

DATU MASIBOEN PROZESAMENDURAKO AZPIEGITURAK

1.LANA



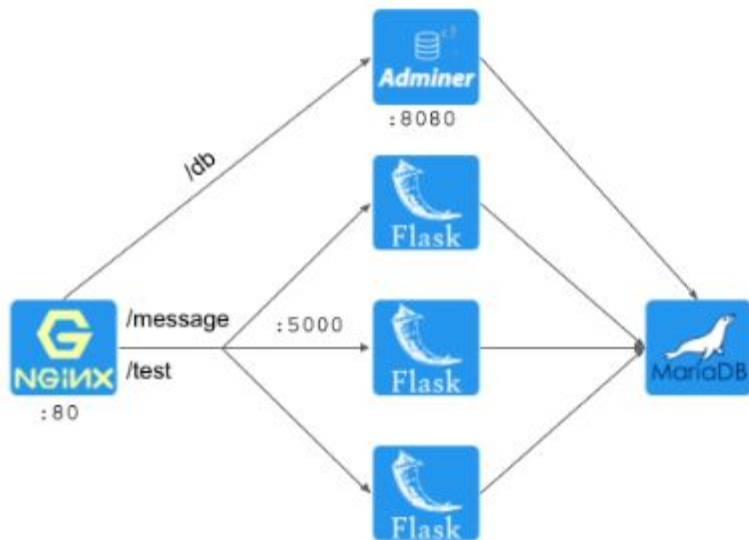
Egileak: Andoni Sudupe, Oier Ijurko

AURKIBIDEA

- Eskatzen zaiguna (3.D)
- Compose.yaml (5.D)
- Datu basearen sorkuntza (6.D)
- Dockerfile (7.D)
- nginx.conf (8.D)
- Aplikazioa (9.D)
- Egindako hobekuntzak (14.D)
- Emaitzak (17.D)

ESKATZEN ZAIGUNA

- NGINX kontainer bat eskaerak jasotzen dituen eta huen arabera kontainer desberdinetan berbidaltzen duena
- Flask Aplikazio bat implementatzen duen kontainer replika multzoa
- Adminer kontainer bat, mariadb duen beste kontainer bati lotuta dagoena
- Hau guztia gauzatzeko, beharrezkoak diren `compose.yaml`, `nginx.conf`, `Dockerfile` eta `app.py` fitxategiak implementatzea



Requests

From

Content

Cont_ID

Compose.yaml fitxategia

Hedapen guztia kudeatzen duen fitxategia da

Hauek definitzen dira bertan:

- nginx zerbitzaria
- Flask aplikazioaren 3 replikak
- Adminer datubase ingurunea
- MariaDB datubasea

