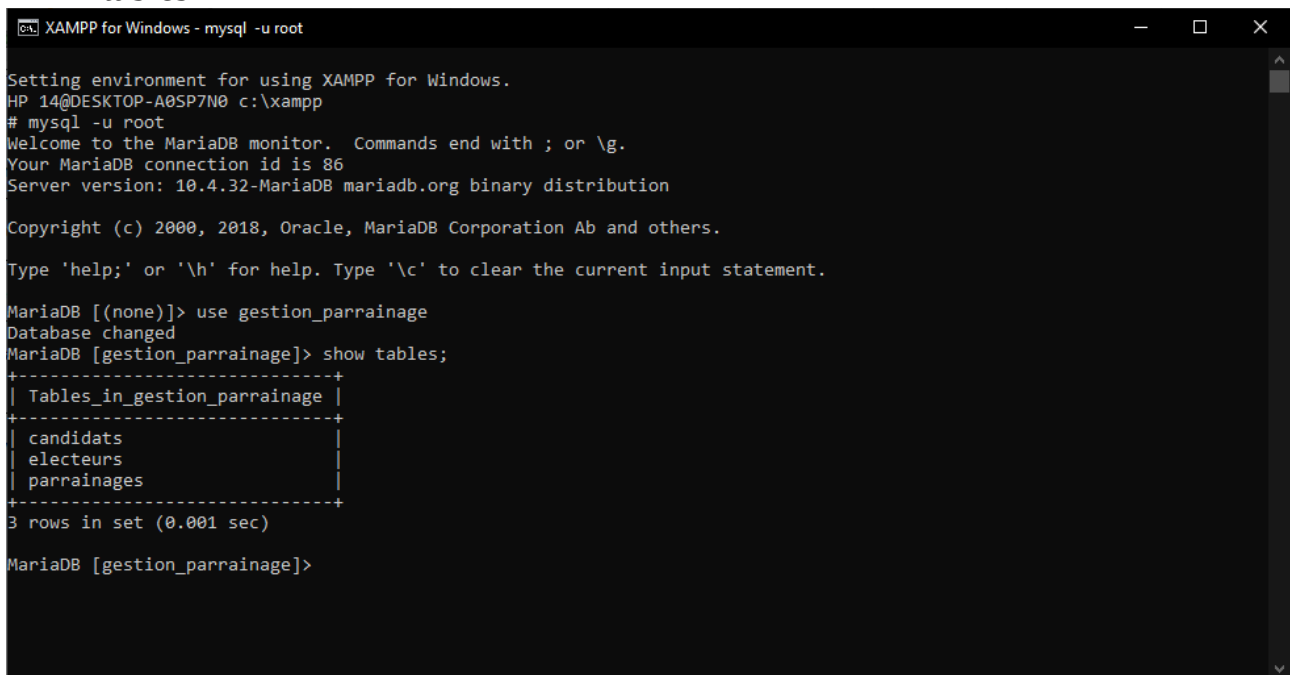


Fatou Kiné NDIAYE

Pour une **gestion optimale des bases de données**, il est essentiel de mettre en place **une conception**, une **optimisation** et une **gestion efficaces** des bases de données, tout en assurant la vérification et le contrôle de la qualité des données.

