Documentation Technique
M2L

Technologies Utilisées

- Langage de programmation : JavaScript.
- Framework : Express.js
- Base de données : Identifier la base de données utilisée (MySQL).
- Dépendances :
- Js-cookie pour la gestion des sessions et la personnalisation du contenu
- bcrypt pour le hachage des mots de passe
- jsonwebtoken pour la gestion des tokens JWT
- crypto pour la génération d'UUID cryptographiquement sécurisés

Formulaire de connexion

- L'utilisateur remplit le formulaire de connexion en fournissant son adresse e-mail et son mot de passe.

```
    <form onSubmit={handleSubmit}>
        <label>Email:</label>
        <input
            type="email"
            name="email"
            value={formData.email}
            onChange={handleChange}
            required
        />

        <label>Mot de passe:</label>
        <input
            type="password"
            name="password"
            value={formData.password}
            onChange={handleChange}
            required
        />

        <button type="submit">Se connecter</button>

        {error && <p style={{ color: 'red' }}>{error}</p>}
    </form>
);
```

Gestion des Données de Formulaire et soumission

```jsx
const [formData, setFormData] = useState({
    email: '',
    password: '',
});

const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
};
```

```jsx
const handleSubmit = async (e) => {
    e.preventDefault();

    try {
        const response = await fetch('http://localhost:3000/api/usersroute/connexi
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            credentials: 'include',
            body: JSON.stringify(formData),
        });

        if (response.ok) {
            const { token } = await response.json();

            // Créez le cookie token
            Cookies.set('token', token);



            window.dispatchEvent(new Event('tokenUpdated'));
            handleTokenUpdate(token);
        } else {
            const errorData = await response.json();
            setError(errorData.error);
        }
    } catch (error) {
        console.error('Erreur lors de la soumission du formulaire :', error);
        setError('Erreur réseau lors de la connexion.');
    }
};
```

- Lorsque l'utilisateur soumet le formulaire, le navigateur envoie une requête POST au point de terminaison de connexion de l'API de l'application, a LURL spécifique (http://localhost:3000/api/usersroute/connexion). Une fois que le token JWT est généré, le serveur renvoie ce token dans la réponse HTTP vers le client. Côté client, le token est stocké dans un cookie nommé 'token' à l'aide de la méthode Cookies.set('token', token);. Ce cookie est automatiquement envoyé dans toutes les futures requêtes HTTP envoyées au serveur, facilitant ainsi l'authentification de l'utilisateur.

Redirection et Rafraîchissement de la Page

- Après la création et le stockage du token JWT, l'application peut rediriger l'utilisateur vers une page spécifique, telle que le tableau de bord de l'utilisateur, ou lui permettre d'accéder à d'autres fonctionnalités réservées aux utilisateurs connectés.

```
const handleTokenUpdate = (token) => {
    const decodedToken = jwtDecode(token);
    if (decodedToken.isAdmin) {
        navigate('/admin/dashboard');
        window.location.reload();
    } else {
        navigate('/acceuil');
    }
};
```

Vérification de l'existence de l'utilisateur dans la base de données

Dans le contrôleur de connexion, les informations d'identification fournies par l'utilisateur (adresse e-mail et mot de passe) sont extraites de la requête

```
const connection = await db.getConnection();

const [result] = await connection.query(
    'SELECT * FROM users WHERE email = ?',
    [email]
);

connection.release();

if (result.length === 0) {
    return res.status(401).json({ error: 'Utilisateur non trouvé' });
}
```

Vérification de la correspondance du mot de passe

Vérification de la correspondance entre le mot de passe fourni et le mot de passe hashé stocké dans la base de données.

```
const user = result[0];
const passwordMatch = await bcrypt.compare(password, user.password);

if (!passwordMatch) {
    return res.status(401).json({ error: 'Mot de passe incorrect' });
}
```

Génération du token JWT

Si les informations d'identification sont valides, l'application génère un JSON Web Token (JWT) en utilisant la fonction jwt.sign() fournie par le package jsonwebtoken. Ce token contient des informations sur l'utilisateur (telles que son UID, son adresse e-mail, son status admin.) ainsi que la date d'expiration.

```
const isAdmin = user.admin === 1;

const token = jwt.sign({
    uid: user.uid,
    email: user.email,
    isAdmin: isAdmin
}, process.env.API_KEY, { expiresIn: '1h' });
```

Stockage du token dans un cookie

```
res.cookie('token', token, { httpOnly: true, maxAge: 3600000 });
res.json({ token });
```