**API Server สำหรับสถานีวิทยุ**

**โครงสร้าง API Server**

**✅ Stack ที่ใช้**

* **Node.js** (Runtime)
* **Express.js** (Web Framework)
* **SQLite** (Database แบบ lightweight) — เปลี่ยนเป็น MySQL/PostgreSQL ได้ง่ายภายหลัง
* **Sequelize** (ORM สำหรับจัดการฐานข้อมูล)
* **📦 โครงสร้างโปรเจกต์**
* pgsql
* CopyEdit
* radio-api/
* ├── models/
* │ ├── station.js
* │ └── program.js
* ├── routes/
* │ ├── stations.js
* │ └── programs.js
* ├── app.js
* ├── database.js
* ├── package.json
* └── .env

**📄 ฐานข้อมูล (Schema)**

**Table: stations**

sql

CopyEdit

CREATE TABLE stations (

id INTEGER PRIMARY KEY AUTOINCREMENT,

code TEXT UNIQUE NOT NULL,

name TEXT NOT NULL,

description TEXT,

logo\_url TEXT,

stream\_url TEXT,

organization TEXT,

phone TEXT,

website TEXT,

facebook TEXT,

line TEXT,

province TEXT,

latitude REAL,

longitude REAL

);

**Table: programs**

sql

CopyEdit

CREATE TABLE programs (

id INTEGER PRIMARY KEY AUTOINCREMENT,

station\_id INTEGER NOT NULL,

code TEXT UNIQUE NOT NULL,

name TEXT NOT NULL,

description TEXT,

day\_of\_week TEXT NOT NULL,

start\_time TEXT NOT NULL, -- HH:MM format

end\_time TEXT NOT NULL, -- HH:MM format

cover\_url TEXT,

host TEXT,

FOREIGN KEY (station\_id) REFERENCES stations(id)

);

**📡 API Endpoints**

**GET /stations**

* ดึงรายชื่อสถานีวิทยุทั้งหมด

**GET /stations/:id**

* ดึงข้อมูลสถานีวิทยุตาม ID

**POST /stations**

* เพิ่มสถานีใหม่

**PUT /stations/:id**

* แก้ไขข้อมูลสถานี

**DELETE /stations/:id**

* ลบสถานี

**GET /programs**

* ดึงรายการทั้งหมด (option: filter by station, day)

**GET /stations/:id/programs**

* ดึงรายการทั้งหมดของสถานีที่เลือก

**POST /programs**

* เพิ่มรายการใหม่

**PUT /programs/:id**

* แก้ไขรายการ

**DELETE /programs/:id**

* ลบรายการ

**🚀 ตัวอย่าง Express Server (app.js)**

js

CopyEdit

const express = require('express');

const app = express();

const stationsRoutes = require('./routes/stations');

const programsRoutes = require('./routes/programs');

app.use(express.json());

app.use('/stations', stationsRoutes);

app.use('/programs', programsRoutes);

const PORT = process.env.PORT || 3000;

app.listen(PORT, () => {

console.log(`Radio API server running on port ${PORT}`);

});

**✅ ตัวอย่างการใช้งาน**

bash

CopyEdit

GET /stations

POST /stations

GET /stations/1/programs

POST /programs

**✅ STEP 1: อัปเดตระบบและติดตั้งเครื่องมือพื้นฐาน**

bash

CopyEdit

apt update && apt upgrade -y

apt install curl wget git build-essential -y

**✅ STEP 2: ติดตั้ง Node.js (แนะนำ Node.js LTS)**

bash

CopyEdit

curl -fsSL https://deb.nodesource.com/setup\_lts.x | bash -

apt install -y nodejs

ตรวจสอบว่า Node.js ติดตั้งเรียบร้อย:

bash

CopyEdit

node -v

npm -v

**✅ STEP 3: สร้างโฟลเดอร์โปรเจกต์ API**

bash

CopyEdit

mkdir ~/radio-api

cd ~/radio-api

npm init -y

**✅ STEP 4: ติดตั้ง Express และ SQLite3 + Sequelize**

bash

CopyEdit

npm install express sqlite3 sequelize

**✅ STEP 5: สร้างโครงสร้างโปรเจกต์**

bash

CopyEdit

mkdir models routes

touch app.js database.js

**✅ STEP 6: ตั้งค่า Sequelize และ SQLite (ไฟล์: database.js)**

js

CopyEdit

// database.js

const { Sequelize } = require('sequelize');

const sequelize = new Sequelize({

dialect: 'sqlite',

storage: 'radio.db', // ไฟล์ DB จะอยู่ใน root ของโปรเจกต์

});

module.exports = sequelize;