Partie 1 : Conception et gestion de la base de données

- J'ai d'abord créé ma base de données **gestion_parrainage** et les différentes tables



```
XAMPP for Windows - mysql -u root

Setting environment for using XAMPP for Windows.
HP 14@DESKTOP-A0SP7N0 c:\xampp
# mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 86
Server version: 10.4.32-MariaDB mariadb.org binary distribution

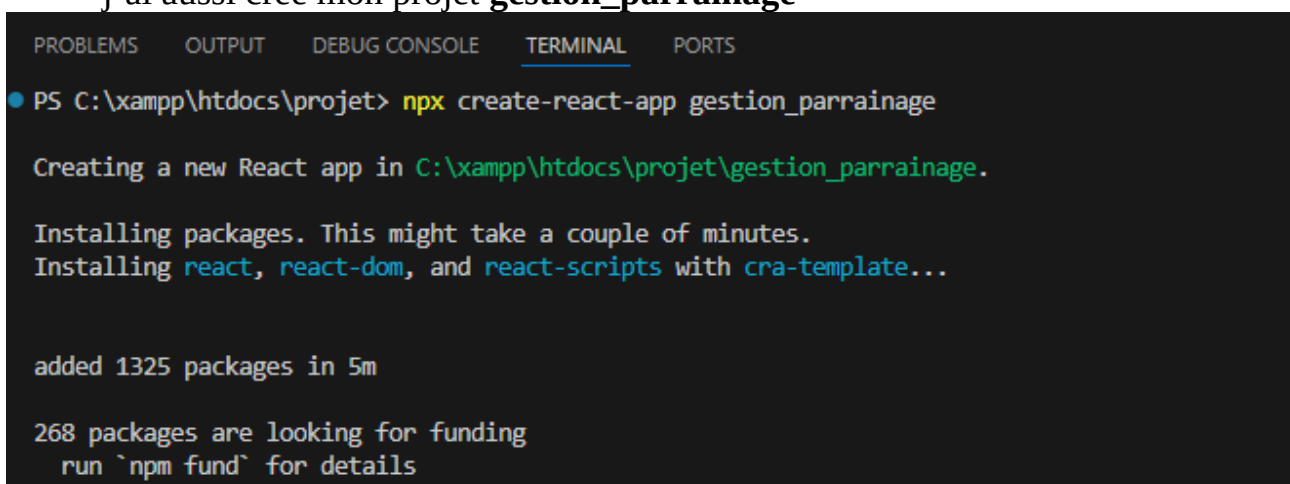
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use gestion_parrainage
Database changed
MariaDB [gestion_parrainage]> show tables;
+-----+
| Tables_in_gestion_parrainage |
+-----+
| candidats                    |
| electeurs                    |
| parrainages                    |
+-----+
3 rows in set (0.001 sec)

MariaDB [gestion_parrainage]>
```

- j'ai aussi créé mon projet **gestion_parrainage**



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

● PS C:\xampp\htdocs\projet> npx create-react-app gestion_parrainage

Creating a new React app in C:\xampp\htdocs\projet\gestion_parrainage.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

added 1325 packages in 5m

268 packages are looking for funding
run `npm fund` for details
```

- Utilise la commande suivante pour créer un fichier package.json

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\xampp\htdocs\projet\gestion_parrainage> npm init -y
Wrote to C:\xampp\htdocs\projet\gestion_parrainage\package.json:

{
  "name": "gestion_parrainage",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/dom": "^10.4.0",
    "@testing-library/jest-dom": "^6.6.3",
    "@testing-library/react": "^16.2.0",
    "@testing-library/user-event": "^13.5.0",
    "react": "^19.0.0",
```

Installer Express :

- Express est le framework qui me permettra de gérer les routes et les requêtes HTTP.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\xampp\htdocs\projet\gestion_parrainage> npm install express

up to date, audited 1343 packages in 7s

268 packages are looking for funding
  run `npm fund` for details

12 vulnerabilities (6 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force
```

Installer MySQL2 :

- MySQL2 est le package qui permet de communiquer avec une base de données MySQL depuis Node.js.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\xampp\htdocs\projet\gestion_parrainage> npm install mysql2

added 11 packages, and audited 1354 packages in 8s

269 packages are looking for funding
  run `npm fund` for details

12 vulnerabilities (6 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

Installer le package csv-parser :

- Ce package va m'aider à lire et à analyser les fichiers CSV.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\xampp\htdocs\projet\gestion_parrainage> npm install csv-parser

added 1 package, and audited 1355 packages in 7s

269 packages are looking for funding
  run `npm fund` for details

12 vulnerabilities (6 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

Installer le package sha256 :

- Ce package permet de vérifier l'empreinte SHA256 d'un fichier pour s'assurer de son intégrité.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\xampp\htdocs\projet\gestion_parrainage> npm install sha256
•
added 3 packages, and audited 1358 packages in 6s

269 packages are looking for funding
  run `npm fund` for details

12 vulnerabilities (6 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

Installer dotenv :

- Si on souhaite gérer les configurations (comme les informations de la base de données) de manière sécurisée dans un fichier `.env`, installe dotenv

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\xampp\htdocs\projet\gestion_parrainage> npm install dotenv
•
added 1 package, changed 1 package, and audited 1359 packages in 6s

270 packages are looking for funding
  run `npm fund` for details

12 vulnerabilities (6 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

- Ensuite, j'ai créé un fichier `.env` pour stocker les informations sensibles

