

# Coaching projet 1 MDV

☒ APIs de Gestion des Anniversaires et  
des Citations & Base de données

Hyejeong IM

CITATION ET ANNIVERSAIRES

LUNDI 29 JUILLET 2024 | 20:45:20

✦

✦


✦

JOYEUX ANNIVERSAIRE

HYEJEONG IM

IL N'Y A PAS DE SUBSTITUT AU TRAVAIL ACHARNÉ.

— THOMAS EDISON



0103

---

## Objectif:

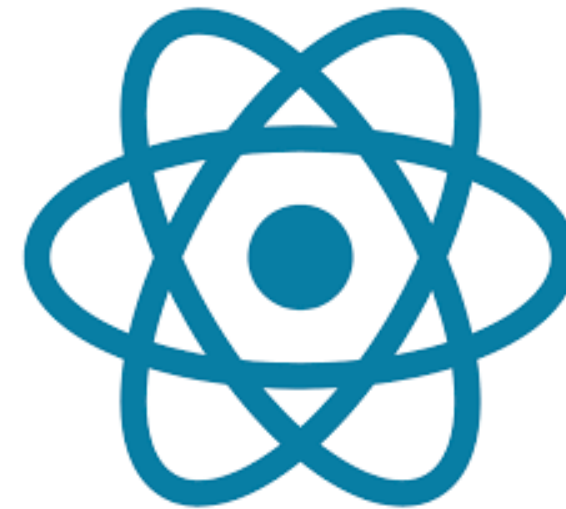
Automatiser l'importation et la gestion des anniversaires et des citations à partir de fichiers CSV.

## Stack technique:

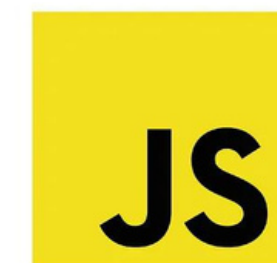
Frontend: React

Backend: Express.js

Base de Données: MySQL



Express



---

# Importation des Fichiers CSV dans la Base de Données

## Table students

```
mysql> describe students;
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
birthday	date	YES		NULL	
lastname	varchar(50)	YES		NULL	
firstname	varchar(50)	YES		NULL	
email	varchar(100)	YES	UNI	NULL	

## Table intervenants

```
mysql> describe intervenants;
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
birthday	date	YES		NULL	
lastname	varchar(50)	YES		NULL	
firstname	varchar(50)	YES		NULL	
email	varchar(100)	YES		NULL	

## Table quotes

```
mysql> describe quotes;
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
quote	text	YES		NULL	
author	varchar(100)	YES		NULL	

## db.js

```
1  const mysql = require("mysql2/promise");
2  require("dotenv").config();
3
4  const pool = mysql.createPool({
5    host: process.env.DB_HOST,
6    user: process.env.DB_USER,
7    password: process.env.DB_PASSWORD,
8    database: process.env.DB_NAME,
9    port: process.env.DB_PORT,
10   waitForConnections: true,
11   queueLimit: 0,
12 });
13
14 module.exports = pool;
```

```
✓ happy-birthday-back
  ✓ src
    > api
      ✓ config
        JS db.js
```

# Implémentation Backend pour l'Importation de CSV

## Points d'API:

/api/controllers/importController ➔ importStudents  
importQuotes

1. Traitement de l'upload de fichier: Utilisation de **multer** pour traiter l'upload des fichiers.
2. Génération du chemin de fichier: Utilisation du module **path** pour générer le chemin absolu du fichier.
3. Lecture et parsing du fichier: Utilisation de **fs** pour lire le fichier et de csv-parse pour le parser.
4. Insertion des données dans la base de données: Utilisation de **pool.query** pour insérer les données dans la base de données.
5. Suppression du fichier: Utilisation de **fs** pour supprimer le fichier du serveur.
6. Gestion des erreurs: Traitement des erreurs lors du processus.

