

ORACLE

2023 - 2024



encadré par :
Meryem EL-
MOUHTADI

Prepared By:
Lakhmiri Mohammed
Elias

Sommaire

- **Problematique du sujet**
- **Creation des tables**
- **Creation d'utilisateur admin et normal**
- **Attribution des privilèges**
- **Simulation au niveau Oracle**
- **cote plateforme**
- **technologies utilisées**
- **Description**
- **code et Simulation au niveau de la platform**

Problematique du sujet

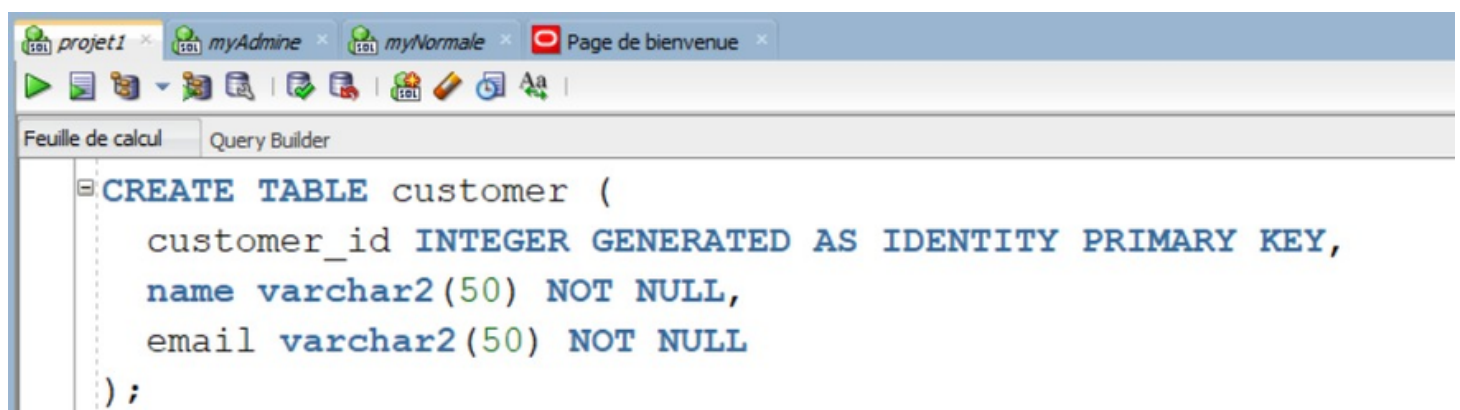
Gestion des utilisateurs et des autorisations. Il est important de s'assurer que les informations de connexion et les mots de passe sont stockées de manière sécurisée dans la base de données pour éviter les violations de sécurité. De plus, il faut s'assurer que les restrictions d'accès sont correctement implémentées pour chaque utilisateur, pour empêcher les utilisateurs non autorisés d'accéder à des informations sensibles. Il peut également y avoir des défis liés à la performance et à la stabilité de la plate-forme, en particulier lorsqu'elle est connectée à une base de données Oracle.

Creation des tables

J'ai commencé par créer une base de données nommée " projet1 " avec l'utilisateur " SYS as sysdba " puis j'ai créé les deux tables suivantes:

