

=====

Titre: Livrable 4 Projet BDD

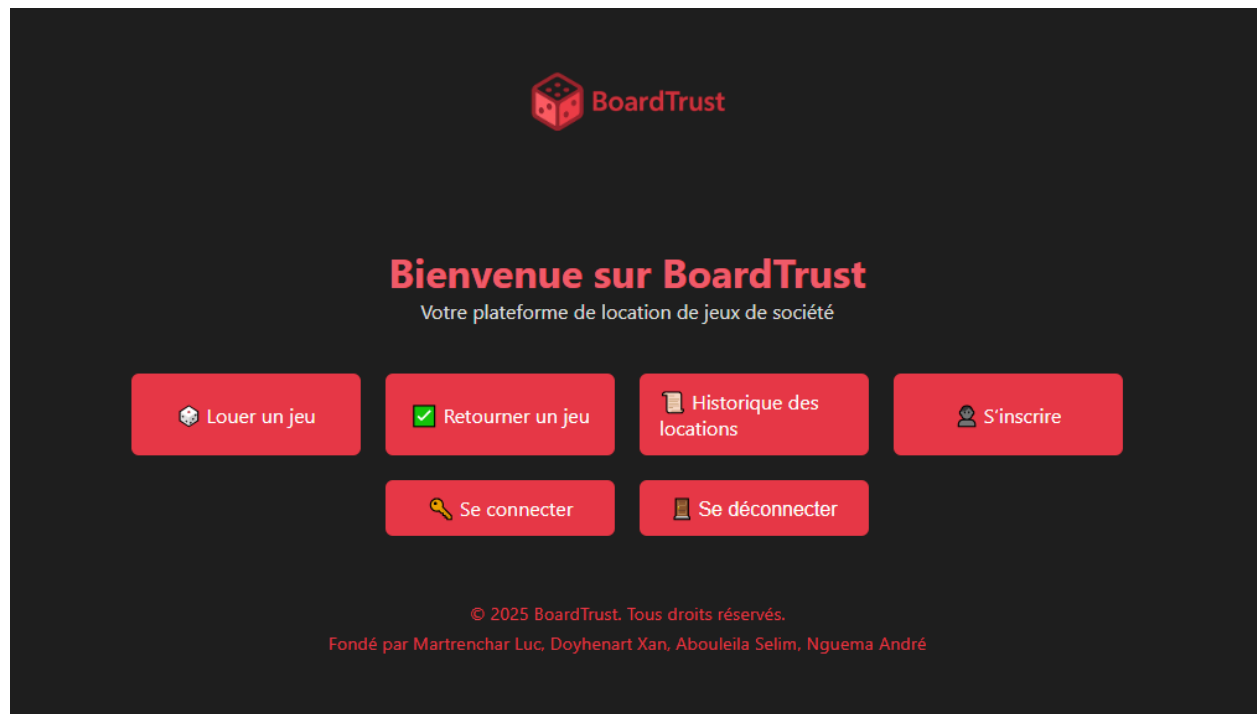
Participants: Martrenchar Luc, Doyhenart Xan, Abouleila Selim, Nguema André

=====


Nous avons décidé de créer un site web de location de jeux, développé entièrement avec Node.js (front-end et back-end), connecté à une base de données. Voici le lien du site : <https://boardtrust-production.up.railway.app/>

### Démonstration:

Page d'accueil:



## Fonctionnalité 1: Création de compte / Authentification



### Inscription

Pseudo

Email

Mot de passe

S'inscrire

© 2025 BoardTrust. Tous droits réservés.  
Fondé par Martrenchar Luc, Doyhenart Xan, Abouleila Selim, Nguema André

### Connexion

Email


Mot de passe

Se connecter

[Vous n'avez pas de compte ?](#) [Inscrivez-vous](#)

© 2025 BoardTrust. Tous droits réservés.  
Fondé par Martrenchar Luc, Doyhenart Xan, Abouleila Selim, Nguema André

## Fonctionnalité 2: Louer / Retourner jeu



Louer un jeu

**Catan**  
ID : 13  
Année de publication : 1995  
Note moyenne : 7  
Note bayésienne : 7  
Nombre d'utilisateurs : 108024

Louer

**Carcassonne**  
ID : 822  
Année de publication : 2000  
Note moyenne : 7  
Note bayésienne : 7  
Nombre d'utilisateurs : 108738

Louer

**Ticket to Ride**  
ID : 9209  
Année de publication : 2004  
Note moyenne : 7  
Note bayésienne : 7  
Nombre d'utilisateurs : 78120

Louer

**Twilight Struggle**  
ID : 12333  
Année de publication : 2005  
Note moyenne : 8  
Note bayésienne : 8  
Nombre d'utilisateurs : 68206

