
-
- `npm` `yarn`

1.

```
mkdir gestion-taches
cd gestion-taches
npm init -y
```

2.

```
npm install express graphql express-graphql
```

```
gestion-taches/
├── index.js
└── tasksData.js
```

`tasksData.js`

```
let tasks = [
  { id: 1, title: "Apprendre GraphQL", completed: false },
  { id: 2, title: "Créer un CRUD", completed: true },
];

module.exports = tasks;
```

`index.js`

```
const express = require("express");
const { graphqlHTTP } = require("express-graphql");
const { buildSchema } = require("graphql");
let tasks = require("./tasksData");

// 1. Définir le schéma GraphQL
const schema = buildSchema(`
  type Task {
    id: ID!
    title: String!
    completed: Boolean!
  }

  input TaskInput {
    title: String!
    completed: Boolean
  }

  type Query {
    getTasks: [Task]
    getTask(id: ID!): Task
  }

  type Mutation {
    createTask(input: TaskInput): Task
    updateTask(id: ID!, input: TaskInput): Task
```

```
    deleteTask(id: ID!): String
  }
`);

// 2. Résolveurs GraphQL
const root = {
  getTasks: () => tasks,
  getTask: ({ id }) => tasks.find((task) => task.id == id),
  createTask: ({ input }) => {
    const newTask = { id: tasks.length + 1,  ...input, completed: input.completed || false };
    tasks.push(newTask);
    return newTask;
  },
  updateTask: ({ id, input }) => {
    const index = tasks.findIndex((task) => task.id == id);
    if (index === -1) throw new Error("Task not found");
    tasks[index] = {  ...tasks[index],  ...input };
    return tasks[index];
  },
  deleteTask: ({ id }) => {
    const index = tasks.findIndex((task) => task.id == id);
    if (index === -1) throw new Error("Task not found");
    tasks.splice(index, 1);
    return `Task ${id} deleted`;
  },
};

// 3. Configurer Express avec GraphQL
const app = express();
app.use(
  "/graphql",
  graphqlHTTP({
    schema,
    rootValue: root,
    graphiql: true, // Interface pour tester les requêtes
  })
);

app.listen(4000, () => console.log("Serveur GraphQL sur http://localhost:4000/graphql"));
```

---

http://localhost:4000/graphql

```graphql
query {
  getTasks {
    id
    title
    completed
  }
}
```

```graphql
query {
  getTask(id: 1) {
    id
    title
    completed
  }
}
```

```graphql
mutation {
  createTask(input: { title: "Tester les mutations", completed: false }) {
    id
    title
    completed
  }
}
```

```graphql
mutation {
  updateTask(id: 1, input: { title: "Apprendre GraphQL en profondeur", completed: true }) {
    id
    title
    completed
  }
}
```

```
mutation {
  deleteTask(id: 2)
}
```

-
-
-

🚀

1.

   ```
   mkdir gestion-taches-apollo
   cd gestion-taches-apollo
   npm init -y
   ```

2.

   ```
   npm install @apollo/server graphql cors body-parser express
   ```

```
gestion-taches-apollo/
├── index.js
└── tasksData.js
```

`tasksData.js`

```
let tasks = [
  { id: "1", title: "Apprendre GraphQL", completed: false },
  { id: "2", title: "Créer un CRUD avec Apollo Server", completed: true },
];

module.exports = tasks;
```

`index.js`

```
const { ApolloServer } = require('@apollo/server');
const { expressMiddleware } = require('@apollo/server/express4');
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors');
let tasks = require('./tasksData');

// 1. Définir le schéma avec SDL
const typeDefs = `
  type Task {
    id: ID!
    title: String!
    completed: Boolean!
  }

  input TaskInput {
    title: String!
    completed: Boolean
  }
```

```
  type Query {
    getTasks: [Task]
    getTask(id: ID!): Task
  }

  type Mutation {
    createTask(input: TaskInput): Task
    updateTask(id: ID!, input: TaskInput): Task
    deleteTask(id: ID!): String
  }
`;

// 2. Définir les résolveurs
const resolvers = {
  Query: {
    getTasks: () => tasks,
    getTask: (_, { id }) => tasks.find((task) => task.id === id),
  },
  Mutation: {
    createTask: (_, { input }) => {
      const newTask = { id: `${tasks.length + 1}`, ...input, completed: input.completed || false };
      tasks.push(newTask);
      return newTask;
    },
    updateTask: (_, { id, input }) => {
      const taskIndex = tasks.findIndex((task) => task.id === id);
      if (taskIndex === -1) throw new Error('Task not found');
      tasks[taskIndex] = { ...tasks[taskIndex], ...input };
      return tasks[taskIndex];
    },
    deleteTask: (_, { id }) => {
      const taskIndex = tasks.findIndex((task) => task.id === id);
      if (taskIndex === -1) throw new Error('Task not found');
      tasks.splice(taskIndex, 1);
      return `Task ${id} deleted`;
    },
  },
};

// 3. Configurer Apollo Server
const server = new ApolloServer({ typeDefs, resolvers });

// 4. Intégrer Apollo avec Express
const app = express();
server.start().then(() => {
  app.use(cors(), bodyParser.json(), expressMiddleware(server));
```

```
  app.listen(4000, () => {
    console.log('🚀 Server ready at http://localhost:4000/');
  });
});
```