• Table customer

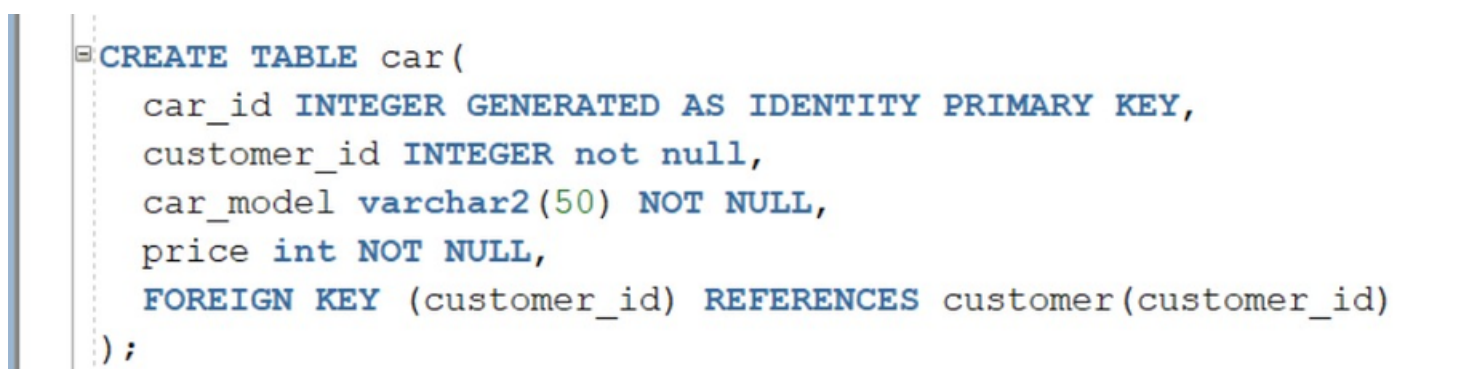
On crée une table appelée "customer". Elle comprend les colonnes "customer_id", "name" et "email". La colonne "customer_id" est définie comme une clé primaire générée automatiquement et unique pour chaque enregistrement de la table. Les colonnes "name" et "email" sont définies comme obligatoires et ne peuvent pas être nulles.

A screenshot of a SQL query editor window. The title bar shows several tabs: 'projet1', 'myAdmin', 'myNormale', and 'Page de bienvenue'. The 'Query Builder' tab is active. The editor contains the following SQL code:

```
CREATE TABLE customer (  
    customer_id INTEGER GENERATED AS IDENTITY PRIMARY KEY,  
    name varchar2(50) NOT NULL,  
    email varchar2(50) NOT NULL  
);
```

• Table car

La deuxième table appelée "car". Elle comprend les colonnes "car_id", "customer_id", "car_model" et "price". La colonne "car_id" est définie comme une clé primaire générée automatiquement et unique pour chaque enregistrement de la table. La colonne "customer_id" est définie comme obligatoire et ne peut pas être nulle. La colonne "car_model" et "price" sont également définies comme obligatoires et ne peuvent pas être nulles. La clé étrangère "FOREIGN KEY (customer_id) REFERENCES customer(customer_id)" établit une relation entre la table "car" et la table "customer", indiquant que pour chaque enregistrement dans la table "car", il y a un enregistrement correspondant dans la table "customer" avec la même valeur dans la colonne "customer_id".

A screenshot of a SQL query editor window showing the creation of the 'car' table. The editor contains the following SQL code:

```
CREATE TABLE car(  
    car_id INTEGER GENERATED AS IDENTITY PRIMARY KEY,  
    customer_id INTEGER not null,  
    car_model varchar2(50) NOT NULL,  
    price int NOT NULL,  
    FOREIGN KEY (customer_id) REFERENCES customer(customer_id)  
);
```

Remplissage des tables

- Insertion

```
INSERT INTO customer (name, email) VALUES ('elias', 'elias@gmail.com');
INSERT INTO customer (name, email) VALUES ('Lakhmiri', 'Lakhmiri@gmail.com');

INSERT INTO car (customer_id, car_model, price) VALUES (1, 'bmw M5', 120);
INSERT INTO car (customer_id, car_model, price) VALUES (2, 'mercede 220', 65);
```

- affichage

```
select * from car;
select * from customer;
```

Sortie de script x Résultat de requête x

SQL | Toutes les lignes extraites : 2 en 0,003 secondes

	CAR_ID	CUSTOMER_ID	CAR_MODEL	PRICE
1	1	1	bmw M5	120
2	2	2	mercede 220	65

```
select * from car;
select * from customer;
```

