2022

CAB230 REST API – Server Side

CAB230 Volcano API – The Server Side Application

Duy Pham (Daniel)

N10640754

6/7/2022

Contents

This template is adapted from one created for a more elaborate application. The original author spends most of his professional life talking to clients and producing architecture and services reports. You may find this a bit more elaborate than you are used to, but it is there to help you get a better mark

This report will probably be around 5 pages or so including screenshots

Introduction

Purpose & description

This Express Application was built to help with CAB230's Web Computing assignment number two. It exposes not only a set of REST endpoints that perform CRUD operations on a database of publicly available volcano data but also some extended end points that allow users to update their profile when login.

Completeness and Limitations

completeness:

- Successful implementation of ***all*** of the Data endpoints at a high level (passed all unit tests).
- Implemented all key security requirements:
  - Use of knex – there should be no raw SQ
  - Use of morgan with a logging level similar to that used in the practicals
  - Appropriate handling of user passwords as described in the JWT Server-Side worksheet – inappropriate storage and handling of passwords will be heavily penalised
  - TLS/HTTPS deployment with a self-signed certificate

/countries

fully functional

/volcanoes

fully functional

/volcano/{id}

fully functional

/user/register

fully functional

/user/login

fully functional

/user/{email}/profile

fully functional

/me

fully functional

Modules used

No additional modules used

Technical Description

Architecture

Here is the overall architecture of my Express application:

bin	6/7/2022 11:00 PM	File folder	
public	6/7/2022 11:00 PM	File folder	
routes	6/7/2022 11:00 PM	File folder	
views	6/7/2022 11:00 PM	File folder	
.env	5/31/2022 11:28 PM	ENV File	1 KB
app	6/7/2022 12:02 PM	JavaScript Source ...	3 KB
dump	5/25/2022 5:22 PM	SQL Text File	172 KB
knexfile	6/1/2022 11:10 AM	JavaScript Source ...	1 KB
package	6/2/2022 1:59 PM	JSON File	1 KB
package-lock	6/2/2022 2:06 PM	JSON File	131 KB
swagger	5/26/2022 1:59 PM	JSON File	30 KB

Some main files need to be noticed about are:

- routes: This folder include all router files of the REST api

  admin
  authentication
  data
  profile

  For each file it will contain all endpoints from appropriate section on swagger web

  Data

  Authentication

  Profile

  Administration

- app: This is the Heart of the application which use all routes provided from the folder above, swagger docs format and also knex file.

```
app.use("/", dataRouter);
app.use("/", profileRouter);
app.use("/", adminRouter);
app.use("/", authenticationRouter);
app.use("/", swaggerUi.serve);
app.get(
  "/",
  swaggerUi.setup(swaggerDoc, {
    swaggerOptions: { defaultModelsExpandDepth: -1 }, // Hide schema section
  })
);
//Capture All 404 errors
app.use(function (req,res,next){
  res.status(404).send('Unable to find the requested resource!');
});
```

```
app.use((req, res, next) => {
  req.db = knex;
  next();
});
```

- knexfile: Contains configuration to connect with database
- swagger: Swagger website format setup
- dump: The given volcano Database
- .env: Custom port to run REST API

Security

Application securities:

- Use of `knex` or other query builder without raw SQL: setup a knex form to connect with a .sql dump file

```
module.exports = {
  client: "mysql2",
  connection: {
    host: "localhost",
    port: 3306,
    database: "volcanoes",
    user: "root",
    password: "Cab230!",
  },
};
```

- Use of `morgan` with a logging level similar to that used in the pracs.

```
var logger = require("morgan");
var cors = require("cors");
```

```
logger.token("req", (req, res) => JSON.stringify(req.headers));
logger.token("res", (req, res) => {
  const headers = {};
  res.getHeaderNames().map((h) => (headers[h] = res.getHeader(h)))
  return JSON.stringify(headers);
});
```

- Appropriate handling of user passwords as described in the JWT Server-Side worksheet.
  Password Harsh before insert to database:

```
.insert([
  {
    email: req.body.email,
    password: bcrypt.hashSync(req.body.password, 10),
  },
])
```