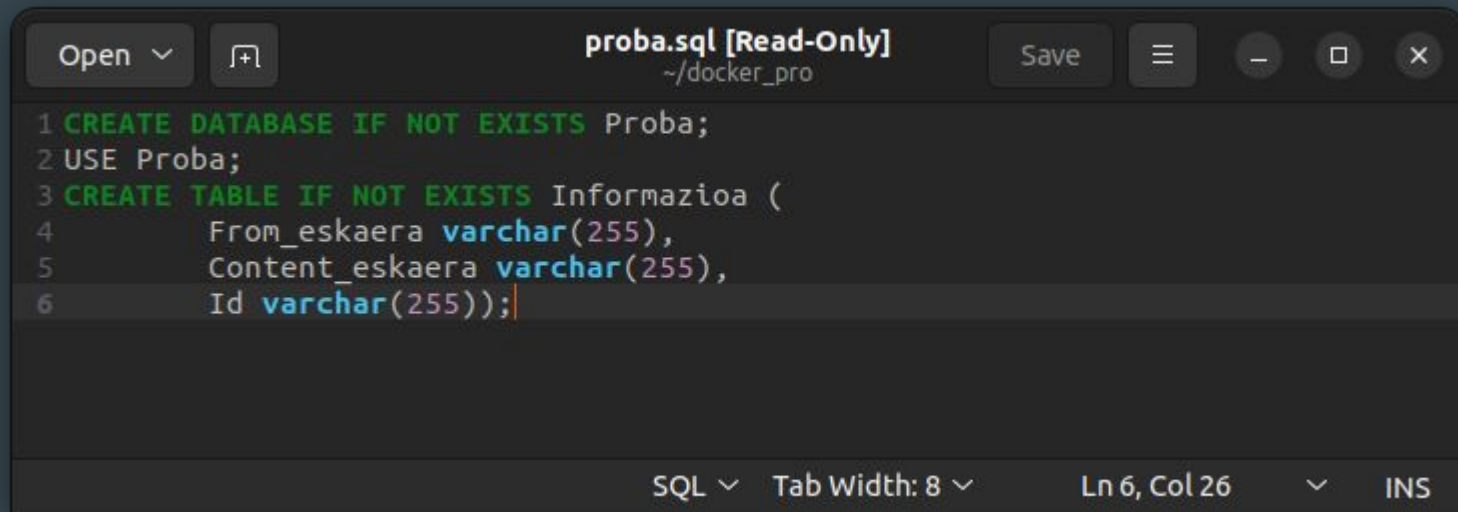
```
compose.yaml
~/docker_pro

1 services:
2   web:
3     build: .
4     deploy:
5       mode: replicated
6       replicas: 3
7     depends_on:
8       - db
9   # --- database - 'mariadb' ---
10  db:
11    image: mariadb
12    container_name: gure-mariadb
13    command: --init-file /var/lib/mysql/proba.sql
14    volumes:
15      - ./dbdata:/var/lib/mysql
16      - ./proba.sql:/var/lib/mysql/proba.sql
17    ports:
18      - "3306:3306"
19    expose:
20      - '3306'
21    environment:
22      MYSQL_ROOT_PASSWORD: changeme
23      MYSQL_DATABASE: Proba
24      MYSQL_USER: user
25      MYSQL_PASSWORD: pass
26
27  # --- server - 'nginx' ---
28  nginx:
29    image: nginx:1.23.3
30    container_name: gure-nginx
31    volumes:
32      - ./nginx.conf:/etc/nginx/nginx.conf
33    ports:
34      - "80:80"
35    depends_on:
36      - web
37      - adminer
38  # --- database environment - 'adminer' ---
39  adminer:
40    image: adminer
41    container_name: gure_adminer
42    ports:
43      - "8080:8080"
44    depends_on:
45      - db
```

YAML ▾ Tab Width: 8 ▾ Ln 45, Col 11 ▾ INS

DATU BASEAREN SORKUNTZA

- .sql fitxategi batekin sortuko dugu
- docker compose up egiterakoan exekutatu da