Sortie de script x Résultat de requête x

SQL | Toutes les lignes extraites : 2 en 0,014 secondes

	CUSTOMER_ID	NAME	EMAIL
1	1	elias	elias@gmail.com
2	2	Lakhmiri	Lakhmiri@gmail.com

- Creation d'un utilisateur admin et un autre normal

je crée un nouvel utilisateur appelé AD1 qui possèdera tout les privilèges avec le mot de passe AD1. Le tablespace par défaut pour l'utilisateur est users.

puis je crée un nouvel utilisateur appelé N1 qui possèdera des privilèges limite avec le mot de passe N1. Le tablespace par défaut pour l'utilisateur est users.

```
create user AD1 identified by AD1 default tablespace users;
create user N1 identified by N1 default tablespace users;
```

Privilèges

- Attribution des privilèges

```
grant all privileges to AD1;  
grant create session to N1;  
grant select on car to N1;
```

grant all privileges to AD1 : Cette ligne accorde tous les privilèges à l'utilisateur AD1.

grant create session to N1 : Cette ligne accorde le privilège create session à l'utilisateur N1. Ce privilège permet à l'utilisateur de se connecter à la base de données.

grant select on car to N1 : Cette ligne accorde le privilège select sur la table car à l'utilisateur N1. Cela permet à l'utilisateur de récupérer des données de la table car.

Simulation au niveau Oracle

- myNormale

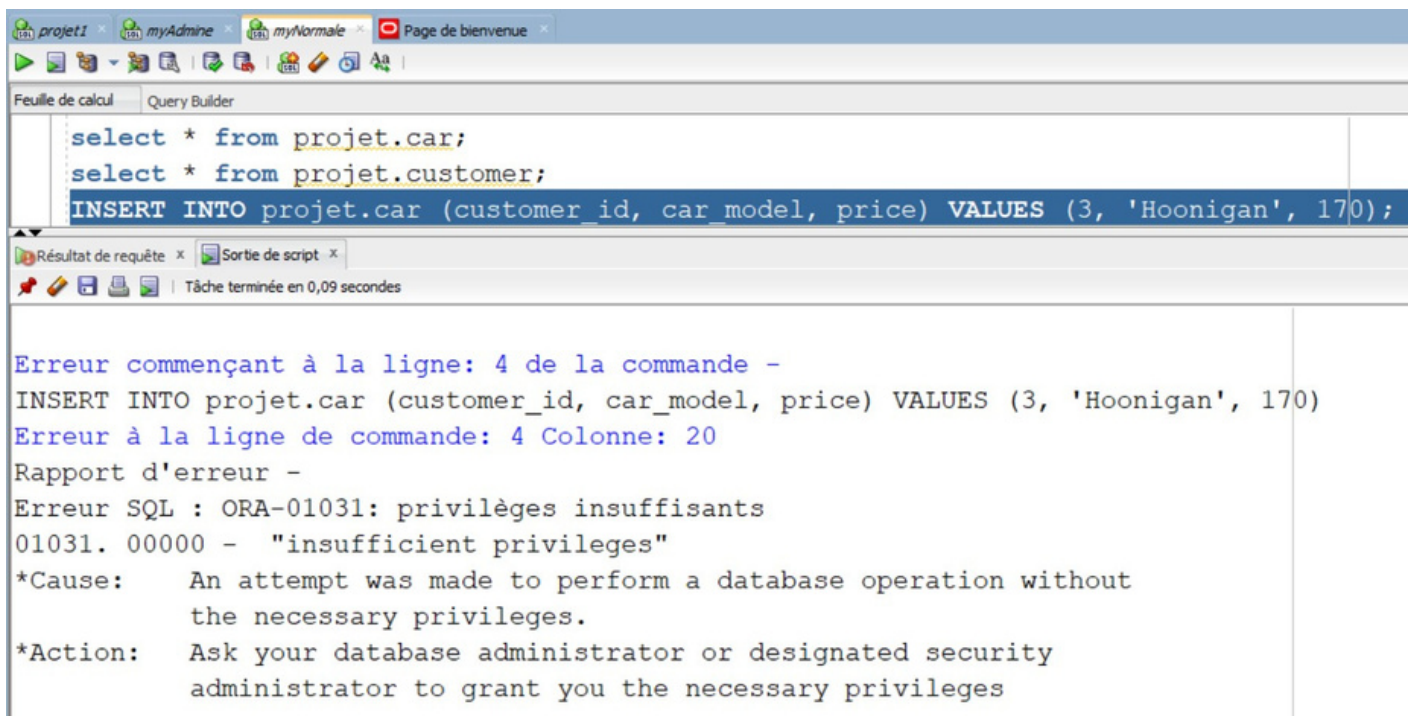
L'utilisateur N1 peut seulement récupérer des données de la table "car".
il ne peut ni modifier ni supprimer ni rajouter ni récupérer les données de la table "customer "