**✅ STEP 7: สร้าง Models**

**🔹 models/station.js**

js

CopyEdit

const { DataTypes } = require('sequelize');

const sequelize = require('../database');

const Station = sequelize.define('Station', {

code: { type: DataTypes.STRING, unique: true, allowNull: false },

name: { type: DataTypes.STRING, allowNull: false },

description: DataTypes.TEXT,

logo\_url: DataTypes.STRING,

stream\_url: DataTypes.STRING,

organization: DataTypes.STRING,

phone: DataTypes.STRING,

website: DataTypes.STRING,

facebook: DataTypes.STRING,

line: DataTypes.STRING,

province: DataTypes.STRING,

latitude: DataTypes.FLOAT,

longitude: DataTypes.FLOAT,

});

module.exports = Station;

**🔹 models/program.js**

js

CopyEdit

const { DataTypes } = require('sequelize');

const sequelize = require('../database');

const Station = require('./station');

const Program = sequelize.define('Program', {

code: { type: DataTypes.STRING, unique: true, allowNull: false },

name: { type: DataTypes.STRING, allowNull: false },

description: DataTypes.TEXT,

day\_of\_week: DataTypes.STRING,

start\_time: DataTypes.STRING,

end\_time: DataTypes.STRING,

cover\_url: DataTypes.STRING,

host: DataTypes.STRING,

});

Program.belongsTo(Station, { foreignKey: 'stationId' });

Station.hasMany(Program, { foreignKey: 'stationId' });

module.exports = Program;

**✅ STEP 8: สร้าง Express Server (ไฟล์: app.js)**

js

CopyEdit

const express = require('express');

const sequelize = require('./database');

const Station = require('./models/station');

const Program = require('./models/program');

const app = express();

app.use(express.json());

// Route ตัวอย่าง

app.get('/', (req, res) => res.send('Radio API Running'));

// TODO: import routes/stations.js, routes/programs.js

// Sync DB

sequelize.sync().then(() => {

console.log('Database synced');

const PORT = process.env.PORT || 3000;

app.listen(PORT, () => {

console.log(`API Server running on port ${PORT}`);

});

});

**✅ STEP 9: รัน API Server**

bash

CopyEdit

node app.js

เปิดในเบราว์เซอร์หรือ Postman:

arduino

CopyEdit

http://your-server-ip:3000/

**✅ STEP 10: ติดตั้ง PM2 (ถ้าต้องการให้รัน background)**

bash

CopyEdit

npm install -g pm2

pm2 start app.js --name radio-api

pm2 save

pm2 startup # ตั้งให้รันอัตโนมัติหลัง reboot

**✅ 1. แก้ไขใน app.js**

ค้นหาบรรทัดนี้ในไฟล์ app.js:

js

CopyEdit

const PORT = process.env.PORT || 3000;

เปลี่ยนเป็น:

js

CopyEdit

const PORT = process.env.PORT || 9005;

หรือจะกำหนดแบบ hard-coded ไปเลย:

js

CopyEdit

const PORT = 9005;

**✅ 2. ถ้าคุณใช้ pm2 ให้รีสตาร์ทด้วย:**

bash

CopyEdit

pm2 restart radio-api

หากยังไม่เคยใช้ชื่อ radio-api:

bash

CopyEdit

pm2 start app.js --name radio-api

**✅ 3. ตรวจสอบว่าเซิร์ฟเวอร์ฟังที่พอร์ตใหม่:**

bash

CopyEdit

ss -tuln | grep 9005

หากไม่แสดง แปลว่ามีบางอย่างผิดพลาดในการรัน

**✅ 4. เปิด firewall (ถ้าใช้ UFW หรือ iptables)**

ถ้าคุณเปิด firewall (เช่น UFW):

bash

CopyEdit

ufw allow 9005/tcp

**✅ 5. เข้าใช้งานผ่านเบราว์เซอร์หรือ Postman:**

arduino

CopyEdit

http://your-server-ip:9005/

**🔁 สรุปสั้น ๆ:**

| **คำว่า** | **หมายถึง** |
| --- | --- |
| **Route** | URL path เช่น /stations, /programs |
| **Router** | กลุ่มของ route ที่อยู่ในไฟล์เดียวกัน เช่น routes/stations.js |