- Deployment using TLS/HTTPS
  Edited in bin/www file of the application

```
var https = require('https');

/**
 * Get port from environment and store in Express.
 */

var port = normalizePort(process.env.PORT || "443");
app.set("port", port);

const fs = require("fs");
const privateKey = fs.readFileSync(
  "/etc/ssl/private/node-selfsigned.key",
  "utf8"
);
const certificate = fs.readFileSync(
  "/etc/ssl/certs/node-selfsigned.crt",
  "utf8"
);
const credentials = {
  key: privateKey,
  cert: certificate,
};

/**
 * Create HTTPS server.
 */

var server = https.createServer(credentials, app);
```

Reflection:

Strength:

- Prevent man-in-the-middle attacks, domain spoofing, and other methods attackers use to impersonate a website and trick users.
- Prevent data breaches (TLS)
- Preventing SQL Injection (Knex.js)
- A02:2021 – Cryptographic Failures:
    - MD5 as compute power and encryption standards by using Bcrypt
- A08:2021 – Software and Data integrity failure:
    - Use digital signatures or similar mechanisms to verify the software or data is from the expected source and has not been altered
    - Ensure libraries and dependencies, such as npm or Maven, are consuming trusted repositories. If you have a higher risk profile, consider hosting an internal known-good repository that's vetted.
    - Ensure that unsigned or unencrypted serialized data is not sent to untrusted clients without some form of integrity check or digital signature to detect tampering or replay of the serialized data
- A01:2021 – Broken Access control:
    - Except for public resources, deny by default.
    - Implement access control mechanisms once and re-use them throughout the application, including minimizing Cross-Origin Resource Sharing (CORS) usage

Weakness:

- MITM attacks can be exploited from the browser (huge risk)
- A02:2021 – Cryptographic Failures:
    - Don't store sensitive data unnecessarily. Discard it as soon as possible or use PCI DSS compliant tokenization or even truncation. Data that is not retained cannot be stolen.

Testing

Test Report

Started: 2022-06-07 23:55:43

```
Suites (1)
  1 passed
  0 failed
  0 pending
```

```
Tests (276)
  276 passed
  0 failed
  0 pending
```

Difficulties / Exclusions / unresolved & persistent errors /

- What were your major roadblocks / how did you resolve them?
  - Misunderstanding ".then()" chain: I have get the error of trying to re-setting header property after sent the header to client.
    The problem was solved by using a system of try…catch error with the use of throw error for each condition block instead of using "return"

```
//Profile endpoint
//get profile
router.get("/user/:email/profile", function (req, res, next) {
  const authorization = req.headers.authorization;
  let token = null;
  try {
    //if existed token
    if (authorization) {
      token = authorization.split(" ")[1];

      var bearer = authorization.split(" ")[0];
      console.log("Token: ", token);
      console.log("Bearer: ", bearer);
      if (
        authorization.split(" ").length < 2 ||
        authorization.split(" ").length > 2 ||
        bearer !== "Bearer"
      ) {
        console.log("Malformed Bearer");
        throw new Error("Malformed Bearer");
      }
      //verify token and response any error about JWT token
      const decoded = jwt.verify(token, secretKey);

      //if the token is expired
      if (decoded.exp < Date.now()) {
        console.log("expired");
        throw new Error("JWT token has expired");
      }
```

```
  } catch (e) {
    if (e.message === "jwt malformed") {
      res.status(401).json({
        error: true,
        message: "Invalid JWT token",
      });
    } else if (e.message === "JWT token has expired")
      res.status(401).json({
        error: true,
        message: "JWT token has expired",
      });
    else if (e.message === "Malformed Bearer") {
      res.status(401).json({
        error: true,
        message: "Authorization header is malformed",
      });
    }
  }
});
```

- Updating date of birth for user profile: I have use a complicate solution for updating valid DoB and after a while I have switched to use regular expression of JavaScript to find valid format and date.getTime() to find valid date (counting method)

```
//function to check invalid date with invalid format
//ref: https://stackoverflow.com/questions/18758772/how-do-
function checkValidDate(date) {
  var dateFormat = /^\d{4}-\d{2}-\d{2}$/;

  // Invalid format
  if (!date.match(dateFormat)) return false;
  var d = new Date(date);
  var dateCount = d.getTime();
  // NaN value, Invalid date
  if (!dateCount && dateCount !== 0) return false;

  return d.toISOString().slice(0, 10) === date;
}
```

- Swagger route skip 404 error handler: I fixed this problem by making some adjustments for swagger route.

```
app.get(
  "/",
  swaggerUi.setup(swaggerDoc, {
    swaggerOptions: { defaultModelsExpandDepth: -1 }, // Hide sche
  })
);
//Capture All 404 errors
app.use(function (req,res,next){
  res.status(404).send('Unable to find the requested resource!');
});
```