The screenshot shows the SQL Developer interface with the user 'myNormale' selected. The query window contains two SQL statements: `select * from projet.car;` and `select * from projet.customer;`. The 'Résultat de requête' (Query Result) window displays the results of the first query, showing two rows of data from the 'car' table.

	CAR_ID	CUSTOMER_ID	CAR_MODEL	PRICE
1	1		1bmw M5	120
2	2		2mercede 220	65

The screenshot shows the SQL Developer interface with the user 'myNormale' selected. The query window contains the same two SQL statements as the previous screenshot. However, the 'Résultat de requête' window displays an error message: 'ORA-00942: Table ou vue inexistante' (Table or view does not exist). The error details indicate the cause is 'table or view does not exist' and the action is 'Erreur à la ligne 2, colonne 22' (Error at line 2, column 22).

Simulation au niveau Oracle



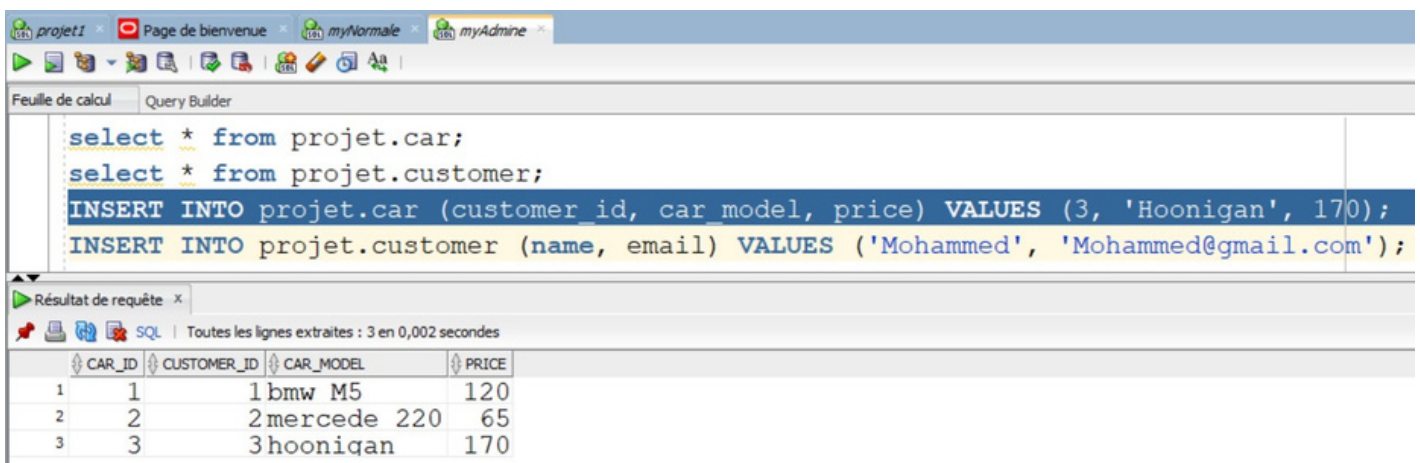
The screenshot shows the myAdmine interface with a query window. The query contains three lines: two SELECT statements and one INSERT statement. The INSERT statement is highlighted in blue. Below the query, the 'Résultat de requête' tab is active, displaying an error message in French. The error message states: 'Erreur commençant à la ligne: 4 de la commande - INSERT INTO projet.car (customer_id, car_model, price) VALUES (3, 'Hoonigan', 170) Erreur à la ligne de commande: 4 Colonne: 20 Rapport d'erreur - Erreur SQL : ORA-01031: privilèges insuffisants 01031. 00000 - "insufficient privileges" *Cause: An attempt was made to perform a database operation without the necessary privileges. *Action: Ask your database administrator or designated security administrator to grant you the necessary privileges'.

```
select * from projet.car;
select * from projet.customer;
INSERT INTO projet.car (customer_id, car_model, price) VALUES (3, 'Hoonigan', 170);
```

