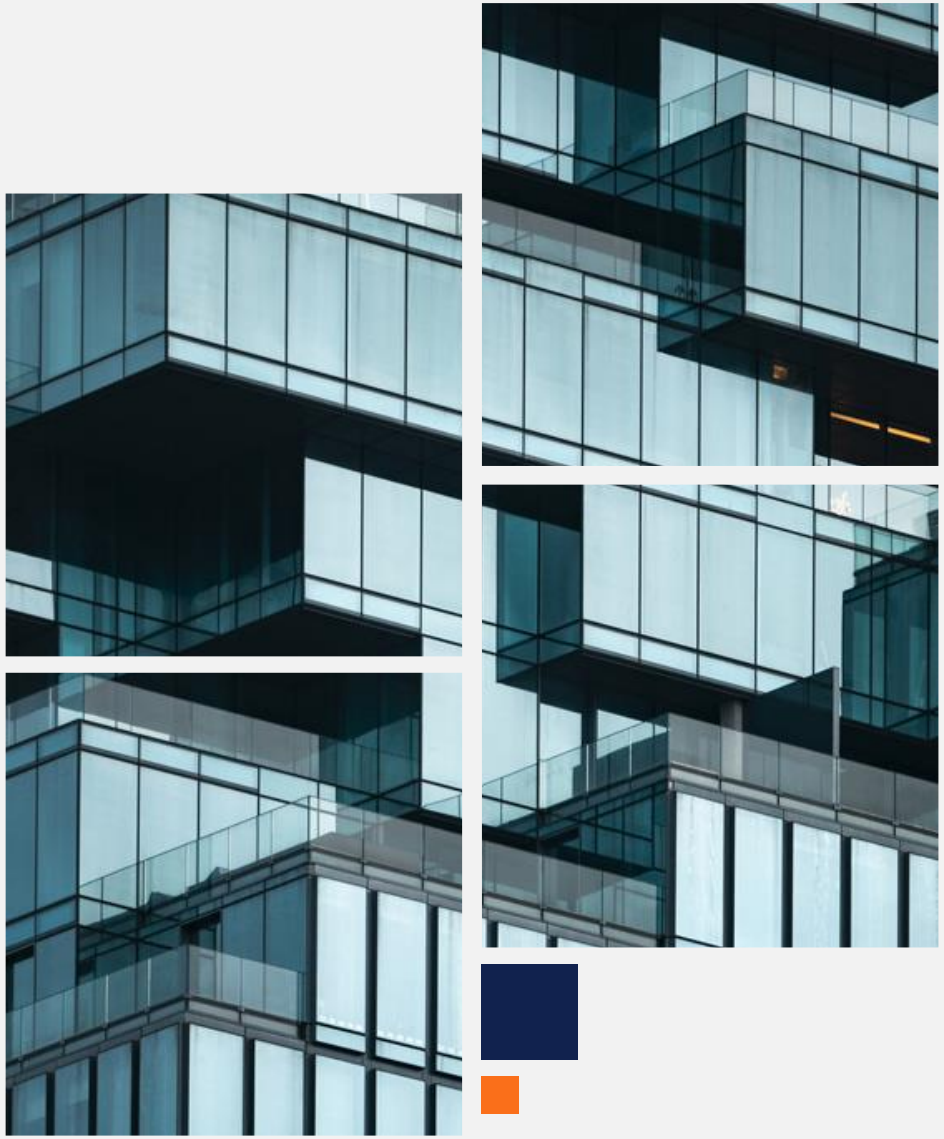
RESTful API with Express.js and SQLite

March 21, 2024

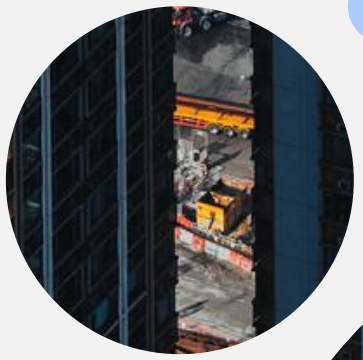
Antti Venetjoki

Introduction

This project involves building a RESTful API using Express.js and SQLite. The API allows users to query a database containing information about artists and albums from the Chinook music store database.



