Secure Remote Control:
Local database setup

For this project, MongoDB is used as the primary database, but any other NoSQL database can be used as well.

The database is deployed on MongoDB Atlas, while for local development and testing, a Docker image is used to run the database inside a container.

To manage the database structure, the tool "mongo-migrate" is used, which allows version control and migration management.

MongoDB was chosen due to its flexibility in handling unstructured data, and because it allows multiple services to access the database without conflicts.

Since the system uses two independent services that communicate with the database, it was important to avoid complications related to concurrent access and simultaneous migrations.

MongoDB with the mongo-migrate tool enables controlled migrations, preventing issues in multi-environment access.

INSTRUCTIONS

First, make sure Docker is installed on your computer:

https://www.docker.com/products/docker-desktop

You also need to install the Mongo shell. To install it, do the following:

For Windows:
Go to the following link:
https://www.mongodb.com/try/download/shell

Choose the **msi** option under packages
After installation, add **mongosh** to your PATH

For macOS:
Open the terminal and run the following command:
brew install mongodb/brew/mongo-shell

For Linux:

Open the terminal and follow the official MongoDB documentation for your distribution.
For Ubuntu and Debian-based systems, you can run the following commands:

curl -fsSL https://pgp.mongodb.com/server-6.0.asc | sudo gpg -o /usr/share/keyrings/mongodb-server-6.0.gpg --dearmor

echo "deb [signed-by=/usr/share/keyrings/mongodb-server-6.0.gpg] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/6.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-6.0.list

sudo apt update
sudo apt install -y mongodb-mongosh

This will install the MongoDB shell (`mongosh`).

After installation, you can run **mongosh** in the terminal to enter the Mongo shell.

Setup:

After cloning or pulling the repository and accessing the project code, you need the .env file we created for the **backend:**

PORT=8080
DB_URI='mongodb+srv://root:root@cluster.qciyr2x.mongodb.net/Cluster?retryWrites=true&w=majority&appName=Cluster'
DB_URI_LOCAL='mongodb://localhost:27017/'
USE_LOCAL_DB=true

These are the two URIs used to connect to the deployed and local databases, respectively.

Make sure that when **testing**, you are using your local database by setting **USE_LOCAL_DB=true**, so that you avoid **interfering** with the production database.

In the terminal, run the following command to install all required dependencies:

npm install

Then, to start the application:

npm start

In the project folder, there is a file called **docker-compose.yml**

Run the following command:

docker-compose up -d

This will start the Docker container.

Check if the container is running by typing the following:

docker ps

You should see a list of containers.

Look for the one named **local-mongo**

Then run the following command

docker exec -it local-mongo mongosh

This will open the interactive Mongo shell inside the locally running MongoDB container

Once inside, you will see the prompt
test>

From there, you can do the following

To list all databases,
show dbs

To enter a database,
use <database_name>

and much more.

You can explore and inspect the database and its collections using various commands.