

# Chapitre 3

## API avec le module HTTP

# Rappels : http

- Node.js embarque un module natif nommé http qui permet de créer un serveur web.
- Il écoute des requêtes HTTP (GET, POST, etc.) et renvoie une réponse.

```
Uploaded using RayThis Extension

const http = require('http');

const server = http.createServer((req, res) => {
    res.statusCode
    res.setHeader
    res.end('Hello World');
});

server.listen(3000, () => {
    console.log('Server running at http://localhost:3000/');
});
```

# Gestion des routes

- On analyse `req.url` (l'URL) et `req.method` (méthode HTTP) pour router les actions.

```
Uploaded using RayThis Extension

const server = http.createServer((req, res) => {
  res.setHeader('Content-Type', 'application/json');

  if (req.url === '/api/users' && req.method === 'GET') {
    const users = [
      { id: 1, name: 'Alice' },
      { id: 2, name: 'Bob' }
    ];
    return res.end(JSON.stringify(users));
  }

  res.writeHead(404);
  res.end(JSON.stringify({ message: 'Route non trouvée' }));
});
```

# Récupérer les données envoyées (POST JSON)

- On doit écouter la réception du corps `req.on('data')`



Uploaded using RayThis Extension

```
const server = http.createServer((req, res) => {
  res.setHeader('Content-Type', 'application/json');

  if (req.url === '/api/users' && req.method === 'POST') {
    let body = '';

    req.on('data', chunk => {
      body += chunk.toString(); // conversion string
    });

    req.on('end', () => {
      const parsedData = JSON.parse(body);
      users.push(parsedData);

      res.writeHead(201);
      res.end(JSON.stringify({ message: 'Utilisateur ajouté', users }));
    });
  } else {
    res.writeHead(404);
    res.end(JSON.stringify({ message: 'Not found' }));
  });
});
```

# Tester avec un outil (ex postman)

- Méthode POST
- Route <http://localhost:3000/api/users>
- Contenu {"id":3,"name":"Charlie"}



# **Exercices**

# Stockage dans un fichier JSON

- Nouvelle structure :
  - /server.js
  - /users.json
- User.json contient []
- Import des modules indispensables dans server.js

```
Upgraded using RayThis Extension

const http = require('http');
const fs = require('fs');
const path = require('path');

const dataPath = path.join(__dirname, 'users.json');
```

# Ajout des fonctions utilitaires aux données

```
function readUsers() {  
    const data = fs.readFileSync(dataPath);  
    return JSON.parse(data);  
}  
  
function writeUsers(data) {  
    fs.writeFileSync(dataPath, JSON.stringify(data, null, 2));  
}
```

# CRUD : GET et GET ID

GET

```
const server = http.createServer((req, res) => {
  res.setHeader('Content-Type', 'application/json');

  if (req.url === '/api/users' && req.method === 'GET') {
    const users = readUsers();
    return res.end(JSON.stringify(users));
  }

  res.writeHead(404);
  res.end(JSON.stringify({ message: 'Route non trouvée' }));
});
```

GET ID

```
if (req.url.match('/api/users/([0-9]+)/') && req.method === 'GET') {
  const id = req.url.split('/')[3];
  const users = readUsers();
  const user = users.find(u => u.id === id);

  if (!user) {
    res.writeHead(404);
    return res.end(JSON.stringify({ message: 'Utilisateur introuvable' }));
  }

  return res.end(JSON.stringify(user));
}
```

# CRUD : POST et DELETE

POST

```
if (req.url === '/api/users' && req.method === 'POST') {
  let body = '';

  req.on('data', chunk => {
    body += chunk.toString();
  });

  req.on('end', () => {
    const newUser = JSON.parse(body);
    const users = readUsers();
    users.push(newUser);
    writeUsers(users);

    res.writeHead(201);
    res.end(JSON.stringify({ message: 'Utilisateur ajouté', newUser }));
  });
  return;
}
```

DELETE

```
if (req.url.match('/api/users/([0-9]+)') && req.method === 'DELETE') {
  const id = req.url.split('/')[3];
  let users = readUsers();
  const filteredUsers = users.filter(u => u.id != id);

  if (filteredUsers.length === users.length) {
    res.writeHead(404);
    return res.end(JSON.stringify({ message: 'Utilisateur introuvable' }));
  }

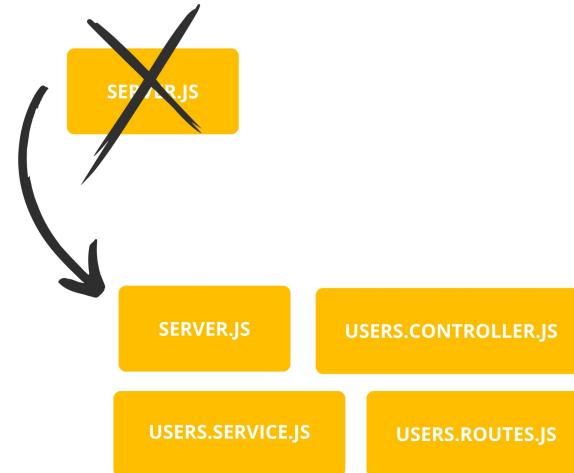
  writeUsers(filteredUsers);
  return res.end(JSON.stringify({ message: 'Utilisateur supprimé' }));
}
```