**🔍 ตัวอย่าง Route:**

เช่น เมื่อผู้ใช้เรียก:

bash

CopyEdit

GET http://your-server-ip:9005/stations

เซิร์ฟเวอร์จะต้องรู้ว่าเมื่อมีคน GET เข้ามาที่ /stations ต้อง **ดึงรายชื่อสถานี** จากฐานข้อมูลไปแสดง

**✨ ตัวอย่างง่าย ๆ ของ Route**

**ไฟล์: routes/stations.js**

js

CopyEdit

const express = require('express');

const router = express.Router();

const Station = require('../models/station');

// GET /stations

router.get('/', async (req, res) => {

const stations = await Station.findAll();

res.json(stations);

});

// POST /stations

router.post('/', async (req, res) => {

const station = await Station.create(req.body);

res.status(201).json(station);

});

module.exports = router;

**✨ แล้วจะใช้ยังไง?**

ใน app.js คุณต้องเชื่อม Route นี้เข้า Express app ด้วย:

js

CopyEdit

const stationRoutes = require('./routes/stations');

app.use('/stations', stationRoutes);

จากนั้น API จะทำงานตามนี้:

| **Method** | **URL** | **ทำอะไร** |
| --- | --- | --- |
| GET | /stations | แสดงสถานีทั้งหมด |
| POST | /stations | เพิ่มสถานีใหม่ |

ต่อไปนี้คือไฟล์ **routes/stations.js** และ **routes/programs.js** ที่คุณสามารถใช้ได้เลยกับระบบที่คุณสร้างไว้

**✅ 1. routes/stations.js**

js

CopyEdit

const express = require('express');

const router = express.Router();

const Station = require('../models/station');

// GET /stations - รายชื่อสถานีทั้งหมด

router.get('/', async (req, res) => {

const stations = await Station.findAll();

res.json(stations);

});

// GET /stations/:id - รายละเอียดสถานีเดียว

router.get('/:id', async (req, res) => {

const station = await Station.findByPk(req.params.id);

if (!station) return res.status(404).json({ error: 'Station not found' });

res.json(station);

});

// POST /stations - เพิ่มสถานีใหม่

router.post('/', async (req, res) => {

try {

const station = await Station.create(req.body);

res.status(201).json(station);

} catch (err) {

res.status(400).json({ error: err.message });

}

});

// PUT /stations/:id - แก้ไขสถานี

router.put('/:id', async (req, res) => {

const station = await Station.findByPk(req.params.id);

if (!station) return res.status(404).json({ error: 'Station not found' });

await station.update(req.body);

res.json(station);

});

// DELETE /stations/:id - ลบสถานี

router.delete('/:id', async (req, res) => {

const station = await Station.findByPk(req.params.id);

if (!station) return res.status(404).json({ error: 'Station not found' });

await station.destroy();

res.json({ message: 'Station deleted' });

});

module.exports = router;

**✅ 2. routes/programs.js**

js

CopyEdit

const express = require('express');

const router = express.Router();

const Program = require('../models/program');

const Station = require('../models/station');

// GET /programs - รายการทั้งหมด (option: ?stationId=1)

router.get('/', async (req, res) => {

const where = {};

if (req.query.stationId) where.stationId = req.query.stationId;

const programs = await Program.findAll({ where, include: Station });

res.json(programs);

});

// GET /programs/:id - รายละเอียดรายการ

router.get('/:id', async (req, res) => {

const program = await Program.findByPk(req.params.id, { include: Station });

if (!program) return res.status(404).json({ error: 'Program not found' });

res.json(program);

});

// POST /programs - เพิ่มรายการ

router.post('/', async (req, res) => {

try {

const program = await Program.create(req.body);

res.status(201).json(program);

} catch (err) {

res.status(400).json({ error: err.message });

}

});

// PUT /programs/:id - แก้ไขรายการ

router.put('/:id', async (req, res) => {

const program = await Program.findByPk(req.params.id);

if (!program) return res.status(404).json({ error: 'Program not found' });

await program.update(req.body);

res.json(program);

});

// DELETE /programs/:id - ลบรายการ

router.delete('/:id', async (req, res) => {

const program = await Program.findByPk(req.params.id);

if (!program) return res.status(404).json({ error: 'Program not found' });

await program.destroy();

res.json({ message: 'Program deleted' });