Louer

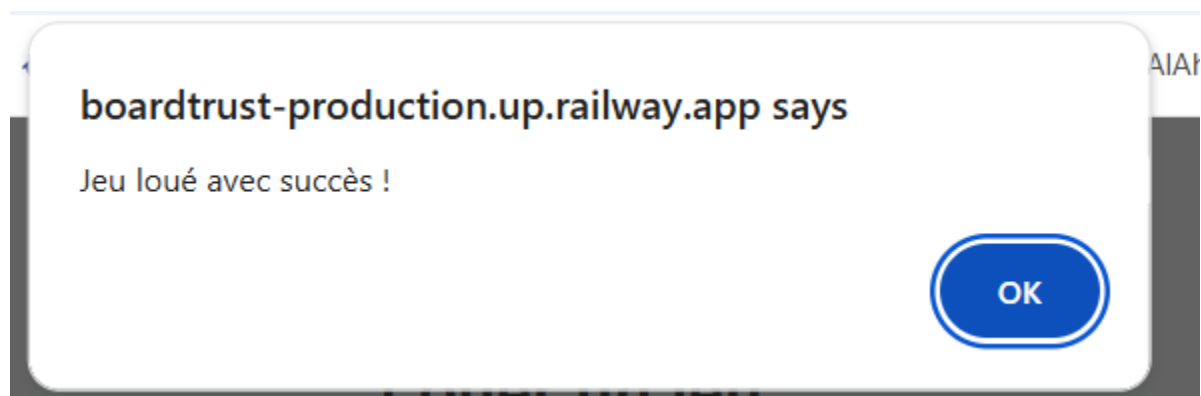
**Race for the Galaxy**  
ID : 25613  
Année de publication : 2007  
Note moyenne : 8  
Note bayésienne : 8  
Nombre d'utilisateurs : 58472

Louer

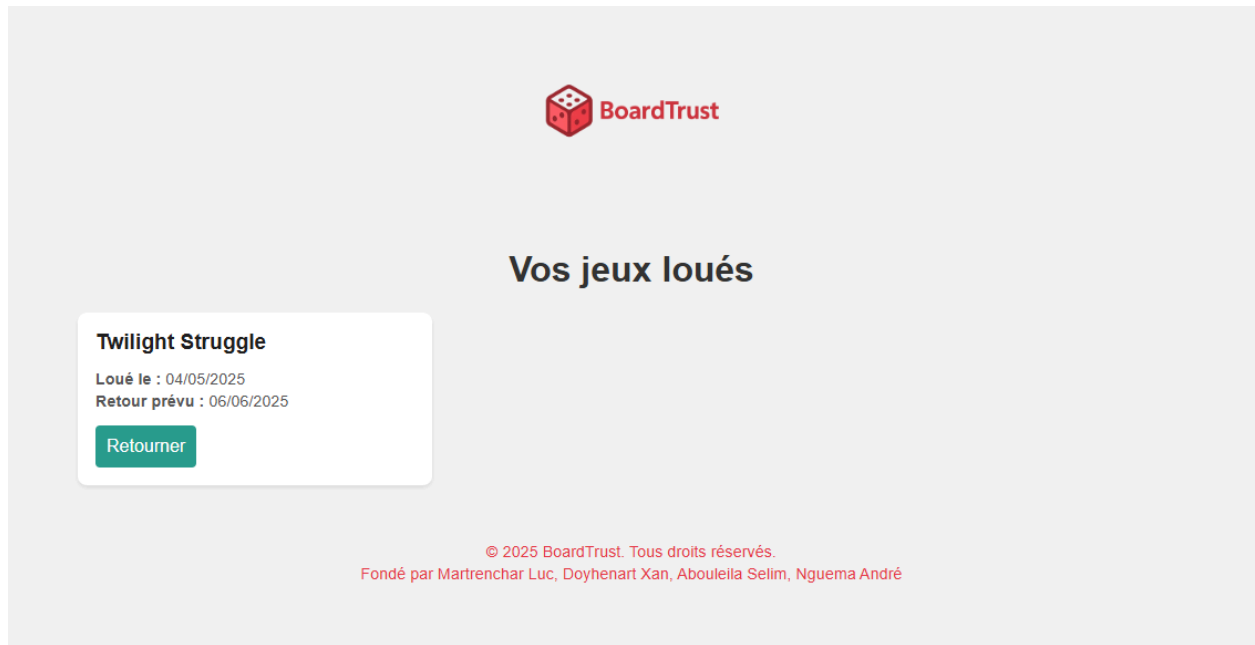
**Pandemic**  
ID : 30549  
Année de publication : 2008  
Note moyenne : 8  
Note bayésienne : 7  
Nombre d'utilisateurs : 108975