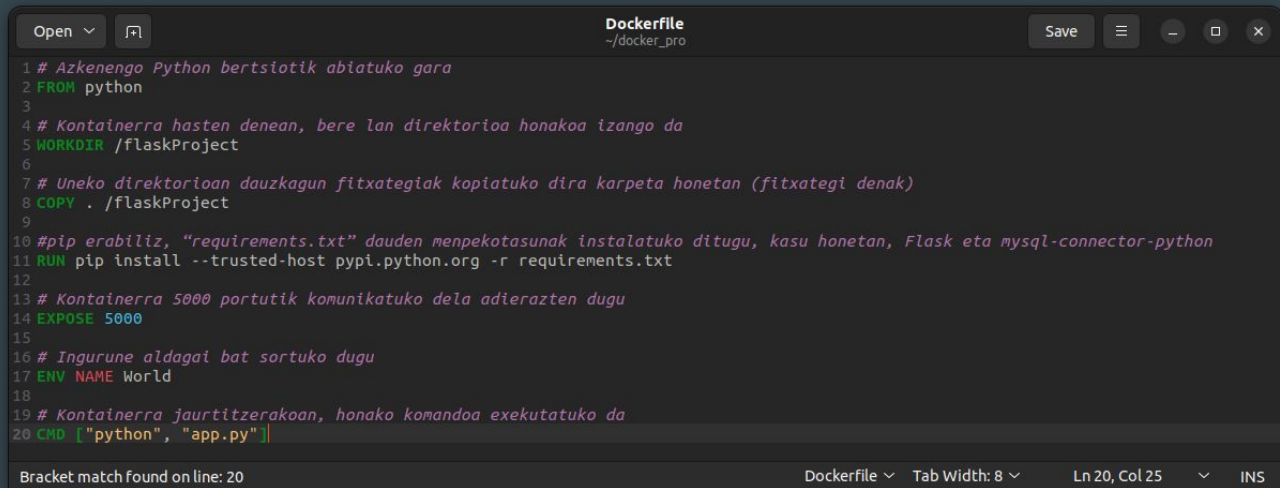


The image shows a code editor window with a dark theme. The title bar indicates the file is 'proba.sql [Read-Only]' located at '~/docker_pro'. The editor contains six lines of SQL code: 1. 'CREATE DATABASE IF NOT EXISTS Proba;', 2. 'USE Proba;', 3. 'CREATE TABLE IF NOT EXISTS Informazioa (' (with an opening parenthesis), 4. ' From_eskaera varchar(255),', 5. ' Content_eskaera varchar(255),', and 6. ' Id varchar(255));'. The cursor is at the end of line 6. The bottom status bar shows 'SQL', 'Tab Width: 8', 'Ln 6, Col 26', and 'INS'.

```
1 CREATE DATABASE IF NOT EXISTS Proba;
2 USE Proba;
3 CREATE TABLE IF NOT EXISTS Informazioa (
4     From_eskaera varchar(255),
5     Content_eskaera varchar(255),
6     Id varchar(255));
```

Dockerfile

- Komando-multzo bat duen testu-fitxategia da.
- Docker irudi pertsonalizatuak sortzeko erabiltzen da.



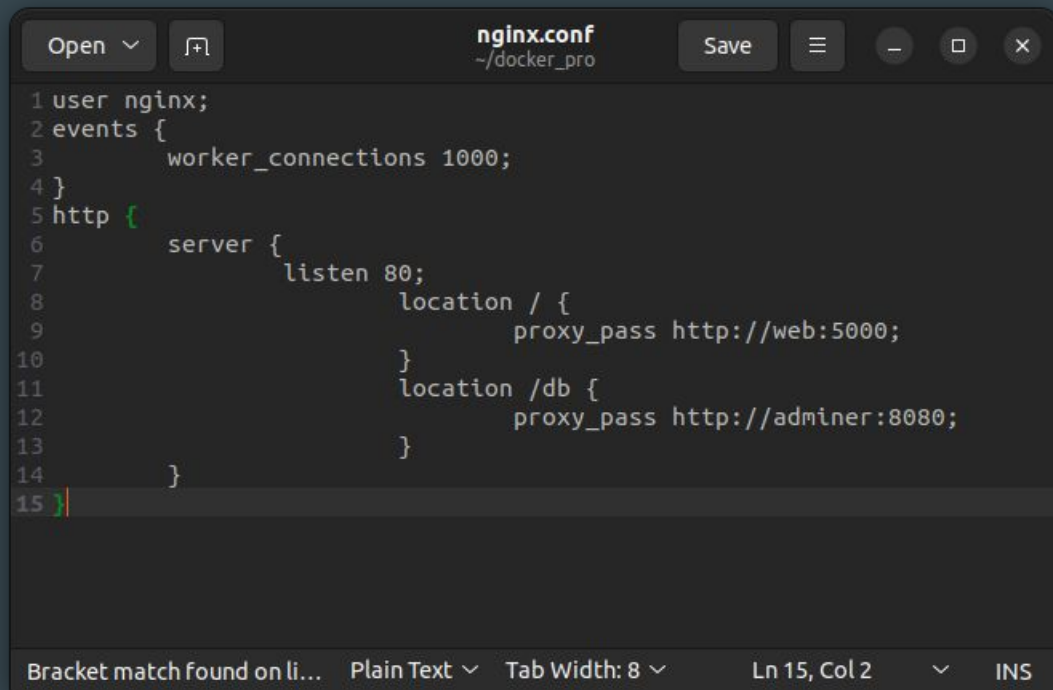
```
1 # Azkenengo Python bertsiotik abiatuko gara
2 FROM python
3
4 # Kontainerra hasten denean, bere lan direktorioa honakoa izango da
5 WORKDIR /flaskProject
6
7 # Uneke direktorioan dauzkagun fitxategiak kopiatuko dira karpeta honetan (fitxategi denak)
8 COPY . /flaskProject
9
10 #pip erabiliz, "requirements.txt" dauden menpekotasunak instalatuko ditugu, kasu honetan, Flask eta mysql-connector-python
11 RUN pip install --trusted-host pypi.python.org -r requirements.txt
12
13 # Kontainerra 5000 portutik komunikatuko dela adierazten dugu
14 EXPOSE 5000
15
16 # Ingurune aldagai bat sortuko dugu
17 ENV NAME World
18
19 # Kontainerra jaurtitzerakoan, honako komandoa exekutatuko da
20 CMD ["python", "app.py"]
```