- Any functionality you didn't or couldn't finish and the technical issues encountered
  No functionality that I missed
- Are there any outstanding bugs?
  Not really

Installation guide

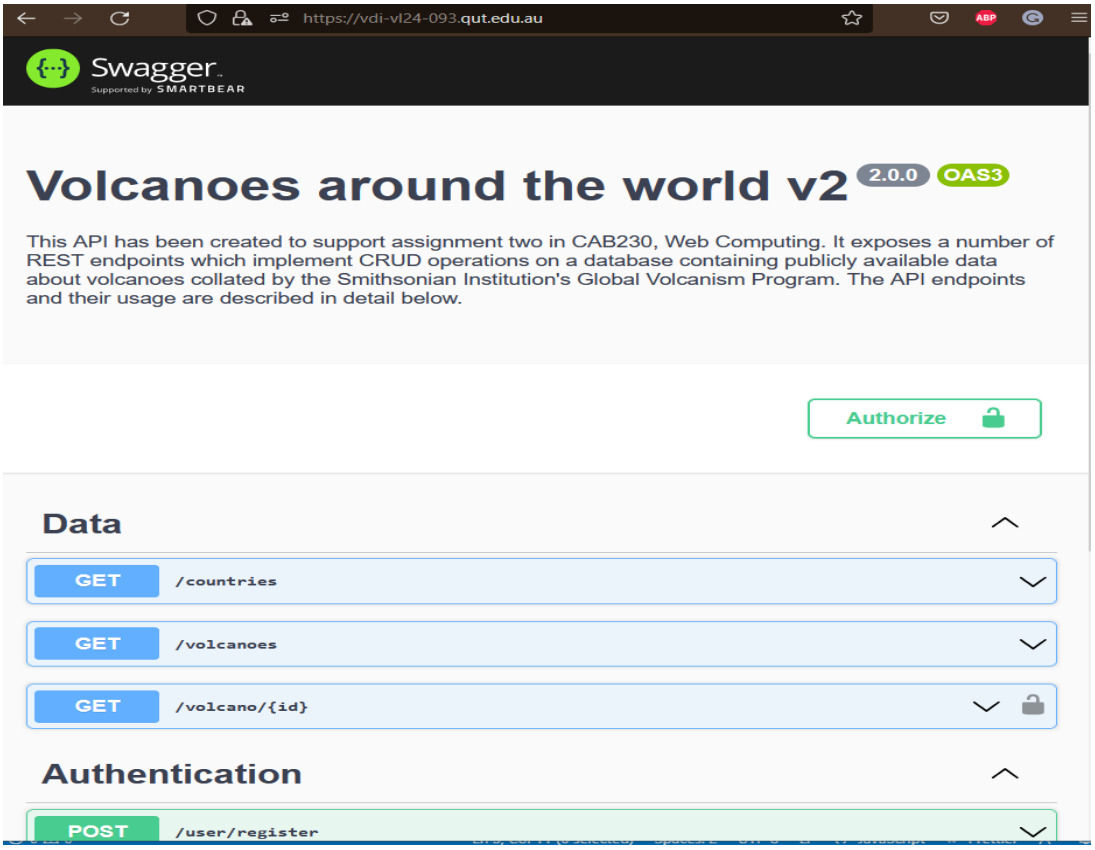
1. Download folder then unzip
2. Navigate to expvolcano file (assign\volcanoweb_server_side\expvolcano)
3. Open terminal (VSCode or anything else) navigate the path then run
   npm install

```
PS D:\QUT Student\sem1\CAB230\Assignment 2\assign\volcanoweb_server_side\expvo
lcano> npm install
```

4. When Successfully install node_modules we can then run
   npm run start

```
PS D:\QUT Student\sem1\CAB230\Assignment 2\assign\volcanoweb_server_side\expvo
lcano> npm run start
```

5. You can now see the the Rest API is running on your
   https://localhost:443/ or https://vdi-vl24-093.qut.edu.au/ (Make sure to connect with QUT internet)



References

Use a standard approach to referencing – see the guidance at https://www.citewrite.qut.edu.au/cite/.

These references are not for the node modules but rather refer to any blogs or tutorials or whatever else you have used in the introduction or the Technical Description of the application.

Appendices as you require them

Anything you think should be included but is better left to the end.