---

http://localhost:4000/

```
query {
  getTasks {
    id
    title
    completed
  }
}
```

```
query {
  getTask(id: "1") {
    id
    title
    completed
  }
}
```

```
mutation {
  createTask(input: { title: "Tester Apollo Server", completed: false }) {
    id
    title
    completed
  }
}
```

```
mutation {
  updateTask(id: "1", input: { title: "Maîtriser GraphQL avec Apollo", completed: true }) {
    id
    title
    completed
  }
}
```

```
mutation {
  deleteTask(id: "2")
}
```

1.
2.
3.

🚀

`import/export` `arrow functions`

`import/export`

1.
2. `"type": "module"` `package.json`

```
{
  "name": "gestion-taches-apollo",
  "version": "1.0.0",
  "type": "module",
```

```
  "main": "index.js",
  "dependencies": {
    "@apollo/server": "^4.3.0",
    "cors": "^2.8.5",
    "body-parser": "^1.20.2",
    "express": "^4.18.2",
    "graphql": "^16.6.0"
  }
}
```

```
gestion-taches-apollo/
├── index.js        // Point d'entrée
└── tasksData.js    // Données simulées
```

---

tasksData.js

```
export let tasks = [
  { id: "1", title: "Apprendre GraphQL avec ES6", completed: false },
  { id: "2", title: "Utiliser Apollo Server en mode moderne", completed: true },
];

// Fonction pour ajouter, mettre à jour, ou supprimer une tâche
export const updateTasks = (newTasks) => {
  tasks = newTasks;
};
```

---

index.js

```
import { ApolloServer } from "@apollo/server";
import { expressMiddleware } from "@apollo/server/express4";
import express from "express";
import bodyParser from "body-parser";
import cors from "cors";
import { tasks, updateTasks } from "./tasksData.js";

// 1. Schéma GraphQL
const typeDefs = `
  type Task {
    id: ID!
    title: String!
    completed: Boolean!
  }

  input TaskInput {
    title: String!
    completed: Boolean
  }

  type Query {
    getTasks: [Task]
    getTask(id: ID!): Task
  }

  type Mutation {
    createTask(input: TaskInput): Task
    updateTask(id: ID!, input: TaskInput): Task
    deleteTask(id: ID!): String
  }
`;

// 2. Résolveurs avec des fonctions fléchées
const resolvers = {
  Query: {
    getTasks: () => tasks,
    getTask: (_, { id }) => tasks.find((task) => task.id === id),
  },
  Mutation: {
    createTask: (_, { input }) => {
      const newTask = {
        id: `${tasks.length + 1}`,
        ...input,
        completed: input.completed || false,
      };
      tasks.push(newTask);
```

```
      return newTask;
    },
    updateTask: (_, { id, input }) => {
      const index = tasks.findIndex((task) => task.id === id);
      if (index === -1) throw new Error("Task not found");
      tasks[index] = { ...tasks[index], ...input };
      return tasks[index];
    },
    deleteTask: (_, { id }) => {
      const index = tasks.findIndex((task) => task.id === id);
      if (index === -1) throw new Error("Task not found");
      const deletedTask = tasks[index];
      updateTasks(tasks.filter((task) => task.id !== id));
      return `Task "${deletedTask.title}" deleted`;
    },
  },
};

// 3. Configurer Apollo Server
const server = new ApolloServer({ typeDefs, resolvers });

// 4. Intégrer Apollo avec Express
const app = express();
server.start().then(() => {
  app.use(cors(), bodyParser.json(), expressMiddleware(server));
  app.listen(4000, () =>
    console.log("🚀 Apollo Server prêt sur http://localhost:4000/")
  );
});
```

- `import/export` `require` `module.exports`
-
-
- `updateTasks`

---

http://localhost:4000/

1.
2.
3. `tasksData.js`

🚀