Bracket match found on line: 20

Dockerfile Tab Width: 8 Ln 20, Col 25 INS

nginx.conf fitxategia

- nginx kontainerren konfigurazioa duen fitxategia



```
1 user nginx;
2 events {
3     worker_connections 1000;
4 }
5 http {
6     server {
7         listen 80;
8         location / {
9             proxy_pass http://web:5000;
10        }
11        location /db {
12            proxy_pass http://adminer:8080;
13        }
14    }
15 }
```

Bracket match found on li... Plain Text Tab Width: 8 Ln 15, Col 2 INS

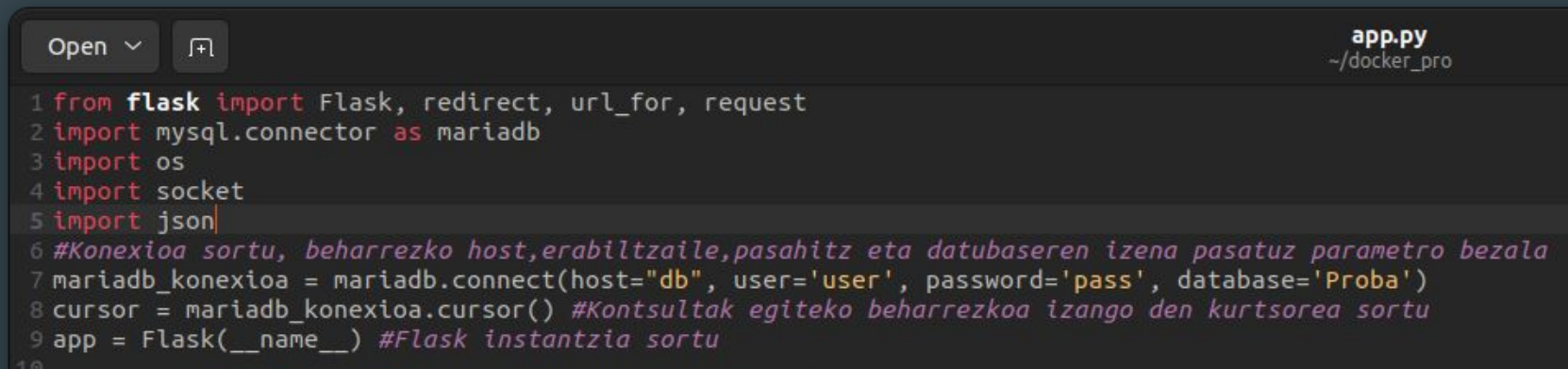
app.py

- Zerbitzaria martxan jarri ondoren exekutatu den python programa

```
Open  app.py -/docker_pro Save
1 from flask import Flask, redirect, url_for, request
2 import mysql.connector as mariadb
3 import os
4 import socket
5 import json
6 #Konexioa sortu, beharrezko host,erabiltzaile,pasahitz eta datubaseren izena pasatuz parametro bezala
7 mariadb_konexioa = mariadb.connect(host="db", user="user", password="pass", database="Proba")
8 cursor = mariadb_konexioa.cursor() #kontsultak egiteko beharrezkoa izango den kurtsorea sortu
9 app = Flask(__name__) #Flask instantzia sortu
10
11 @app.route("/db") #Adminer kontainerreara joateko eskaeraren ruta
12 def db():
13     pass #nginx konfigurazio fitxategian berbidatzen du adminerara iada
14
15 @app.route("/message", methods=['GET', 'POST']) #Bi metodoa posible dituen message ruta, get eta post
16 def message():
17     if request.method == 'GET': #Datu basearen erregistro guztiak bueltatu
18         try:
19             from_eskaera = request.args.get('From') #eskaeraren edukia gordetzen dugu
20             if str(from_eskaera) == "ALL":
21                 cursor.execute("SELECT * FROM Informazioa") #Beharrezko taularen gainean eskaera
22                 return cursor.fetchall()
23             else:
24                 a = "SELECT * FROM Informazioa WHERE From_eskaera = '" + str(from_eskaera) + "'" #Beharrezko taularen gainean eskaera
25                 cursor.execute(a)
26                 return cursor.fetchall() #Lortutako emaitzak bueltatzeko beharrezko komandoa
27         except Exception as e:
28             cursor.execute("SELECT * FROM Informazioa") #Beharrezko taularen gainean eskaera
29             return cursor.fetchall()
30     elif request.method == 'POST': #Eskaeraren edukia datu basean gordeko du. Json formatuko edukia
31         try:
32             from_eskaera = request.json["From"] #eskaeraren edukia gordetzen dugu
33             content_eskaera = request.json["Content"] #eskaeraren edukia gordetzen dugu
34             id = socket.gethostname() #Uneko flask kontainerrearen id-a lortu
35             query = ("INSERT INTO Informazioa (From_eskaera, Content_eskaera, Id)"
36                     "VALUES (%s, %s, %s)") #Gordetzeko query-a
37             cursor.execute(query, (from_eskaera, content_eskaera, id)) #Exekutatu query-a
38             mariadb_konexioa.commit() #Aldaketak burutu
39             return "Gordeta"
40         except Exception as e: #Post eskaera desegoki bat tratatzeko mezua
41             return ("Post eskaera desegokia izan da."
42                     "Itxura honetako del batekin salatu: curl -d {From:AAB, Content:ttt}"
43                     "-H Content Type: application/json -X POST localhost:80/message")
44
45 @app.route("/test", methods=['GET']) #Test rutako eskaera
46 def test(): #ALIVE testua bueltatuko digu, bere flask kontainerrearen id-arekin batera
47     try:
48         return "ALIVE "+ socket.gethostname() #buelatu alive eta kontainerrearen id-a
49     except Exception as e:
50         return "Test eskaera desegokia. Itxura honetako del batekin salatu: localhost:80/test" #Ez da gertatuko, baina bazpadaere
51
52 if __name__ == "__main__":
53     app.run(host='0.0.0.0', port=5000)
```