Louer

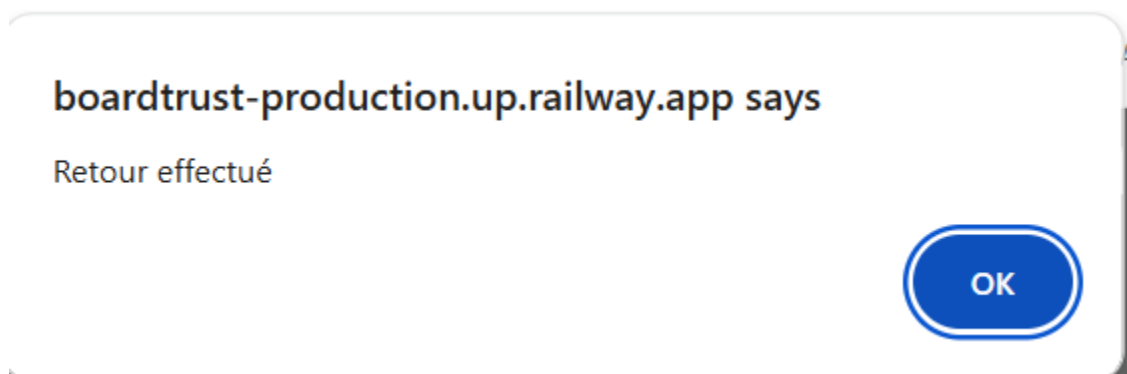
Par exemple nous allons louer Twilight Struggle pour le 06 / 06 / 2025



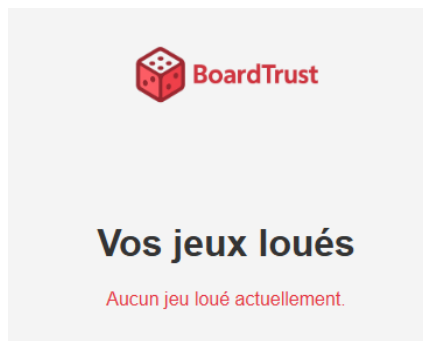
Maintenant, nous allons retourner le jeu qui va apparaître quand nous cliquons sur “retourner un jeu”



Après avoir cliqué sur “Retourner”



On rechargeant la page:



### Fonctionnalité 3: Voir l'historique



### Historique des locations

Game name	Date location	Date retour prevue	Date retour effective
Twilight Struggle	04/05/2025	06/06/2025	04/05/2025

© 2025 BoardTrust. Tous droits réservés.  
Fondé par Martrenchar Luc, Doyhenart Xan, Abouleila Selim, Nguema André

Tout fonctionne normalement !

```
const mysql = require('mysql2');
require('dotenv').config();

// Create the MySQL connection
const db = mysql.createConnection({
  host: process.env.DB_HOST,
  port: process.env.DB_PORT,
  user: process.env.DB_USER,
  password: process.env.DB_PASSWORD,
  database: process.env.DB_NAME
});

// Connect to database
db.connect((err) => {
  if (err) {
    console.error('Database connection failed: ' + err.stack);
    return;
  }
  console.log('Connected to database.');
```

L'application interagit avec la base de données via le back-end. Dans notre cas, le back-end est connecté à la base de données Railway, qui reste en ligne en permanence grâce au fichier `database.js`, où la connexion est établie comme vous pouvez le voir sur l'image ci-dessus. D'autres configurations ont aussi été effectuées sur le site de Railway pour relier le back-end déployé à la base de données.

Les requêtes vers la base de données sont gérées dans le fichier `index.js`, situé dans le dossier `backend`. C'est le cœur du site, car il gère l'interaction avec la base de données, comme vous pourrez le voir sur l'image suivante.

```
res.json({ message: 'Game returned successfully' });
});

/* READ-ONLY HELPERS */

app.get('/games', (_, res) => {
  db.query('SELECT * FROM VueJeuSansDescription', (err, results) => {
    if (err) return res.status(500).json({ error: 'Error fetching games' });
    res.json(results);
  });
});

app.post('/isRented', (req, res) => {
  const { id_utilisateur, id_jeu } = req.body;
  if (!id_utilisateur || !id_jeu) return res.status(400).json({ error: 'Missing id_utilisateur or id_jeu' });

  db.query('CALL EstLoueParUtilisateur(?, ?, @isRented)', [id_utilisateur, id_jeu], err => {
    if (err) return res.status(500).json({ error: 'Error calling procedure' });

    db.query('SELECT @isRented AS rented', (error, rows) => {
      if (error) return res.status(500).json({ error: 'Error retrieving rental status' });
      res.json({ rented: rows[0].rented === 1 });
    });
  });
});

/* Logout endpoint */
app.post('/logout', (req, res) => {
  console.log('Logout request received, session:', req.session);

  if (!req.session.userId) {
    console.log('Logout attempt: User not logged in');
  }
});
```

Par exemple, sur cette image, on peut voir comment le fichier `index.js` du back-end appelle la vue "historique" avec la requête `SELECT * FROM VueJeuSansDescription`, qui se trouve dans une requête GET accessible depuis le front-end. Bien sûr, ce n'est qu'un extrait du code disponible dans `index.js`. Il contient environ 30 autres fonctions similaires à `GET /games`.

Juste en dessous de ceci, on trouve la requête de déconnexion (logout), qui est cette fois une requête POST, car c'est le client qui envoie des données au serveur, et non l'inverse.