```

gestion_parrainage > .env
1  DB_HOST=localhost
2  DB_USER=root
3  DB_PASSWORD=
4  DB_NAME=gestion_parrainage
5
6
7  require('dotenv').config();
8
9  const db = mysql.createPool({
10     host: process.env.DB_HOST,
11     user: process.env.DB_USER,
12     password: process.env.DB_PASSWORD,
13     database: process.env.DB_NAME,
14 });
15

```

Configuration de la base de données MySQL.

```

gestion_parrainage > src > config > JS db.js > ...
1  const mysql = require('mysql2');
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     waitForConnections: true,
10    connectionLimit: 10,
11    queueLimit: 0
12 });
13
14 pool.getConnection((err, connection) => {
15     if (err) {
16         console.error('Erreur de connexion à la base de données :', err);
17     } else {
18         console.log('Connexion à la base de données réussie');
19         connection.release();
20     }
21 });
22
23 module.exports = pool;
24

```

Partie 2 : Validation des fichiers CSV

Pour implémenter la fonction PL/SQL ControlierFichierElecteurs afin de vérifier l'empreinte SHA256 et le format UTF-8 du fichier CSV ainsi que la gestion de stockage des données dans une table temporaire en attendant validation, j'ai commencer par :

- Utilisez chardet pour vérifier l'encodage

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\xampp\htdocs\projet\gestion_parrainage> npm install chardet

added 1 package, and audited 1362 packages in 8s

270 packages are looking for funding
  run `npm fund` for details

12 vulnerabilities (6 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

- J'ai créé le fichier server.js

```
gestion_parrainage > src > JS server.js > ...
41 // API pour uploader et vérifier le fichier
42 app.post('/upload', upload.single('file'), (req, res) => {
43   if (!req.file) {
44     return res.status(400).send('Aucun fichier envoyé');
45   }
46
47   const filePath = req.file.path;
48   const { sha256, encoding } = verifyFile(filePath);
49
50   const sql = 'CALL ControlierFichierElecteurs(?, ?)';
51   connection.query(sql, [sha256, encoding], (error, results) => {
52     if (error) {
53       console.error('Erreur dans l\'exécution de la procédure:', error);
54       return res.status(500).send('Erreur dans l\'exécution de la procédure : ' + error);
55     }
56
57     res.send(`Fichier vérifié avec succès. Empreinte SHA256: ${sha256}, Encodage: ${encoding}`);
58   });
59 });
60
61 // Lancer le serveur
62 app.listen(port, () => {
63   console.log(`Serveur en cours d'exécution sur http://localhost:${port}`);
64 });
65
```

- J'ai créé la **procédure PL/SQL** qui effectue une validation de l'empreinte SHA256 pour un fichier CSV, puis insère les données dans une table temporaire (en attendant la validation du fichier).
- J'ai aussi créé la table **fichiers_verifies** pour pouvoir y insérer des données.

```

MariaDB [gestion_parrainage]> DESCRIBE fichiers_verifies;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int(11) | NO | PRI | NULL | auto_increment |
| sha256 | varchar(64) | NO | | NULL | |
| encoding | varchar(10) | NO | | NULL | |
| created_at | timestamp | NO | | current_timestamp() | |
+-----+-----+-----+-----+-----+-----+
1 rows in set (0.019 sec)

MariaDB [gestion_parrainage]>

```

Faisons le teste avec **Postman**, on redémarre d'abord le serveur

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\xampp\htdocs\projet\gestion_parrainage\src> node server.js
Serveur en cours d'exécution sur http://localhost:3000

```

The screenshot shows the Postman interface for a POST request to `http://localhost:3000/upload`. The request body is set to `form-data` with a single key-value pair: `file` (File) with value `TEST3.csv.txt`. The response status is `200 OK` with a response time of `559 ms` and a body size of `376 B`. The response body is in JSON format:

```

{
  "message": "Fichier vérifié avec succès.",
  "sha256": "6d41fa91f01fc3adb681dce3bedb69603413c25dbe876bb6d36ead667f05b6f9",
  "encoding": "UTF-8"
}