Python 2 Tab Width: 8 Ln 5, Col 12 INS

Konexioa eta kursorea



The image shows a code editor window with a dark theme. At the top, there is a tab labeled 'app.py' with a file icon to its left. Below the tab, the file path '~/.docker_pro' is visible. The main area of the editor contains Python code with syntax highlighting. The code imports Flask, mysql.connector, os, socket, and json. It then establishes a MySQL connection using mariadb.connect() with host 'db', user 'user', password 'pass', and database 'Proba'. A cursor is created with mariadb_konexioa.cursor(). Finally, a Flask application instance is created with Flask(__name__).

```
1 from flask import Flask, redirect, url_for, request
2 import mysql.connector as mariadb
3 import os
4 import socket
5 import json
6 #Konexioa sortu, beharrezko host,erabiltzaile,pasahitz eta datubaseren izena pasatuz parametro bezala
7 mariadb_konexioa = mariadb.connect(host="db", user='user', password='pass', database='Proba')
8 cursor = mariadb_konexioa.cursor() #Kontsultak egiteko beharrezkoa izango den kurtsoa sortu
9 app = Flask(__name__) #Flask instantzia sortu
10
```

Adminerrera joateko

```
10
11 @app.route("/db") #Adminer kontainerreko joateko eskaeraren ruta
12 def db():
13     pass #nginx konfigurazio fitxategian berbideratzen du adminerrera iada
14
```

POST /message eta GET /message

```
14
15 @app.route("/message", methods=['GET', 'POST']) #Bi metodo posible dituen message ruta, get eta post
16 def message():
17     if request.method == 'GET': #Datu basearen erregistro guztiak bueltatu
18         try:
19             from_eskaera = request.args.get('From') #eskaeraren edukia gordetzen dugu
20             if str(from_eskaera) == "ALL":
21                 cursor.execute("SELECT * FROM Informazioa") #Beharrezko taularen gainean eskaera
22                 return cursor.fetchall()
23             else:
24                 a = "SELECT * FROM Informazioa WHERE From_eskaera = '" + str(from_eskaera) + "'" #Beharrezko taularen gainean eskaera
25                 cursor.execute(a)
26                 return cursor.fetchall() #Lortutako emaitzak bueltatzeko beharrezko komandoa
27         except Exception as e:
28             cursor.execute("SELECT * FROM Informazioa") #Beharrezko taularen gainean eskaera
29             return cursor.fetchall()
30     elif request.method == 'POST': #Eskaeraren edukia datu basean gordeko du. Json formatuko edukia
31         try:
32             from_eskaera = request.json["From"] #eskaeraren edukia gordetzen dugu
33             content_eskaera = request.json["Content"] #eskaeraren edukia gordetzen dugu
34             id = socket.gethostname() #Uneo flask kontainerraren id-a lortu
35             query = ("INSERT INTO Informazioa (From_eskaera, Content_eskaera, Id)"
36                    "VALUES (%s, %s, %s)") #Gordetzeko query-a
37             cursor.execute(query, (from_eskaera, content_eskaera, id)) #Exekutatu query-a
38             mariadb.konexioa.commit() #Aldaketak burutu
39             return "Gordeta"
40         except Exception as e: #Post eskaera desegoki bat tratatzeko mezua
41             return ("Post eskaera desegokia izan da."
42                    "Itxura honetako del batekin saiatu: curl -d {From:AAB, Content:ttt}"
43                    "-H Content Type: application/json -X POST localhost:80/message")
44
```