```
const upload = multer({ dest: "uploads/" }); 1

exports.importQuotes = [
  upload.single("file"),
  (req, res) => {
    const filePath = path.resolve( 2
      __dirname,
      "../../uploads",
      req.file.filename
    );
    const parser = fs 3
      .createReadStream(filePath)
      .pipe(parse({ delimiter: ";", columns: true, bom: true }));

    parser.on("data", async (row) => {
      try {
        await pool.query("INSERT INTO quotes (quote, author) VALUES (?, ?)", [ 4
          row.quote,
          row.author,
        ]);
      } catch (err) {
        console.error("Error inserting quote:", err);
      }
    });

    parser.on("end", () => { 5
      fs.unlinkSync(filePath);
      res.send("Quotes imported successfully");
    });

    parser.on("error", (error) => { 6
      res.status(500).send("Error processing file: " + error.message);
    });
  },
];
```

---

# Intégration de l'API

## Points d'API:

/api/controllers/birthdayController ➡ getToday'sBirthday  
/api/controllers/quoteController ➡ getRandomQuote

## Requêtes SQL utilisées:

### getToday'sBirthday

```
SELECT id, birthday, lastname, firstname, email  
FROM students  
WHERE DATE_FORMAT(birthday, "%m-%d") = DATE_FORMAT(NOW(), "%m-%d")
```

### getRandomQuote

```
SELECT * FROM quotes ORDER BY RAND() LIMIT 1
```

```
const pool = require("../config/db");  
const { importQuotes } = require("../importController");  
  
exports.getRandomQuote = async (req, res) => {  
  try {  
    const [rows] = await pool.query(  
      "SELECT * FROM quotes ORDER BY RAND() LIMIT 1"  
    );  
    res.json(rows[0]);  
  } catch (err) {  
    console.error("Error reading from database:", err);  
    res.status(500).json({ error: "Database error" });  
  }  
};  
  
exports.uploadAndImportQuotes = importQuotes;
```

---

# Configuration des Routes

## Définit les points de terminaison API

POST /import - Point de terminaison pour importer des citations.

GET /randomquote - Point de terminaison pour obtenir une citation aléatoire.

```
const express = require("express");
const quoteController = require("../controllers/quoteController");

const router = express.Router();

router.post("/import", quoteController.uploadAndImportQuotes);
router.get("/randomquote", quoteController.getRandomQuote);

module.exports = router;
```

```
▼ happy-birthday-back
  ▼ src
    ▼ api
      > controllers
      ▼ routes
        JS birthdayRoute.js
        JS importRoutes.js
        JS quoteRoute.js
```

---

# Couche de Service - Analyse des CSV

## Gère l'analyse des fichiers CSV

Utilise **csv-parse** pour lire et analyser les données CSV.  
Retourne les données analysées sous forme de **Promise**.

```
const fs = require("fs");
const path = require("path");
const { parse } = require("csv-parse");

exports.parseFile = (filename) => {
  let results = [];
  const filePath = path.resolve(__dirname, "../data", filename);

  return new Promise((resolve, reject) => {
    fs.createReadStream(filePath)
      .pipe(
        parse({
          delimiter: ";",
          columns: true,
          bom: true,
        })
      )
      .on("data", function (row) {
        results.push(row);
      })
      .on("end", function () {
        console.log(`Parsed data from ${filePath}:`, results);
        resolve(results);
      })
      .on("error", function (error) {
        reject(error.message);
      });
  });
};
```



---

# Configuration de l'Application(Back)

## Configure le serveur Express et importe les données

Configure les middleware et les routes.

Définit des fonctions pour la conversion de format de date et l'importation de données CSV.

Initialise la base de données avec les données CSV avant de démarrer le serveur.

```
18 // Importing routes
19 const birthdayRoute = require("./api/routes/birthdayRoute");
20 const importRoutes = require("./api/routes/importRoutes");
21 const quoteRoute = require("./api/routes/quoteRoute");
22
23 // Registering routes
24 server.use("/api", importRoutes);
25 server.use("/api", birthdayRoute);
26 server.use("/api", quoteRoute);
27
28 // Function to convert date format from DD/MM/YYYY to YYYY-MM-DD
29 function convertDateFormat(dateStr) {
30   const [day, month, year] = dateStr.split("/");
31   return `${year}-${month}-${day}`;
32 }
```