```

Si on vérifie dans la table `fichiers_vérifiés`, on voit bien l'ajout du fichier

The screenshot shows the phpMyAdmin interface for the 'gestion_parrainage' database. The left sidebar lists various tables and schemas. The main panel displays the structure of the 'table_temporaire' table. At the top, there are links for 'Profilage', 'Éditer en ligne', 'Éditer', 'Expliquer SQL', 'Créer le code source PHP', and 'Actualiser'. Below these are filters for 'Tout afficher', 'Nombre de lignes' (set to 25), 'Filtrer les lignes' (Chercher dans cette table), and 'Trier par clé' (Aucun(e)). The table structure is shown in a grid with columns: id, sha256, encoding, and created_at. There are three rows of data. Below the table, there are options to 'Tout cocher', 'Avec la sélection', 'Éditer', 'Copier', 'Supprimer', and 'Exporter'. At the bottom, there are more filters similar to the top ones.

	id	sha256	encoding	created_at
<input type="checkbox"/>	1	0e5fcd0ab67714bb21a506efb178e3694883793b387f4c4313...	UTF-8	2025-02-20 02:20:58
<input type="checkbox"/>	2	sha256_value	UTF-8	2025-02-20 10:46:54
<input type="checkbox"/>	3	6d41fa91f01fc3adb681dce3bedb69603413c25dbe876bb6d3...	UTF-8	2025-02-20 14:28:22

Partie 3 : Validation des données des électeurs et des candidats

Pour implémenter la fonction PL/SQL ControlerElecteurs afin de vérifier la complétude et la cohérence des données (CIN, nom, prénom, etc.) ainsi que la gestion des erreurs et les enregistrements problématiques dans une table temporaire. Nous allons :

- Tout d'abord, créer une table temporaire pour stocker les enregistrements qui échouent les validations.

```

MariaDB [gestion_parrainage]> DESCRIBE table_temporaire;
+-----+-----+-----+-----+-----+-----+
Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
fichier_blob | longblob  | YES  |     | NULL    |       |
sha256_hash  | varchar(64) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
rows in set (0.023 sec)

MariaDB [gestion_parrainage]>

```

- Puis créer une fonction PL/SQL pour valider les données des électeurs et des candidats. La fonction vérifiera plusieurs critères et enregistrera les erreurs dans la table temporaire.

On teste avec Postman

```

PS C:\xampp\htdocs\projet\gestion_parrainage\src> node server.js
Serveur en cours d'exécution sur http://localhost:3000

```


- J'ai créé Une **table temporaire** `electeurs_temp` contenant les données en attente de validation.

```

type help, or (h) for help. Type (c) to clear the current input statement.

MariaDB [(none)]> use gestion_parrainage;
Database changed
MariaDB [gestion_parrainage]> DESCRIBE electeurs_temp;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int(11) | NO | PRI | NULL | auto_increment |
| cin   | varchar(255) | YES | | NULL | |
| numero_electeur | varchar(255) | YES | | NULL | |
| nom   | varchar(100) | YES | | NULL | |
| prenom | varchar(100) | YES | | NULL | |
| date_naissance | date | YES | | NULL | |
| bureau_vote | varchar(100) | YES | | NULL | |
| email | varchar(100) | YES | | NULL | |
| telephone | varchar(20) | YES | | NULL | |
| code_authentification | varchar(255) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
10 rows in set (0.068 sec)

MariaDB [gestion_parrainage]>