GET /test

```
43         "-H Content Type: application/json -X POST localhost:80/message")
44
45 @app.route("/test", methods=['GET']) #Test rutako eskaera
46 def test(): #ALIVE testua bueltatuko digu, bere flask kontainerraren id-arekin batera
47     try:
48         return "ALIVE "+ socket.gethostname() #buelatu alive eta kontainerraren id-a
49     except Exception as e:
50         return "Test eskaera desegokia. Itxura honetako del batekin saiatu: localhost:80/test" #Ez da gertatuko, baina bazpadaere
51
52 if __name__ == "__main__":
53     app.run(host='0.0.0.0', port=5000)
```

Python 2 ▾

Tab Width: 8 ▾

Ln 5, Col 12 ▾

INS

EGINDAKO HOBKUNTZAK

- Eskaeren erroreak kudeatzea

```
try:
    from_eskaera = request.args.get('From') #eskaeraren edukia gordetzen dugu
    if str(from_eskaera) == "ALL":
        cursor.execute("SELECT * FROM Informazioa") #Beharrezko taularen gainean eskaera
        return cursor.fetchall()
    else:
        a = "SELECT * FROM Informazioa WHERE From_eskaera = '" + str(from_eskaera) + "'" #Beharrezko taularen gainean eskaera
        cursor.execute(a)
        return cursor.fetchall() #Lortutako emaitzak bueltatzeko beharrezko komandoa
except Exception as e:
    cursor.execute("SELECT * FROM Informazioa") #Beharrezko taularen gainean eskaera
    return cursor.fetchall()
```

```

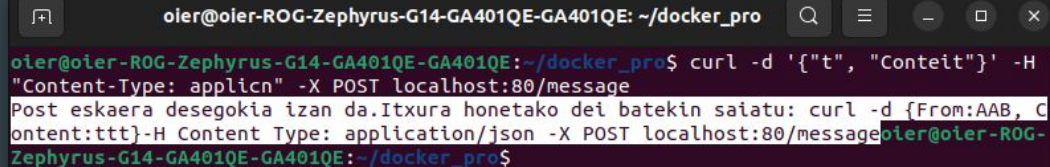
try:
    from_eskaera = request.json["From"] #eskaeraren edukia gordetzen dugu
    content_eskaera = request.json["Content"] #eskaeraren edukia gordetzen dugu
    id = socket.gethostname() #Uneko flask kontainerraren id-a lortu
    query = ("INSERT INTO Informazioa (From_eskaera, Content_eskaera, Id)"
            "VALUES (%s, %s, %s)") #Gordetzeko query-a
    cursor.execute(query, (from_eskaera, content_eskaera, id)) #Exekutatu query-a
    mariadb_konexioa.commit() #Aldaketak burutu
    return "Gordeta"
except Exception as e: #Post eskaera desegoki bat tratatzeko mezua
    return ("Post eskaera desegokia izan da."
            "Itxura honetako dei batekin saiatu: curl -d {From:AAB, Content:ttt}"
            "-H Content Type: application/json -X POST localhost:80/message")

```

```

try:
    return "ALIVE " + socket.gethostname() #buelatu alive eta kontainerraren id-a
except Exception as e:
    return "Test eskaera desegokia. Itxura honetako dei batekin saiatu: localhost:80/test" #Ez da gertatuko, baina bazpadaere

```



A terminal window with a title bar showing the user 'oier' and the host 'oier-ROG-Zephyrus-G14-GA401QE-GA401QE' in the directory '~/docker_pro'. The terminal shows a curl command being executed: `curl -d '{"t", "Conteit"}' -H "Content-Type: applicn" -X POST localhost:80/message`. The output of the command is displayed: `Post eskaera desegokia izan da.Itxura honetako dei batekin saiatu: curl -d {From:AAB, Content:ttt}-H Content Type: application/json -X POST localhost:80/message`. The prompt at the end is `oier@oier-ROG-Zephyrus-G14-GA401QE-GA401QE:~/docker_pro$`.

```

oier@oier-ROG-Zephyrus-G14-GA401QE-GA401QE: ~/docker_pro
oier@oier-ROG-Zephyrus-G14-GA401QE-GA401QE:~/docker_pro$ curl -d '{"t", "Conteit"}' -H
"Content-Type: applicn" -X POST localhost:80/message
Post eskaera desegokia izan da.Itxura honetako dei batekin saiatu: curl -d {From:AAB, C
ontent:ttt}-H Content Type: application/json -X POST localhost:80/messageoier@oier-ROG-
Zephyrus-G14-GA401QE-GA401QE:~/docker_pro$