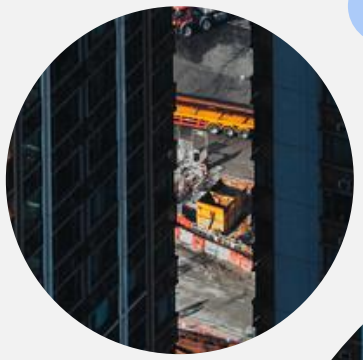
Technologies used



- Express.js
- SQLite
- dotenv

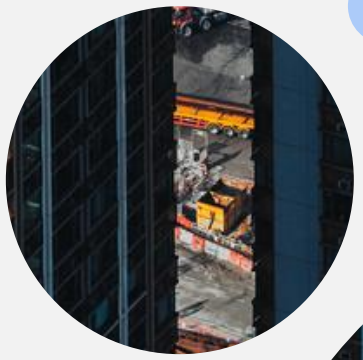
Express.js

- Minimalist Web Framework
  - Express.js is a minimalist web framework for Node.js, providing essential features for building web applications and APIs
- Middleware-driven Architecture
  - Its middleware-based approach allows seamless integration of functionalities like request processing, authentication, and error handling
- Efficient Routing
  - Express simplifies routing, making it easy to define routes for different URL paths and HTTP methods, enhancing the organization and readability



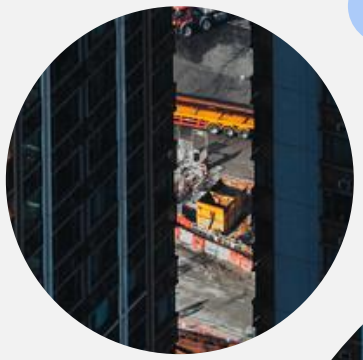
SQLite

- Embedded Database
  - SQLite is a lightweight, file-based SQL database engine designed for embedded systems and small to medium-sized applications
- Serverless Architecture
  - Unlike traditional client-server databases, SQLite operates in a serverless manner, with the database engine being directly integrated into the application.
- Self-contained and Zero Configuration
  - SQLite databases are self-contained, requiring no setup or administration. They can be easily transferred between different systems, making deployment and management hassle-free.



