Synthèse TP02 : Concepts DB + API REST CRUD

1. Qu'est-ce qu'une BASE DE DONNÉES ?

Analogie simple :

```
┌──────────────────────────┐
│ Table "Étudiants"        │
├──────────────────────────┤
│ id │ prénom  │ nom       │
├────┼─────────┼───────────┤
│ 1  │ John    │ Doe       │
│ 2  │ Marie   │ Dupont    │
└────┴─────────┴───────────┘
```

Dans notre TP :

Table "Users" avec 3 colonnes : id, prenom, nom

Sequelize = traducteur qui transforme du JS en requêtes SQL :

```
// JS simple
const user = await User.create({ prenom: "Marie", nom: "Dupont" });

// Devient en SQL (Sequelize le fait pour toi) :
INSERT INTO Users (prenom, nom) VALUES ('Marie', 'Dupont');
```

---

2. API REST = Langage universel du web

Analogie :

Les 4 verbes HTTP (CRUD) :

Verbe HTTP	Action CRUD	Exemple dans TP	URL
`GET`	**Read** (Lire)	"Montre-moi tous les étudiants"	`GET /api/users`
`POST`	**Create** (Créer)	"Ajoute Marie Dupont"	`POST /api/users`
`DELETE`	**Delete** (Supprimer)	"Supprime l'étudiant #5"	`DELETE /api/users/5`
`PUT`	**Update** (Modifier)	"Change Marie → Sophie"	`PUT /api/users/5`

---

3. Comment tout s'articule ? (Flux complet)

```
Navigateur HTML/JS    ↔   Serveur Express   ↔   DB SQLite
       |                       |                    |
    [Bouton Ajouter]       [Route POST]        [Table Users]
       |                       |                    |
    1 Clic user            2 Reçoit JSON      3 INSERT nouvelle ligne
       ↓                       ↓                    ↓
    POST /api/users        User.create()      John Doe, Marie Dupont...
```

Exemple #1 : Ajouter un étudiant

```
//Frontend JS :
fetch('http://localhost:3000/api/users', {
  method: 'POST',
  body: JSON.stringify({ nom: 'Dupont', prenom: 'Marie' })
})

//Backend Express :
router.post('/', async (req, res) => {
  const user = await User.create(req.body);  // Magic Sequelize !
  res.json(user);
})
```

```
DB SQLite :
+----+--------+----------+
| id | prenom | nom      |
+----+--------+----------+
| 1  | John   | Doe      |  ← Déjà là
| 2  | Marie  | Dupont   |  ← NOUVEAU !
+----+--------+----------+
```

Exemple #2 : Supprimer John Doe

```
//Frontend JS :
fetch('http://localhost:3000/api/users/1', { method: 'DELETE' })

//Backend Express :
router.delete('/:id', async (req, res) => {
  await User.destroy({ where: { id: req.params.id } });
})
```

```
DB SQLite :
+----+--------+----------+
```

```
| id | prenom | nom      |
+----+--------+----------+
| 2  | Marie  | Dupont   |  ← John disparu !
+----+--------+----------+
```

---

4. Les 4 opérations CRUD expliquées

READ (GET) - Lire les données

Objectif : "Montre-moi TOUS les étudiants existants"

```
// Frontend
fetch('http://localhost:3000/api/users')  // Au chargement de la page
  .then(res => res.json())
  .then(users => {
    this.users = users;  // Remplit le tableau
    afficherListe();     // Affiche dans HTML
  });
```

Backend :

```
router.get('/', async (req, res) => {
  const users = await User.findAll();  // SELECT * FROM Users
  res.json(users);                     // → [{"id":1,"prenom":"John","nom":"Doe"}]
});
```

CREATE (POST) - Créer

```
Objectif : "Ajoute un nouvel étudiant dans la DB"
```

```
// Frontend - Clic "Ajouter"
fetch('/api/users', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({ nom: 'Martin', prenom: 'Paul' })
});
```

DELETE (DELETE) - Supprimer

Objectif : "Enlève l'étudiant #3 de la DB"

```
// Frontend - Clic bouton "X"
fetch(`/api/users/${userId}`, { method: 'DELETE' })
```

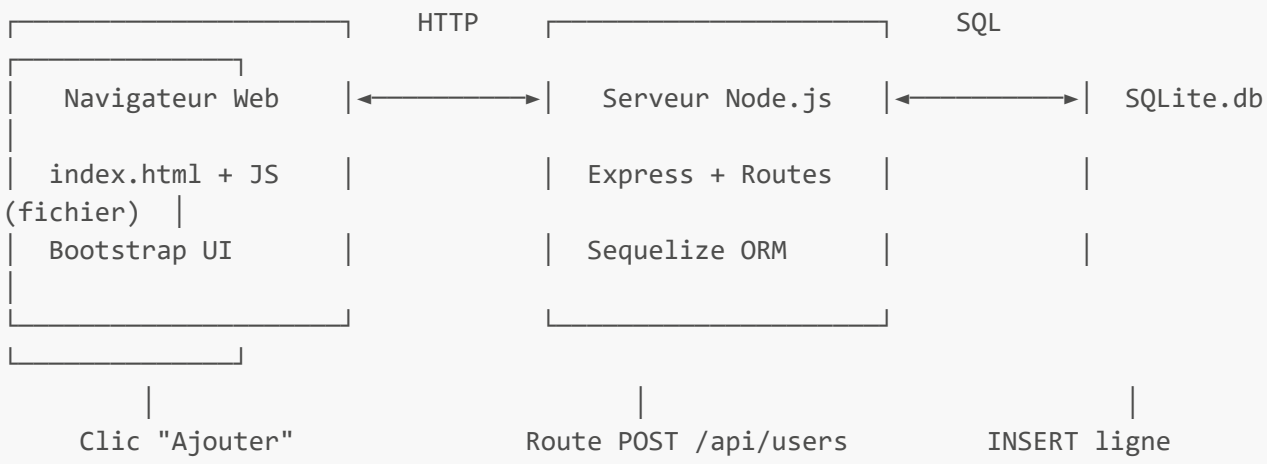
UPDATE (PUT) - Modifier *(pas dans TP mais utile)*

Objectif : "Change le nom de Paul → Pierre"

```
fetch(`/api/users/3`, {
  method: 'PUT',
  body: JSON.stringify({ nom: 'Pierre', prenom: 'Paul' })
});
```

---

5. Architecture de notre application



---

6. Pourquoi cette architecture ?

Problème	Sans DB	Avec DB + API
Fermer navigateur	Données perdues	Données sauvées
Ouvrir sur autre ordi	Rien	Données toujours là
Plusieurs users	Impossible	Tout le monde voit même liste
Backup	Rien à backup	1 fichier `.db`

---

7. Résumé des concepts clés

1. DB = Excel intelligent (SQLite = fichier unique)
2. API REST = 4 verbes magiques (GET=voir, POST=ajouter, DELETE=supprimer)
3. Frontend parle à Backend via fetch()

4. Backend traduit JS → SQL via Sequelize
5. Tout synchronisé temps réel (ajout/suppression immédiate)