});

module.exports = router;

**✅ 3. แก้ app.js เพื่อเชื่อม Route**

เปิดไฟล์ app.js แล้วเพิ่มด้านล่างนี้:

js

CopyEdit

const stationRoutes = require('./routes/stations');

const programRoutes = require('./routes/programs');

app.use('/stations', stationRoutes);

app.use('/programs', programRoutes);

**🚀 ทดลอง API**

* GET /stations
* POST /stations
* GET /programs?stationId=1
* POST /programs

**✅ POST /stations - เพิ่มสถานีวิทยุใหม่**

**🎯 ตัวอย่าง JSON:**

json

CopyEdit

{

"code": "MCOT-BKK",

"name": "MCOT Radio Bangkok",

"description": "สถานีวิทยุกระจายเสียง MCOT กรุงเทพฯ",

"logo\_url": "https://example.com/logo.png",

"stream\_url": "https://stream.mcot.net/live",

"organization": "MCOT Public Company Limited",

"phone": "02-123-4567",

"website": "https://radio.mcot.net",

"facebook": "https://facebook.com/mcotradio",

"line": "@mcotradio",

"province": "กรุงเทพมหานคร",

"latitude": 13.7563,

"longitude": 100.5018

}

**✅ POST /programs - เพิ่มรายการวิทยุใหม่**

**หมายเหตุ:** ต้องมี stationId (id ของสถานี) ที่ได้จาก /stations มาก่อน เช่น 1

**🎯 ตัวอย่าง JSON:**

json

CopyEdit

{

"code": "MORNING-TALK",

"name": "Morning Talk",

"description": "รายการข่าวสารยามเช้า สดจาก MCOT",

"day\_of\_week": "จันทร์-ศุกร์",

"start\_time": "06:00",

"end\_time": "08:00",

"cover\_url": "https://example.com/program-cover.jpg",

"host": "สมชาย ดีเจ",

"stationId": 1

}

**🧪 วิธีทดสอบใน Postman:**

1. เลือก **POST**
2. ใส่ URL เช่น: http://your-server-ip:9005/stations
3. ไปที่แท็บ **Body > raw > JSON**
4. วาง JSON ข้างต้นลงไป แล้วกด Send

**✅ 1. ตรวจสอบว่า API Server ของคุณทำงานอยู่**

บนเครื่อง server ให้คุณสั่ง:

bash

CopyEdit

pm2 status

หรือถ้าคุณยังใช้ node app.js อยู่ให้เปิด session terminal ไว้ แล้วดูว่าแสดงว่า API Server running on port 9005 หรือไม่

**✅ 2. หา IP ของ Server**

บนเครื่อง server:

bash

CopyEdit

ip a

หรือใช้:

bash

CopyEdit

hostname -I

จะได้ IP ของเครื่อง เช่น 192.168.1.50 (ถ้าเป็น LAN) หรือ 203.xxx.xxx.xxx (ถ้าเป็น public IP)

**✅ 3. ตรวจสอบว่า port 9005 เปิดให้เชื่อมจากภายนอก**

ให้สั่งบน server:

bash

CopyEdit

ss -tuln | grep 9005

และถ้าใช้ UFW:

bash

CopyEdit

ufw allow 9005/tcp

**✅ 4. บนเครื่องที่ใช้ Postman (เครื่อง local)**

1. เปิดโปรแกรม **Postman**
2. ตั้ง URL เป็น:

arduino

CopyEdit

http://<your-server-ip>:9005/stations

เช่น:

arduino

CopyEdit

http://192.168.1.50:9005/stations

1. เลือก Method: GET, POST, PUT, หรือ DELETE
2. ถ้าเป็น POST หรือ PUT:
   * ไปที่แท็บ Body
   * เลือก raw และชนิด JSON
   * ใส่ JSON ตัวอย่างที่ผมให้ไว้ก่อนหน้านี้
3. กดปุ่ม **Send**

**✅ ถ้าเชื่อมไม่ได้**

* ตรวจสอบว่า server อยู่ใน network เดียวกับเครื่องที่ใช้ Postman หรือเปิด port แล้ว
* หากยังไม่สำเร็จ ให้ลองพิมพ์ URL จากเบราว์เซอร์ก่อน:

cpp

CopyEdit

http://<your-server-ip>:9005/

ถ้าได้ "Radio API Running" แปลว่า server ทำงานและเชื่อมได้แน่นอน