Erreur commençant à la ligne: 4 de la commande -
INSERT INTO projet.car (customer_id, car_model, price) VALUES (3, 'Hoonigan', 170)
Erreur à la ligne de commande: 4 Colonne: 20
Rapport d'erreur -
Erreur SQL : ORA-01031: privilèges insuffisants
01031. 00000 - "insufficient privileges"
*Cause: An attempt was made to perform a database operation without
the necessary privileges.
*Action: Ask your database administrator or designated security
administrator to grant you the necessary privileges

- **myAdmine**

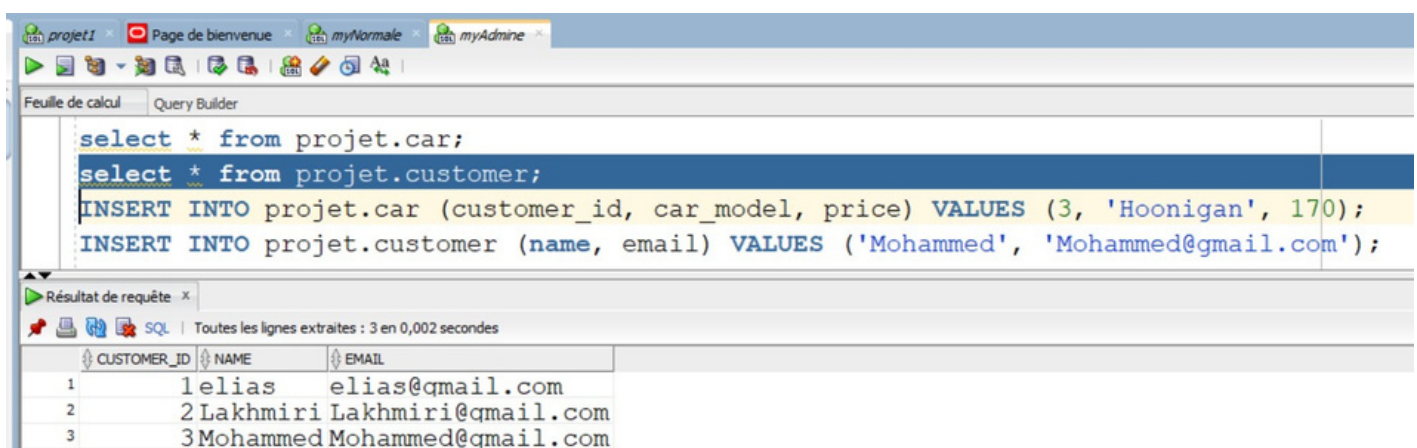
Au contraire, notre utilisateur AD1 qui représente l'administrateur dans notre exemple possède tous les privilèges, donc il peut récupérer les données de toutes les tables, il peut même insérer ou supprimer.