```

- Créer la procédure **ValiderImportation** qui permet de **transférer les données validées** depuis une **table temporaire** (`electeurs_temp`) vers la **table principale** (`electeurs`). Ensuite, elle doit **supprimer les données** de la table temporaire une fois l'importation réussie.

```

XAMPP for Windows - mysql -u root
| character_set_client | collation_co ^
+-----+-----+-----+-----+-----+-----+
| ValidImportation | NO_ZERO_IN_DATE,NO_ZERO_DATE,NO_ENGINE_SUBSTITUTION | CREATE DEFINER='root'@'localhost' PROCEDURE
`ValidImportation`()
BEGIN
  INSERT INTO electeurs (cin, numero_electeur, nom, prenom, date_naissance, bureau_vote, email, telephone, code_authen
tification)
  SELECT cin, numero_electeur, nom, prenom, date_naissance, bureau_vote, email, telephone, code_authentification
  FROM electeurs_temp
  WHERE cin NOT IN (SELECT cin FROM electeurs);

  DELETE FROM electeurs_temp
  WHERE cin IN (SELECT cin FROM electeurs);

END | utf8mb4 | utf8mb4_unicode_ci | utf8mb4_general_ci |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.000 sec)

MariaDB [gestion_parrainage]>

```

- J'ai ajouter une Api pour exécuter la peocédure ValiderImportation

```
// Route pour exécuter la procédure ValiderImportation
app.post('/valider-importation', (req, res) => {
  const sql = 'CALL ValiderImportation()';

  // Exécuter la procédure stockée
  connection.query(sql, (error, results) => {
    if (error) {
      console.error('Erreur lors de l\'exécution de la procédure ValiderImportation:', error);
      return res.status(500).send('Erreur lors de l\'exécution de la procédure : ' + error);
    }

    // Répondre avec le résultat de l'exécution de la procédure
    res.send('La procédure ValiderImportation a été exécutée avec succès.');
```

Testons avec Postman

```
PS C:\xampp\htdocs\projet\gestion_parrainage\src> node server.js
Serveur en cours d'exécution sur http://localhost:3000
```