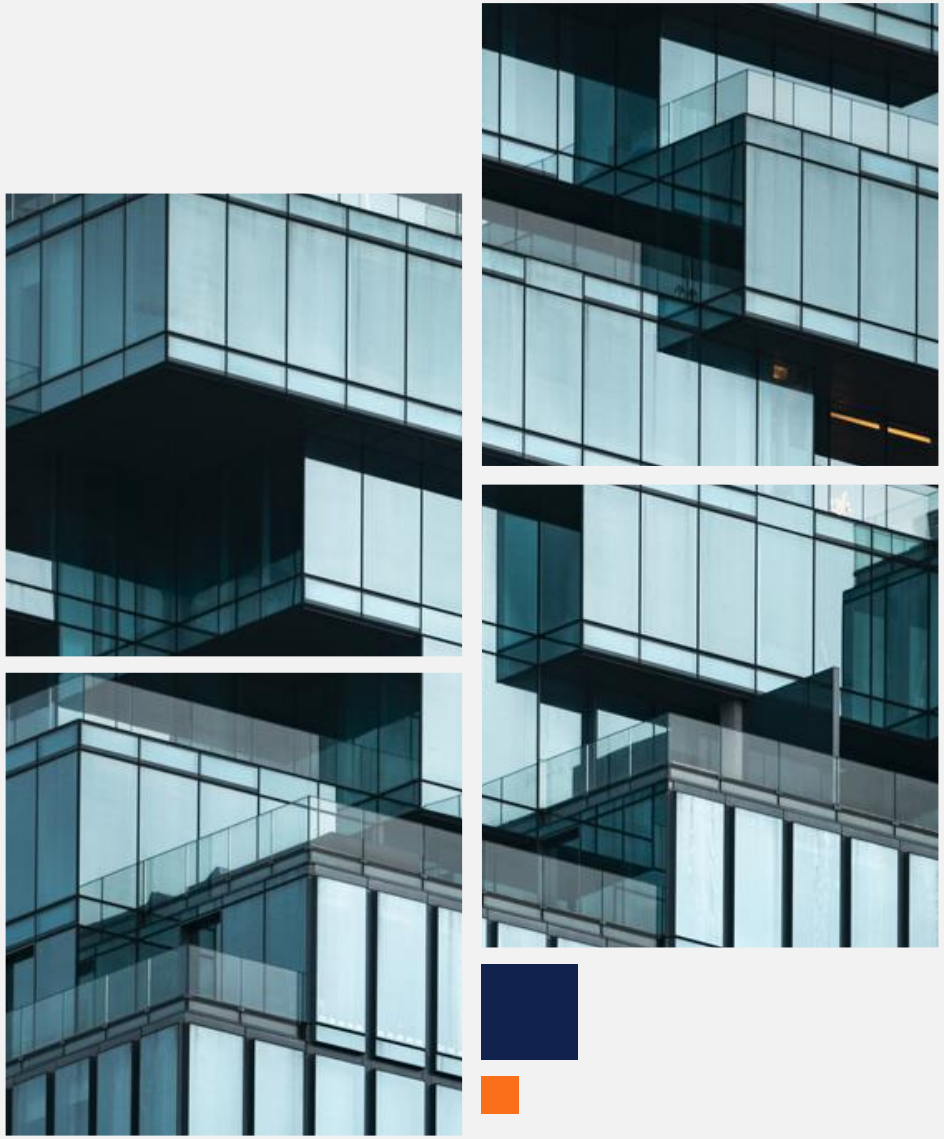
dotenv

- Environment Variable Management
  - dotenv is a Node.js module that facilitates the management of environment variables within applications
- Configuration Centralization
  - It allows developers to centralize configuration settings in a .env file, enhancing organization and simplifying deployment processes
- Cross-Platform Compatibility
  - dotenv ensures consistent behavior across different platforms, providing flexibility and ease of use in development environments

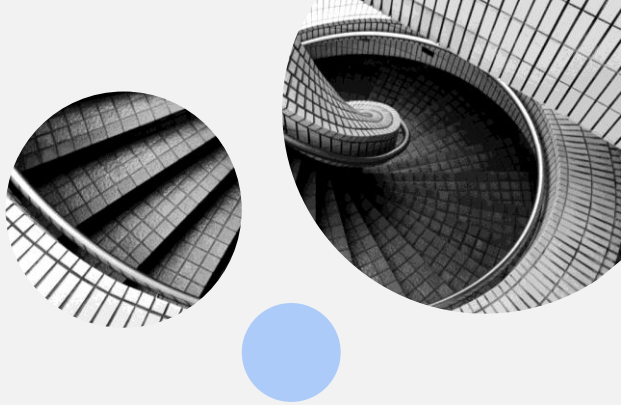


Code overview

Next 4 slides will give a brief overview of the code and the endpoints it creates



General



```
const express = require('express')
const dotenv = require('dotenv')
const sqlite3 = require('sqlite3').verbose()

dotenv.config()

const app = express()
const port = process.env.PORT || 8000
const url = process.env.URL || 'http://localhost'
const database = process.env.DATABASE || 'chinook'

const db = new sqlite3.Database(
  `${database}.db`, sqlite3.OPEN_READWRITE, (err) =>
{
  if (err) {
    console.error(err.message)
  }
  console.log(`Connected to the ${database} database.`)
})
```

- Getting necessary packages
- Getting environment variables
- Connecting to the database

```
app.listen(port, () => {
  console.log(`Server is running on ${url}:${port}`)
})
```

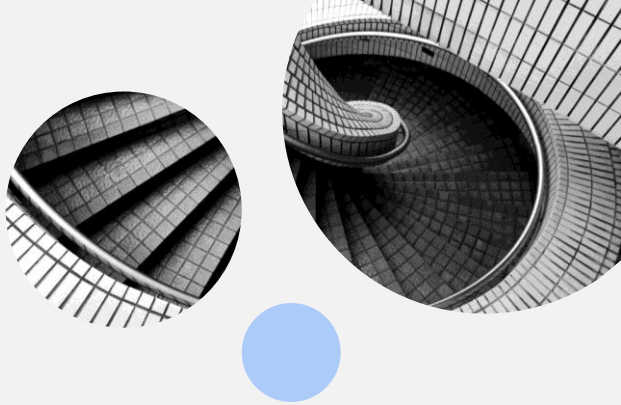
- Listening for requests

```
app.get('/api/v1/artist', (req, res) => {
  const id = req.query.id
  const name = req.query.name
  let sql = 'SELECT * FROM artists'

  const params = []

  if (id && name) {
    sql += ' WHERE UPPER(Name) = UPPER(?) AND ArtistId = ?'
    params.push(name, id)
  } else if (name) {
    sql += ' WHERE UPPER(Name) = UPPER(?)'
    params.push(name)
  } else if (id) {
    sql += ' WHERE ArtistId = ?'
    params.push(id)
  }

  db.all(sql, params, (err, rows) => {
    if (err) {
      res.status(500).json({ error: err.message })
      return
    }
    if (rows.length === 0) {
      res.status(404).json({ message: 'Artist not found' })
      return
    }
    res.json(rows)
  })
})
```

/api/v1/artist

- Creates endpoint for listing artists
- Allows searching with id or name or both
- Search by name is case insensitive
- Returns all artists if no parameters are given