```
61 // Import CSV data before starting the server
62 async function initializeDatabase() {
63   await importCSVData(
64     path.resolve(__dirname, "./data/students.csv"),
65     "INSERT IGNORE INTO students (birthday, lastname, firstname, email) VALUES (?, ?, ?, ?)",
66     ["birthday", "lastname", "firstname", "email"],
67     "birthday"
68   );
69
70   await importCSVData(
71     path.resolve(__dirname, "./data/quotes.csv"),
72     "INSERT IGNORE INTO quotes (quote, author) VALUES (?, ?)",
73     ["quote", "author"]
74   );
75
76   server.listen(port, hostname, () => {
77     console.log(`Serveur qui tourne sur le port ${port}`);
78   });
79 }
80
81 initializeDatabase();
```



---

# Service API pour les Anniversaires et Citations

Définit les fonctions pour interagir avec les API de l'application afin de récupérer les anniversaires d'aujourd'hui et des citations aléatoires.

Utilise **Axios** pour effectuer des requêtes HTTP vers les endpoints API.

**getTodaysBirthday** : Récupère la liste des anniversaires d'aujourd'hui.

**getRandomQuote** : Récupère une citation aléatoire.

Gestion des Erreurs : En cas d'erreur de requête, un message est affiché dans la console et la fonction retourne false.

```
import axios from "axios";

const apiUrl = process.env.REACT_APP_API_URL || "http://localhost:3002";

export const getTodaysBirthday = async () => {
  let queryUrl = `${apiUrl}/api/getBirthday`;

  try {
    const response = await axios.get(queryUrl);
    return response.data;
  } catch (error) {
    console.error("Error fetching birthdays:", error);
    return false;
  }
};

export const getRandomQuote = async () => {
  let queryUrl = `${apiUrl}/api/randomquote`;

  try {
    const response = await axios.get(queryUrl);
    return response.data;
  } catch (error) {
    console.error("Error fetching quote:", error);
    return false;
  }
};
```

---

# Configuration de l'Application (Front)

Configure l'application principale en intégrant les composants et en gérant les états et effets.

Utilisation de Hooks : useState, useEffect, et useMemo pour gérer les états et les effets secondaires.

Gestion des Données :

getTodaysBirthday : Récupère les anniversaires du jour.

getRandomQuote : Récupère une citation aléatoire.

Gestion de l'État :

Birthdays : Liste des anniversaires.

CurrentBirthday : Anniversaire actuellement affiché.

CurrentIndex : Index de l'anniversaire affiché.

CurrentQuote : Citation actuelle.

```
28   useEffect(() => {
29     getTodaysBirthday()
30       .then((result) => {
31         const students_birthday = result.students_birthday || [];
32         const teachers_birthday = result.teachers_birthday || [];
33         let birthdaysList = [...students_birthday, ...teachers_birthday];
34         console.log("Fetched Birthdays:", birthdaysList);
35         setBirthdays(birthdaysList);
36       })
37       .catch(() => {
38         setBirthdays([]);
39       });
40
41     getRandomQuote()
42       .then((result) => {
43         setCurrentQuote(result);
44       })
45       .catch(() => {
46         setCurrentQuote(null);
47       });
48   }, []);
49
50   useEffect(() => {
51     if (Birthdays.length > 0) {
52       setCurrentColor(colorsList[CurrentIndex % colorsList.length]);
53       setCurrentBirthday({
54         ...Birthdays[CurrentIndex],
55         index: CurrentIndex,
56       });
57     }
58   }, [Birthdays, CurrentIndex, colorsList]);
59
```

```

60  useEffect(() => {
61    if (Birthdays.length > 0) {
62      const intervalId = setInterval(() => {
63        const index =
64          | CurrentIndex === Birthdays.length - 1 ? 0 : CurrentIndex + 1;
65        setCurrentIndex(index);
66        setCurrentColor(colorsList[index % colorsList.length]);
67        setCurrentBirthday({
68          | ...Birthdays[index],
69          | index: index,
70        });
71
72        // Update random quote
73        getRandomQuote()
74          .then((result) => {
75            | setCurrentQuote(result);
76          })
77          .catch(() => {
78            | setCurrentQuote(null);
79          });
80      }, 5000);
81
82      return () => clearInterval(intervalId);
83    }
84  }, [Birthdays, CurrentIndex, colorsList]);
85

```

```

86  return (
87    <div>
88      <HeaderComponent />
89      {Birthdays.length > 0 && CurrentBirthday ? (
90        <div className="h-screen w-screen flex">
91          <BirthdayComponent
92            | currentBirthday={CurrentBirthday}
93            | currentColor={CurrentColor}
94          />
95          <SidebarComponent
96            | currentBirthday={CurrentBirthday}
97            | totalBirthdays={Birthdays.length}
98            | currentColor={CurrentColor}
99            | currentQuote={CurrentQuote}
100          />
101        </div>
102      ) : (
103        <QuoteOnlyComponent currentColor={CurrentColor} />
104      )}
105    </div>
106  );
107

```

---

# Démonstration

---