The screenshot shows the myAdmine interface with a query window. The query contains three lines: two SELECT statements and two INSERT statements. The INSERT statements are highlighted in blue. Below the query, the 'Résultat de requête' tab is active, displaying the results of the query. The results are shown in a table with four columns: CAR_ID, CUSTOMER_ID, CAR_MODEL, and PRICE. The table contains three rows of data.

```
select * from projet.car;
select * from projet.customer;
INSERT INTO projet.car (customer_id, car_model, price) VALUES (3, 'Hoonigan', 170);
INSERT INTO projet.customer (name, email) VALUES ('Mohammed', 'Mohammed@gmail.com');
```

	CAR_ID	CUSTOMER_ID	CAR_MODEL	PRICE
1	1	1	1bmw M5	120
2	2	2	2mercede 220	65
3	3	3	3hoonigan	170



The screenshot shows the myAdmine interface with a query window. The query contains three lines: two SELECT statements and two INSERT statements. The INSERT statements are highlighted in blue. Below the query, the 'Résultat de requête' tab is active, displaying the results of the query. The results are shown in a table with three columns: CUSTOMER_ID, NAME, and EMAIL. The table contains three rows of data.

```
select * from projet.car;
select * from projet.customer;
INSERT INTO projet.car (customer_id, car_model, price) VALUES (3, 'Hoonigan', 170);
INSERT INTO projet.customer (name, email) VALUES ('Mohammed', 'Mohammed@gmail.com');
```

	CUSTOMER_ID	NAME	EMAIL
1	1	1elias	elias@gmail.com
2	2	2Lakhmiri	Lakhmiri@gmail.com
3	3	3Mohammed	Mohammed@gmail.com

cote plateforme

- **technologies utilisées**

Dans ce projet, les technologies utilisées sont:

- Express (framework Node.js) pour la création de la plateforme web
- Oracledb pour la connexion à la base de données Oracle
- HTML, CSS et JavaScript pour la création de la page de connexion et de redirection
- La base de données utilisée est Oracle.

Description

- **Description d'une plateforme d'accès sécurisé avec plusieurs profils d'utilisateurs connectée à une base de données Oracle.**

Le code suivant permet de créer une plateforme web sécurisée avec une page de connexion qui demande un nom d'utilisateur et un mot de passe. Les utilisateurs autorisés sont stockés dans un tableau **allowedUsers**.

L'application utilise Express pour gérer les requêtes HTTP et les réponses. Elle utilise également Oracledb pour se connecter à la base de données Oracle.

Lorsque l'utilisateur soumet les informations de connexion, l'application vérifie si le nom d'utilisateur et le mot de passe correspondent à l'un des utilisateurs autorisés dans le tableau **allowedUsers**. Si la vérification réussit, l'utilisateur est redirigé vers une page avec les informations de la base de données Oracle. Sinon, un message d'erreur est affiché.

L'application peut être testée en connectant simultanément deux utilisateurs avec des droits d'accès différents pour vérifier si les restrictions d'accès sont respectées.

CODE

```
const express = require('express');
const oracledb = require('oracledb');
const fs = require('fs');

const app = express();

app.use(express.json());
app.use(express.urlencoded({ extended: true }));

const allowedUsers = [
  {
    username: 'AD1',
    password: 'AD1',
    schema: 'schema1'
  },
  {
    username: 'N1',
    password: 'N1',
    schema: 'schema2'
  }
];
```

on importe les modules 'express', 'oracledb' et 'fs' pour utilisation dans une application web. La variable 'app' est initialisée pour utiliser les fonctionnalités de 'express' pour gérer les requêtes HTTP. Le tableau 'allowedUsers' stocke des informations de connexion pour les utilisateurs autorisés à accéder à la plateforme.

```
app.post('/login', async (req, res) => {
  let connection;
  const { username, password } = req.body;

  const user = allowedUsers.find(u => u.username === username && u.password === password);

  if (!user) {
    return res.send('Login Failed');
  }
  if (username === 'AD1') {
    res.redirect('/admin');
  } else {
    res.redirect('/normal');
  }

  const log = `[${new Date().toString()}] User ${username} logged in\n`;
  fs.appendFile('journalisation.txt', log, (err) => {
    if (err) throw err;
    console.log('The log was saved!');
  });
});
```