On a les données ci dessous dans la table electeurs_temps

Procédures

Tables

Nouvelle table

candidats

electeurs

electeurs_temp

erreurs_validation

fichiers_verifies

parrainages

table_temporaire

information_schema

mysql

performance_schema

☐

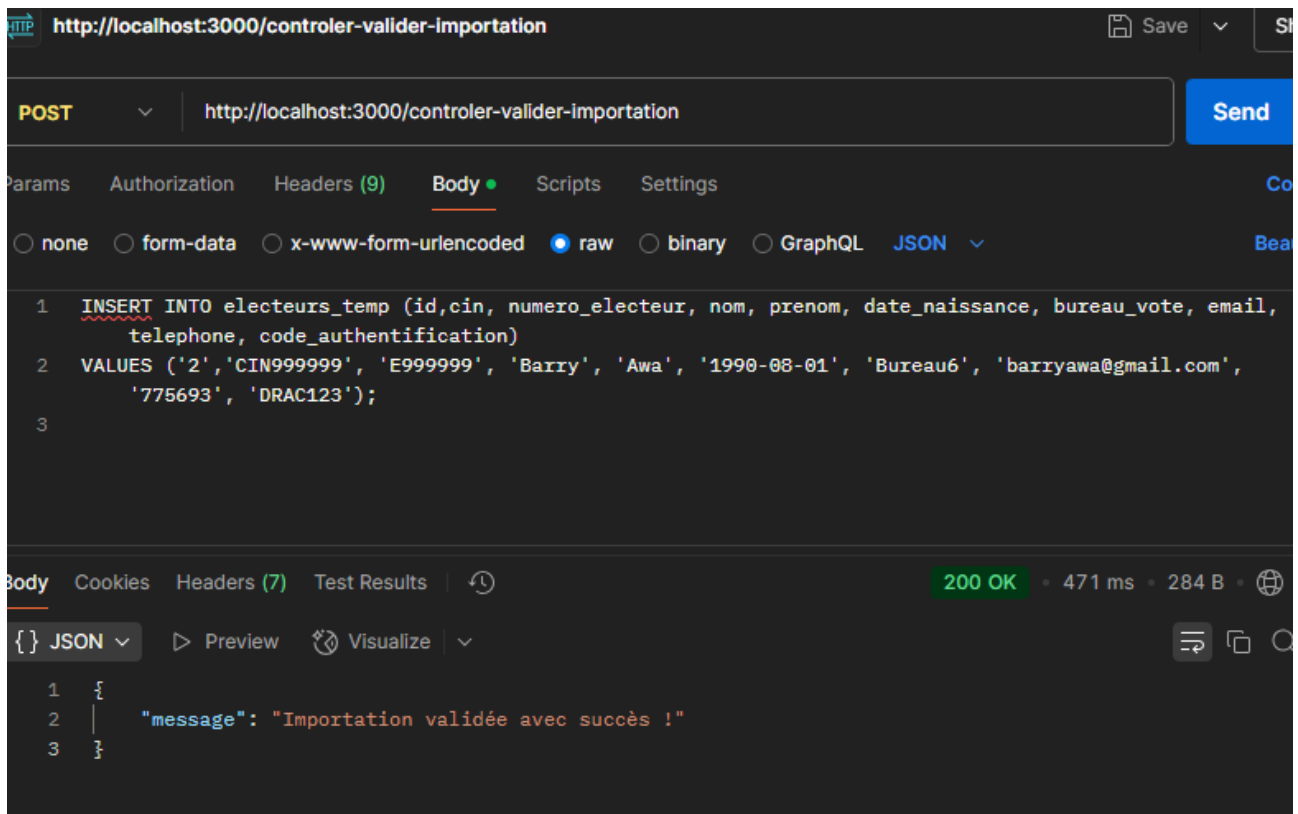
Tout afficher

Nombre de lignes : 25

Filtrer les lignes:

Options supplémentaires

On voit que l'importation a été faite



Si on vérifie de nouveau la table `electeurs_temp`, on voit que la table est vide.

```
MariaDB [gestion_parrainage]> SELECT * FROM electeurs_temp;
Empty set (0.001 sec)

MariaDB [gestion_parrainage]>
```

Les données de la tables `electeurs_temp` ont été importées dans la table `electeurs`

```
MariaDB [gestion_parrainage]> SELECT * FROM electeurs;
+-----+-----+-----+-----+-----+-----+-----+-----+
id | cin      | numero_electeur | nom  | prenom | date_naissance | bureau_vote | email
+-----+-----+-----+-----+-----+-----+-----+-----+
1 | CIN123456 | E123456         | Diop | Awa    | 1985-05-12     | Bureau 1   | NULL
2 | CIN234567 | E234567         | Samb | Bintou | 1990-03-22     | Bureau 2   | NULL
3 | CIN345678 | E345678         | Gaye | Omar   | 1982-11-04     | Bureau 3   | NULL
4 | CIN456789 | E456789         | Dieng | Sophie | 1978-07-18     | Bureau 4   | NULL
5 | CIN567890 | E567890         | Sow  | Modou  | 1995-02-25     | Bureau 5   | NULL
6 | 987654321 | 123456789       | Ndiaye | Fatou Kin| 2003-02-10     | BV001      | ndiayefatoukine232@gmail.com
7 | 777777777 | ABC123         | Test  | Utilisateur | 1990-01-01     | Bureau Test | test@example.com
8 | CIN999999 | E999999         | Barry | Awa    | 1990-08-01     | Bureau6    | barryawa@gmail.com
9 | CIN999999 | E999999         | Barry | Awa    | 1990-08-01     | Bureau6    | barryawa@gmail.com
+-----+-----+-----+-----+-----+-----+-----+-----+
rows in set (0.551 sec)

MariaDB [gestion_parrainage]>
```

Récapitulatif

1. Base de données optimisée

- Création des tables électeurs, candidats et parrainages avec leurs relations.
- Utilisation de clés primaires et étrangères pour assurer l'intégrité des données.
- Optimisation des requêtes SQL pour améliorer les performances.

2. Validation des fichiers CSV

- Implémentation de la fonction `ControlerFichierElecteurs` pour vérifier l'empreinte **SHA256** et le format **UTF-8** des fichiers importés.
- Stockage temporaire des données avant validation.

3. Contrôle des données électorales

- Vérification de la complétude et de la cohérence des informations électorales avec `ControlerElecteurs` (CIN, nom, prénom, etc.).
- Gestion des erreurs avec une table temporaire pour les données incorrectes.

4. Validation et transfert des données

- Utilisation de `ValiderImportation` pour déplacer les données validées de la **table temporaire** vers la **table définitive** (électeurs).
- Suppression des données temporaires après validation pour éviter toute redondance.

Ces étapes garantissent une importation **sécurisée, contrôlée et optimisée** des électeurs et candidats dans la base de données. 