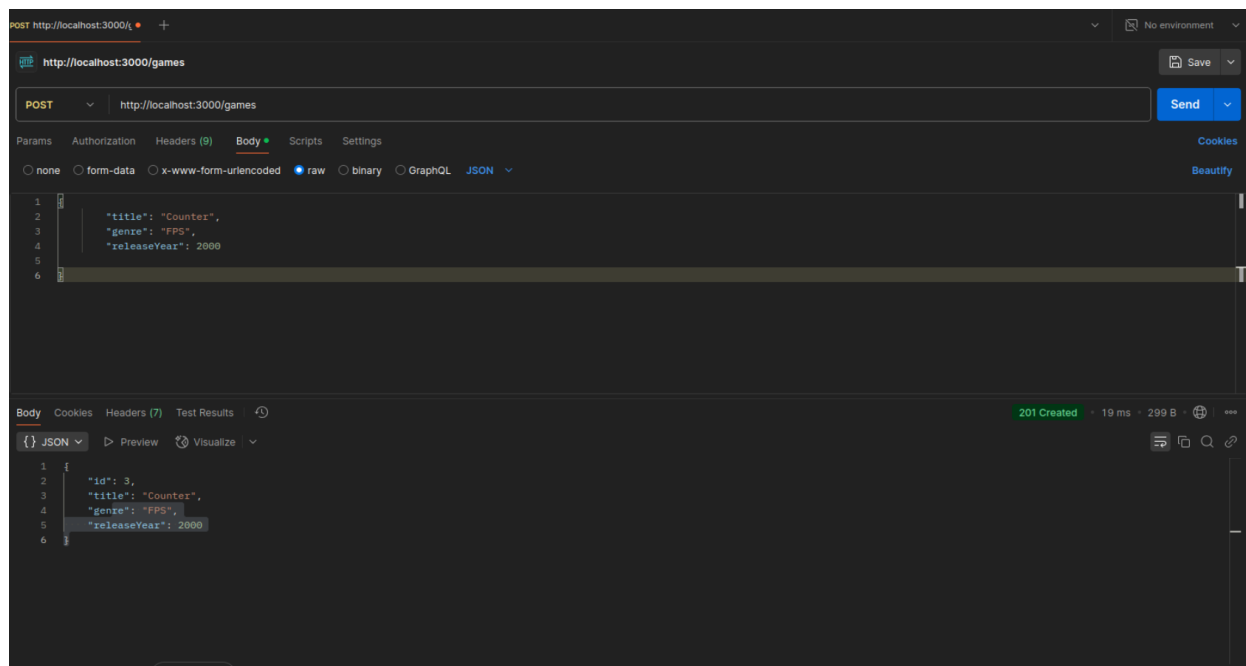
دستور GET



The screenshot shows the Postman interface for a GET request to `http://localhost:3000/games`. The request is successful, returning a `200 OK` status with a response time of `32 ms` and a body size of `304 B`. The response body is a JSON array containing one game object:

```
1 [
2   {
3     "id": 1,
4     "title": "stronghold",
5     "genre": "strategy",
6     "releaseYear": 2003
7   }
8 ]
```

دستور POST



The screenshot shows the Postman interface for a POST request to `http://localhost:3000/games`. The request is successful, returning a `201 Created` status with a response time of `19 ms` and a body size of `289 B`. The request body is a JSON object:

```
1 {
2   "title": "Counter",
3   "genre": "FPS",
4   "releaseYear": 2000
5 }
6
```

The response body is a JSON object containing the created game:

```
1 {
2   "id": 3,
3   "title": "Counter",
4   "genre": "FPS",
5   "releaseYear": 2000
6 }
```

که `201 Created` که به رنگ سبز نمایش داده شده به معنای اجرا صحیح دستور میباشد.

دستور PUT (یا همان Update)

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:3000/games/3`
- Method:** `PUT`
- Body:** A JSON object:

```
{  "title": "Counter Strike",  "genre": "FPS",  "releaseYear": 2004}
```
- Response:** `200 OK` with a response body:

```
{  "message": "Game updated successfully"}
```

دستور Delete

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:3000/games/3`
- Method:** `DELETE`
- Body:** A JSON object (shown for reference):

```
{  "title": "Counter Strike",  "genre": "FPS",  "releaseYear": 2004}
```
- Response:** `200 OK` with a response body:

```
{  "message": "Game deleted successfully"}
```

نکاتی که قابل توجه هستند:

من پروژه رو در محیط ویندوز ایجاد کردم و آن را بر روی github خودم push کردم، و سپس آن را در لینوکس clone کردم، نکته ای که باید توجه داشته باشیم این است که ابتدا باید یک فایل gitignore. به این صورت داشته باشیم:

```
.gitignore
1  node_modules
2  .env
3
```

تا node modules و فایل .env را push نکند به گیت، از آنجایی که به .env نیاز است باید آن را به صورت فایل .env.example ایجاد کنیم و سپس push کنیم.

CRUD-NODE	\$.env.example
.env	1 PORT=3000
.env.example	2 DB_HOST=db
.gitignore	3 DB_USER=root
docker-compose-app.yml	4 DB_PASSWORD=rootpassword
docker-compose-db.yml	5 DB_NAME=games
Dockerfile	

همچنین با توجه به نکته ای که ذکر شد، باید تیبل games در لینوکس را نیز ایجاد کنم که برای این کار یک دیتابیس جدید ایجاد کرده و به آن متصل میشوم و بعد با کوئری زیر ایجاد میشود:

```
CREATE TABLE games (
  id INT AUTO_INCREMENT PRIMARY KEY,
  title VARCHAR(255) NOT NULL,
  genre VARCHAR(100) NOT NULL,
  releaseYear INT NOT NULL
);
```


در نهایت شایان ذکر است که دشواری اصلی این پروژه اتصالات اینترنت بود که به علت تحریم ایران توسط docker باید با استفاده از dns انجام میشد و اجرای دستوراتی مثل docker compose up که نیاز به اینترنت دارند و حجم اینترنت زیادی مصرف میکنند بسیار طاقت فرسا بود که شخصا برای این مشکل دی ان اس الکترو را پیشنهاد میکنم که بسیار سرعت مناسبی داشت:

78.157.42.100

78.157.42.101