puis on définit une route HTTP POST pour l'URL "/login". Lorsqu'un utilisateur soumet un formulaire de connexion avec un nom d'utilisateur et un mot de passe, la route vérifie si les informations de connexion sont présentes dans la liste des utilisateurs autorisés. Si les informations de connexion sont valides, le code redirige l'utilisateur vers une page "/admin" ou "/normal" en fonction de son nom d'utilisateur. Enfin, le code enregistre une entrée de journalisation avec le nom d'utilisateur, la date et l'heure de connexion dans un fichier texte.

CODE

```
app.get('/admin', async(req, res) => {
  let connection;
  try {
    connection = await oracledb.getConnection({
      user: "AD1",
      password: "AD1",
      connectionString: "localhost/orcl"
    });
    const sqlQueries = [ "select * from projet.customer", "select * from projet.car"];
    // const sqlQueries = [ "select * from projet.car"];
    const tables = [];
    for (const sql of sqlQueries) {
      const result = await connection.execute(sql);
      const tableHTML = result.rows.map(row => {
        let rowHTML = '';
        for (let i = 0; i < result.metaData.length; i++) {
          rowHTML += `<td>${row[i]}</td>`;
        }
        return `<tr>${rowHTML}</tr>`;
      }).join('');
      tables.push({ metaData: result.metaData, tableHTML });
    }
  }
});
```

cette partie Pour se connecter à notre base de données Oracle en local et exécuter nos requêtes SQL pour obtenir des données à partir des tables "projet.customer" et "projet.car". Les résultats des requêtes sont convertis en HTML et envoyés à la page d'administration. En cas d'erreur, un message "Error Occurred" est affiché dans la console et sur la page.

```
app.get('/normal', async(req, res) => {
  let connection;
  try {
    connection = await oracledb.getConnection({
      user: 'N1',
      password: 'N1',
      connectionString: "localhost/orcl"
    });
    //const sqlQueries = [ "select * from projet.customer", "select * from projet.car"];
    const sqlQueries = [ "select * from projet.car" ];

    const tables = [];
    for (const sql of sqlQueries) {
      const result = await connection.execute(sql);
      const tableHTML = result.rows.map(row => {
        let rowHTML = '';
        for (let i = 0; i < result.metaData.length; i++) {
          rowHTML += `<td>${row[i]}</td>`;
        }
        return `<tr>${rowHTML}</tr>`;
      }).join('');
      tables.push({ metaData: result.metaData, tableHTML });
    }
  }
});
```

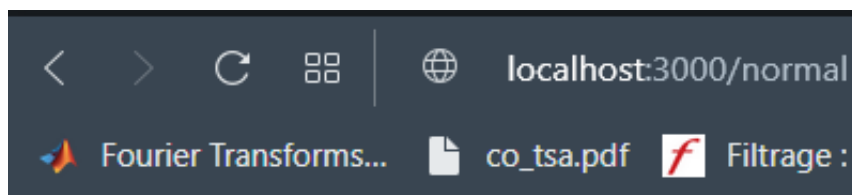
C'est le code pour gérer la route "/normal" dans notre application Express. Il se connecte à notre base de données Oracle avec les identifiants d'utilisateur "N1" et le mot de passe "N1", puis exécute une requête SQL pour obtenir les informations de la table "car".