/api/v1/album



```
app.get('/api/v1/album', (req, res) => {
  const albumId = req.query.albumId
  const artistId = req.query.artistId
  const albumName = req.query.albumName
  const artistName = req.query.artistName

  let sql = 'SELECT '
  sql += 'al.Title AS \'Album Name\', '
  sql += 'al.albumId AS \'Album ID\', '
  sql += 'ar.Name AS \'Artist Name\', '
  sql += 'ar.ArtistId AS \'Artist ID\' '
  sql += 'FROM albums AS al '
  sql += 'JOIN artists AS ar ON al.ArtistId = ar.ArtistId'

  const params = []
```

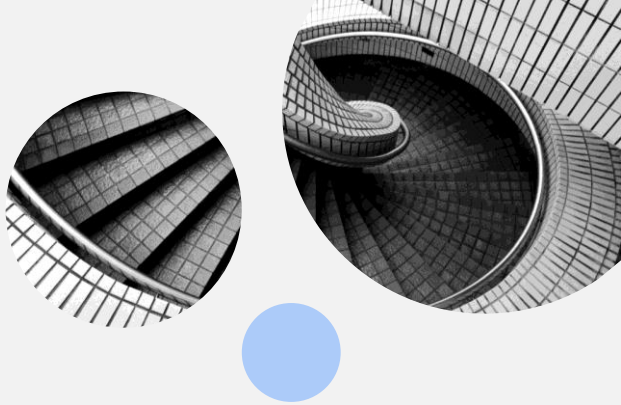
- Creates endpoint for listing albums
- Allows searching by album and by artist
- Searches by name is case insensitive
- Returns all albums if no parameters are give

```
if (artistId || artistName || albumId || albumName) {
  sql += ' WHERE'
}
if (artistId) {
  sql += ' al.artistId = ?'
  params.push(artistId)
}
if (artistName) {
  if (params.length > 0) {
    sql += ' AND'
  }
  sql += ' UPPER(ar.Name) = UPPER(?)'
  params.push(artistName)
}
if (albumName) {
  if (params.length > 0) {
    sql += ' AND'
  }
  sql += ' UPPER(al.Title) = UPPER(?)'
  params.push(albumName)
}
if (albumId) {
  if (params.length > 0) {
    sql += ' AND'
  }
  sql += ' al.albumId = ?'
  params.push(albumId)
}
```

/api/v1/album

- WHERE statements is being created
- All necessary conditions are added to the SQL query

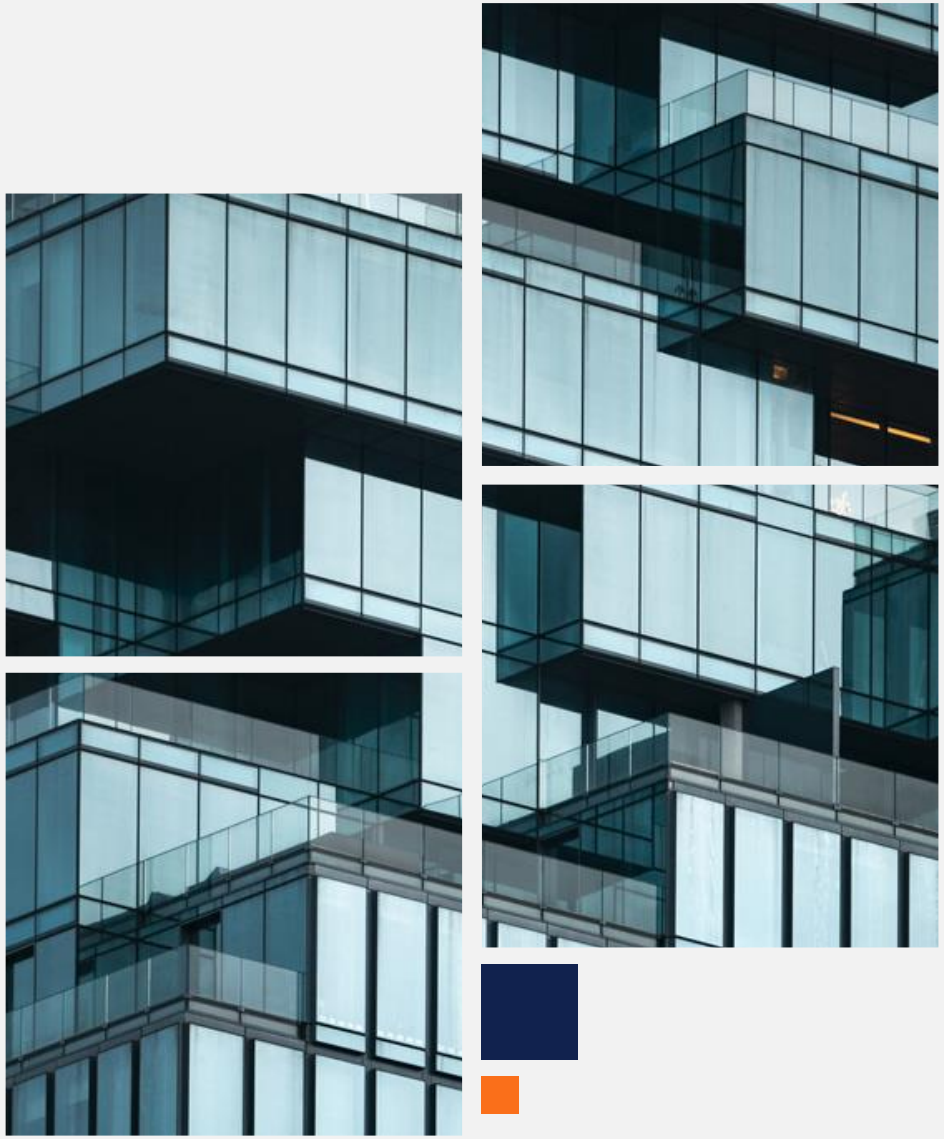
/api/v1/album



```
db.all(sql, params, (err, rows) => {
  if (err) {
    res.status(500).json({ error: err.message })
    return
  }
  if (rows.length === 0) {
    res.status(404).json({ message: 'Album not found' })
  }
  res.json(rows)
})
```