# **Exercices**

# Problématique : un seul fichier

- Server.js s'occupe de toutes les fonctionnalités
- Difficilement maintenable
- Aucune séparation des responsabilités

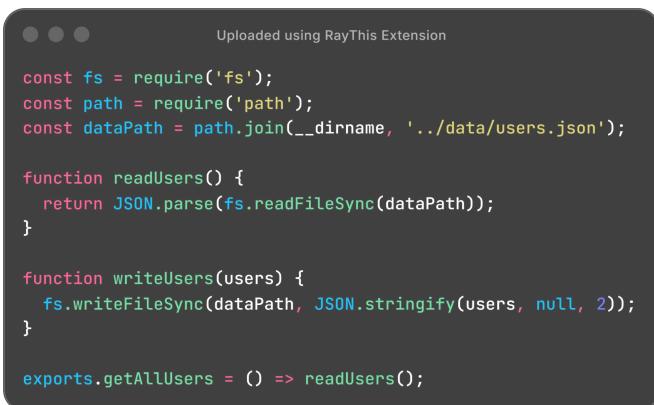


# Séparation des responsabilités

- server.js : point d'entrée
- users.controller.js : Récupère requête et appel services
- users.service.js : logique métier
- users.routes.js : redirection (routing API)

# Le service

- Uniquement la logique métier



```
Uploaded using RayThis Extension

const fs = require('fs');
const path = require('path');
const dataPath = path.join(__dirname, '../data/users.json');

function readUsers() {
  return JSON.parse(fs.readFileSync(dataPath));
}

function writeUsers(users) {
  fs.writeFileSync(dataPath, JSON.stringify(users, null, 2));
}

exports.getAllUsers = () => readUsers();
```



```
Uploaded using RayThis Extension

exports.getUserId = (id) => {
  const users = readUsers();
  return users.find(u => u.id == id);
};

exports.addUser = (user) => {
  const users = readUsers();
  users.push(user);
  writeUsers(users);
  return user;
};

exports.deleteUser = (id) => {
  const users = readUsers();
  const updated = users.filter(u => u.id != id);
  writeUsers(updated);
  return updated.length < users.length;
};
```

# Le contrôleur

- Gestion des requêtes et appels des bonnes méthodes

Uploaded using RayThis Extension

```
const service = require('../services/users.service');

exports.getUsers = (req, res) => {
  const users = service.getAllUsers();
  res.end(JSON.stringify(users));
};

exports.getUser = (req, res, id) => {
  const user = service.getUserById(id);
  if (!user) {
    res.writeHead(404);
    return res.end(JSON.stringify({ message: 'Not found' }));
  }
  res.end(JSON.stringify(user));
};
```

Uploaded using RayThis Extension

```
exports.createUser = (req, res) => {
  let body = '';
  req.on('data', chunk => body += chunk);
  req.on('end', () => {
    const user = JSON.parse(body);
    service.addUser(user);
    res.writeHead(201);
    res.end(JSON.stringify(user));
  });
};

exports.deleteUser = (req, res, id) => {
  const ok = service.deleteUser(id);
  if (!ok) {
    res.writeHead(404);
    return res.end(JSON.stringify({ message: 'Not found' }));
  }
  res.end(JSON.stringify({ message: 'Deleted' }));
};
```

# Le routeur

- En fonction des urls, appelle les bonnes méthodes du contrôleur.

```
Uploaded using RayThis Extension

const ctrl = require('../controllers/users.controller');

module.exports = (req, res) => {
  const url = new URL(req.url, `http://${req.headers.host}`);
  const path = url.pathname;

  res.setHeader('Content-Type', 'application/json');

  if (path === '/api/users' && req.method === 'GET')
    return ctrl.getUsers(req, res);

  if (path === '/api/users' && req.method === 'POST')
    return ctrl.createUser(req, res);

  const match = path.match(/^\/api\/users\/(\d+)/);
  if (match) {
    const id = match[1];

    if (req.method === 'GET')
      return ctrl.getUser(req, res, id);

    if (req.method === 'DELETE')
      return ctrl.deleteUser(req, res, id);
  }
};
```

# server.js

- Bien plus allégé et lisible

```
Uploaded using RayThis Extension

const http = require('http');
const userRoutes = require('./routes/users.routes');

const server = http.createServer((req, res) => {
  if (req.url.startsWith('/api/users'))
    return userRoutes(req, res);

  // if ... /api/tasks
  // return tasksRoutes .. par exemple

  res.writeHead(404);
  res.end(JSON.stringify({ message: '404 Not Found' }));
});

server.listen(3000, () => {
  console.log('Serveur HTTP Node.js en écoute sur http://localhost:3000');
});
```

# **Exercices**



# TP 2 : Validation des acquis