CODE

```
app.get('/normal', async(req, res) => {
  let connection;
  try {
    connection = await oracledb.getConnection({
      user: 'N1',
      password: 'N1',
      connectionString: "localhost/orcl"
    });

    //const sqlQueries = [ "select * from projet.customer", "select * from projet.car"];
    const sqlQueries = [ "select * from projet.car","select * from projet.customer"];
  }
});
```

si on rajoute la requete **"select * from projet.customer"** on obtiendra une erreur , Cela arriver car l'utilisateur "N1" n'a pas les autorisations pour accéder à cette table



Error Occurred

CODE

Enfin, On définit un point de terminaison POST pour l'URL '/logout'. Lorsqu'un utilisateur envoie une requête POST à cette URL, le serveur ajoutera une entrée au fichier `journalisation.txt` indiquant que l'utilisateur s'est déconnecté. Après cela, le serveur redirigera l'utilisateur vers l'URL '/'.


```
<h1>Tables</h1>
${tables.map(table => `
  <table>
    <thead>
      <tr>
        ${table.metaData.map(col => `<th>${col.name}</th>`).join('')}
      </tr>
    </thead>
    <tbody>
      ${table.tableHTML}
    </tbody>
  </table>
`).join('')}
`);
} catch (error) {
  console.error(error);
  res.send('Error Occurred');
} finally {
}
});

app.post('/logout', async (req, res) => {
  try {
    // await connection.close();
    const log = `[${new Date().toString()}] User ${req.body.username} logged out\n`;
    fs.appendFile('journalisation.txt', log, (err) => {
      if (err) throw err;
      console.log('The log was saved!');
    });
  } catch (error) {
    console.error(error);
  }
  res.redirect('/');
});

app.listen(3000, () => console.log('Server Started'));
```


Simulation au niveau de la platform

- Login Admin



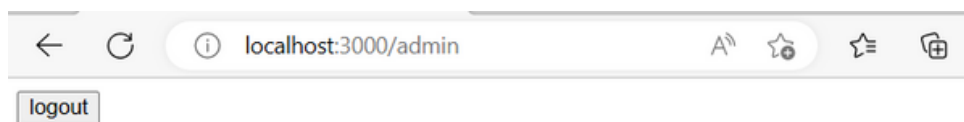
localhost:3000

AD1

...

Submit

- Tables(Admin)



localhost:3000/admin

logout

Tables

CUSTOMER_ID	NAME	EMAIL
1	elias	elias@gmail.com
2	Lakhmiri	Lakhmiri@gmail.com
3	Mohammed	Mohammed@gmail.com

CAR_ID	CUSTOMER_ID	CAR_MODEL	PRICE
1	1	bmw M5	120
2	2	mercede 220	65
3	3	hoonigan	170

Simulation au niveau de la platform

- Login Admin



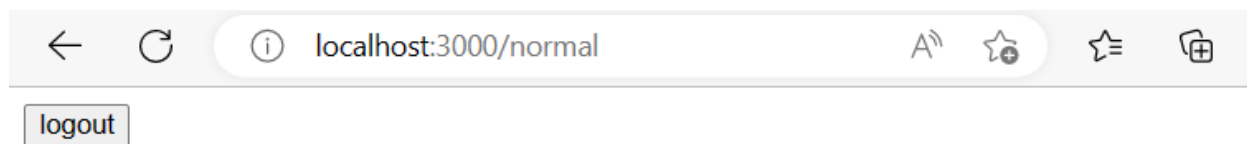
localhost:3000

N1

..

Submit

- Table(Normale utilisateur)



localhost:3000/normal

logout

Tables

CAR_ID	CUSTOMER_ID	CAR_MODEL	PRICE
1	1	bmw M5	120
2	2	mercede 220	65
3	3	hoonigan	170

CONCLUSION

En conclusion, ce projet a montré comment implémenter une application web simple avec ExpressJS et OracleDB. Il comprenait la création de pages de connexion et de déconnexion pour les utilisateurs, ainsi que l'affichage des données stockées dans une base de données Oracle. Le code fourni dans ce projet peut être utilisé comme base pour des projets plus avancés en utilisant ces technologies. Cependant, il est important de noter que ce code ne doit pas être utilisé dans un environnement de production sans une sécurité supplémentaire car il n'est pas sécurisé pour des raisons évidentes.