- The database is searched with SQL query from previous slides
- Appropriate HTTP response is sent

Testing the API



localhost:8000/api/v1/artist?id=123

JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON

```
0:
    ArtistId:   123
    Name:       "R.E.M. Feat. KRS-One"
```

localhost:8000/api/v1/artist?name=accept

JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON

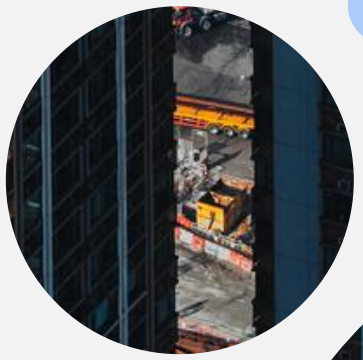
```
0:
    ArtistId:   2
    Name:       "Accept"
```

localhost:8000/api/v1/artist?name=Accept&id=2

JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON

```
0:
    ArtistId:   2
    Name:       "Accept"
```



localhost:8000/api/v1/album?albumId=3

JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON

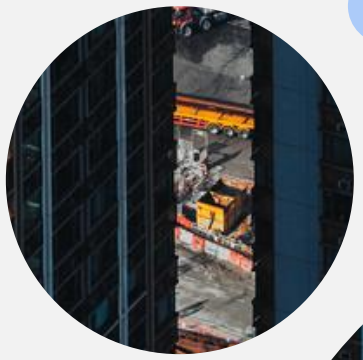

```
▼ 0:
    Album Name:    "Restless and Wild"
    Album ID:      3
    Artist Name:   "Accept"
    Artist ID:     2
```


localhost:8000/api/v1/album?albumName=unplugged

JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON


```
▼ 0:
    Album Name:    "Unplugged"
    Album ID:      73
    Artist Name:   "Eric Clapton"
    Artist ID:     81
```

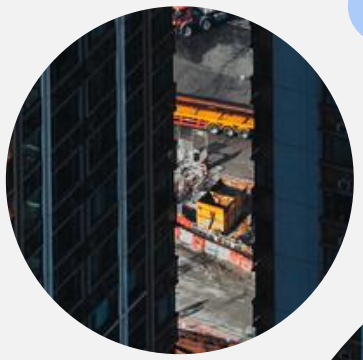



localhost:8000/api/v1/album?artistName=Accept

JSON | Raw Data | Headers

Save | Copy | Collapse All | Expand All | Filter JSON


```
0:
    Album Name:     "Balls to the Wall"
    Album ID:       2
    Artist Name:    "Accept"
    Artist ID:      2
1:
    Album Name:     "Restless and Wild"
    Album ID:       3
    Artist Name:    "Accept"
    Artist ID:      2
```

localhost:8000/api/v1/album?artistId=10

JSON | Raw Data | Headers

Save | Copy | Collapse All | Expand All | Filter JSON


```
0:
    Album Name:     "The Best Of Billy Cobham"
    Album ID:       13
    Artist Name:    "Billy Cobham"
    Artist ID:      10
```

Conclusion

This project was completed without major challenges

Biggest problem was to find suitable data that didn’t need much reformatting

No timetable was kept

A passing grade was targeted