```


- "GET /message" eskaeren emaitza filtratzea

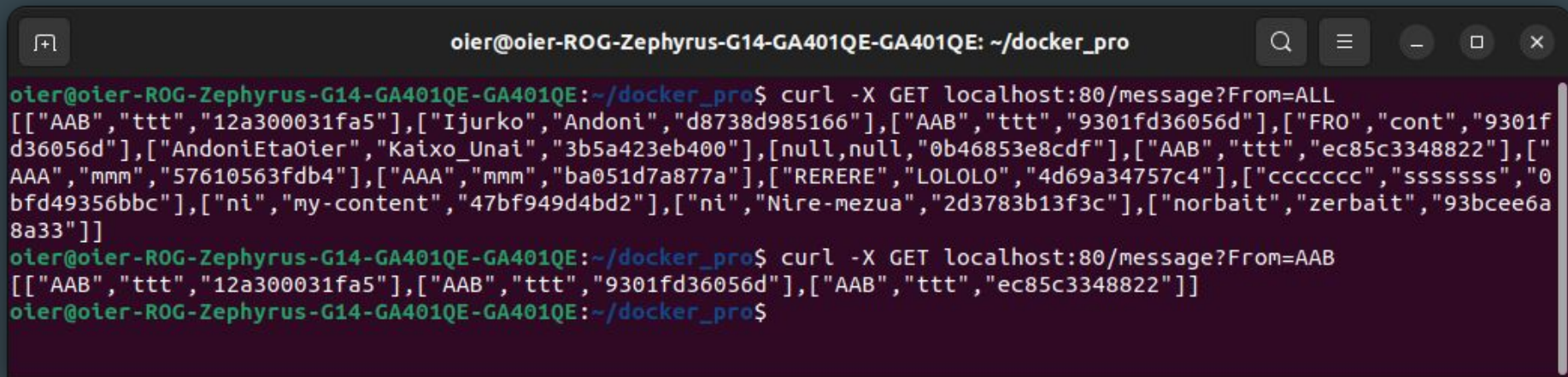
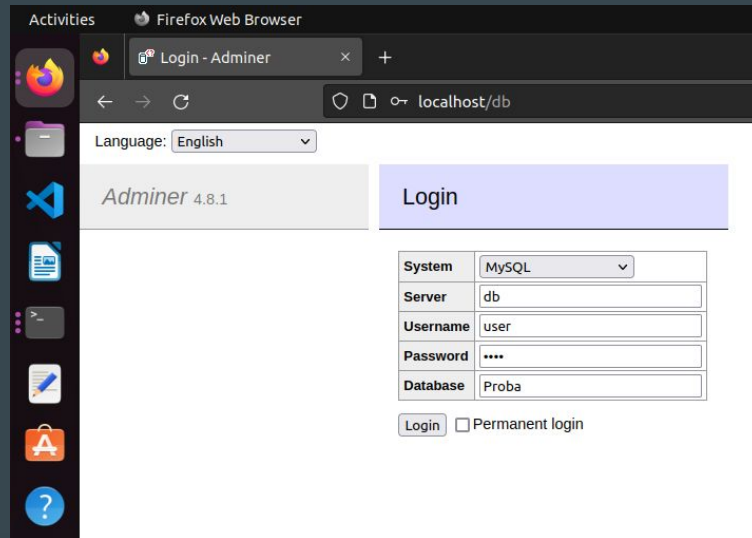
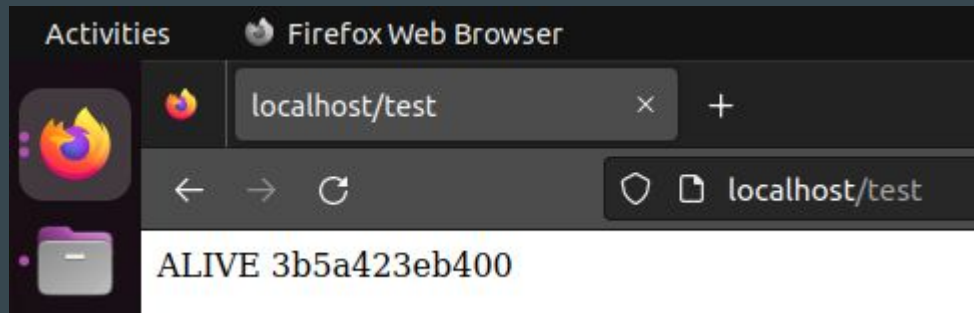
```
from_eskaera = request.args.get('From') #eskaeraren edukia gordetzen dugu
if str(from_eskaera) == "ALL":
    cursor.execute("SELECT * FROM Informazioa") #Beharrezko taularen gainean eskaera
    return cursor.fetchall()
else:
    a = "SELECT * FROM Informazioa WHERE From_eskaera = '" + str(from_eskaera) + "'" #Beharrezko taularen gainean eskaera
    cursor.execute(a)
    return cursor.fetchall() #Lortutako emaitzak bueltatzeko beharrezko komandoa
```

