Exemplo de Microserviço: Gerenciamento de Usuários

Estrutura do Projeto

```
plaintext
Copiar código
user-service/
├── app.js            # Configuração do servidor
├── routes/
│   └── users.js      # Definição das rotas de usuários
├── controllers/
│   └── userController.js  # Lógica do negócio
└── data/
    └── users.json    # Banco de dados em JSON (para simplificação)
```

Passo 1: Instale as dependências

```
bash
Copiar código
mkdir user-service
cd user-service
npm init -y
npm install express
```

Passo 2: Crie o arquivo `app.js`

Esse arquivo inicializa o servidor e define o roteamento principal.

```
javascript
Copiar código
// app.js
const express = require('express');
const userRoutes = require('./routes/users');
const app = express();
const PORT = 3000;

// Middleware para interpretar JSON
app.use(express.json());

// Rota principal para o serviço de usuários
app.use('/users', userRoutes);

// Inicia o servidor
app.listen(PORT, () => {
  console.log(`User service is running on port ${PORT}`);
});
```

Passo 3: Crie o arquivo `data/users.json`

Esse será um banco de dados fictício usando um arquivo JSON.

```
json
Copiar código
[
  { "id": 1, "name": "Alice", "email": "alice@example.com" },
  { "id": 2, "name": "Bob", "email": "bob@example.com" }
]
```

Passo 4: Crie o arquivo `controllers/userController.js`

Esse arquivo contém a lógica do negócio, incluindo as operações de CRUD.

```
javascript
Copiar código
// controllers/userController.js
const fs = require('fs');
const path = require('path');
const usersFilePath = path.join(__dirname, '../data/users.json');

// Função para ler os usuários do arquivo JSON
function readUsers() {
  const usersData = fs.readFileSync(usersFilePath);
  return JSON.parse(usersData);
}

// Função para escrever os usuários no arquivo JSON
function writeUsers(users) {
  fs.writeFileSync(usersFilePath, JSON.stringify(users, null, 2));
}

// Controlador de criação de usuário
exports.createUser = (req, res) => {
  const users = readUsers();
  const newUser = {
    id: users.length + 1,
    ...req.body,
  };
  users.push(newUser);
  writeUsers(users);
  res.status(201).json(newUser);
};

// Controlador para listar todos os usuários
exports.getAllUsers = (req, res) => {
  const users = readUsers();
  res.json(users);
};

// Controlador para obter um usuário específico por ID
exports.getUserById = (req, res) => {
  const users = readUsers();
  const user = users.find((u) => u.id === parseInt(req.params.id));
  if (user) {
    res.json(user);
  } else {
    res.status(404).send('User not found');
  }
};

// Controlador para atualizar um usuário
exports.updateUser = (req, res) => {
  const users = readUsers();
  const userIndex = users.findIndex((u) => u.id ===
parseInt(req.params.id));
  if (userIndex !== -1) {
    users[userIndex] = { ...users[userIndex], ...req.body };
    writeUsers(users);
    res.json(users[userIndex]);
```

```
  } else {
    res.status(404).send('User not found');
  }
};

// Controlador para deletar um usuário
exports.deleteUser = (req, res) => {
  let users = readUsers();
  users = users.filter((u) => u.id !== parseInt(req.params.id));
  writeUsers(users);
  res.status(204).send();
};
```

Passo 5: Crie o arquivo `routes/users.js`

Defina as rotas do serviço para chamar os controladores.

```
javascript
Copiar código
// routes/users.js
const express = require('express');
const router = express.Router();
const userController = require('../controllers/userController');

// Rotas de CRUD para o usuário
router.post('/', userController.createUser);
router.get('/', userController.getAllUsers);
router.get('/:id', userController.getUserById);
router.put('/:id', userController.updateUser);
router.delete('/:id', userController.deleteUser);

module.exports = router;
```

Executando o Microserviço

1. No terminal, inicie o servidor:

   ```
   bash
   Copiar código
   node app.js
   ```

2. Teste as rotas usando o Postman, `curl` ou outra ferramenta.

Exemplos de Requisição

1. **Criar um novo usuário**:

   ```
   bash
   Copiar código
   curl -X POST http://localhost:3000/users -H "Content-Type:
   application/json" -d '{"name": "Charlie", "email":
   "charlie@example.com"}'
   ```

2. **Listar todos os usuários**:

   ```
   bash
   Copiar código
   ```

```
curl http://localhost:3000/users
```

3. **Obter um usuário específico**:

```
bash
Copiar código
curl http://localhost:3000/users/1
```

4. **Atualizar um usuário**:

```
bash
Copiar código
curl -X PUT http://localhost:3000/users/1 -H "Content-Type:
application/json" -d '{"name": "Alice Updated", "email":
"alice_updated@example.com"}'
```

5. **Deletar um usuário**:

```
bash
Copiar código
curl -X DELETE http://localhost:3000/users/1
```

Como esse Microserviço Funciona

Este microserviço é autônomo e serve para gerenciar os dados de usuários. Ele pode ser implementado como parte de uma arquitetura de microsserviços, onde outros microserviços (como autenticação, pedidos, ou pagamento) podem se comunicar com ele através de APIs para realizar operações relacionadas a usuários.