Erregistro guztiak →

Erregistro filtratuak →

```
oier@oier-ROG-Zephyrus-G14-GA401QE-GA401QE: ~/docker_pro
oier@oier-ROG-Zephyrus-G14-GA401QE-GA401QE:~/docker_pro$ curl -X GET localhost:80/message?From=ALL
[["AAB","ttt","12a300031fa5"],["Ijurko","Andoni","d8738d985166"],["AAB","ttt","9301fd36056d"],["FRO","cont","9301fd36056d"],["AndoniEtaOier","Kaixo_Unai","3b5a423eb400"],["null,null","0b46853e8cdf"],["AAB","ttt","ec85c3348822"],["AAA","mmm","57610563fdb4"],["AAA","mmm","ba051d7a877a"],["RERERE","LOLOLO","4d69a34757c4"],["ccccccc","sssssss","0bfd49356bbc"],["ni","my-content","47bf949d4bd2"],["ni","Nire-mezua","2d3783b13f3c"],["norbait","zerbait","93bcee6a8a33"]]
oier@oier-ROG-Zephyrus-G14-GA401QE-GA401QE:~/docker_pro$ curl -X GET localhost:80/message?From=AAB
[["AAB","ttt","12a300031fa5"],["AAB","ttt","9301fd36056d"],["AAB","ttt","ec85c3348822"]]
oier@oier-ROG-Zephyrus-G14-GA401QE-GA401QE:~/docker_pro$
```


Emaitzak



```
oier@oier-ROG-Zephyrus-G14-GA401QE-GA401QE: ~/docker_pro
oier@oier-ROG-Zephyrus-G14-GA401QE-GA401QE:~/docker_pro$ curl -d '{"From":"aurkezpena2", "Content":"aurkezpen2"}'
-H "Content-Type: application/json" -X POST localhost:80/message
Gordetaoier@oier-ROG-Zephyrus-G14-GA401QE-GA401QE:~/docker_pro$
```

select Informazioa

SELECT * FROM 'Informazioa' LIMIT 50 (0.000 s) Edit

<input type="checkbox"/> Modify	From_eskaera	Content_eskaera	Id
<input type="checkbox"/> edit	AAB	ttt	12a300031fa5
<input type="checkbox"/> edit	Ijurko	Andoni	d8738d985166
<input type="checkbox"/> edit	AAB	ttt	9301fd36056d
<input type="checkbox"/> edit	FRO	cont	9301fd36056d
<input type="checkbox"/> edit	AndoniEtaOier	Kaixo_Unai	3b5a423eb400
<input type="checkbox"/> edit	NULL	NULL	0b46853e8cdf
<input type="checkbox"/> edit	AAB	ttt	ec85c3348822
<input type="checkbox"/> edit	AAA	mmm	57610563fdb4
<input type="checkbox"/> edit	AAA	mmm	ba051d7a877a
<input type="checkbox"/> edit	RERERE	LOLOLO	4d69a34757c4
<input type="checkbox"/> edit	cccccc	ssssss	0bfd49356bbc
<input type="checkbox"/> edit	ni	my-content	47bf949d4bd2
<input type="checkbox"/> edit	ni	Nire-mezua	2d3783b13f3c
<input type="checkbox"/> edit	norbait	zerbait	93bcee6a8a33
<input type="checkbox"/> edit	aurkezpena	aurkezpenerako	f8aec5e381f7
<input type="checkbox"/> edit	aurkezpena2	aurkezpen2	16a683aebcae

**ESKERRIK ASKO ZUEN
ARRETAGATIK**