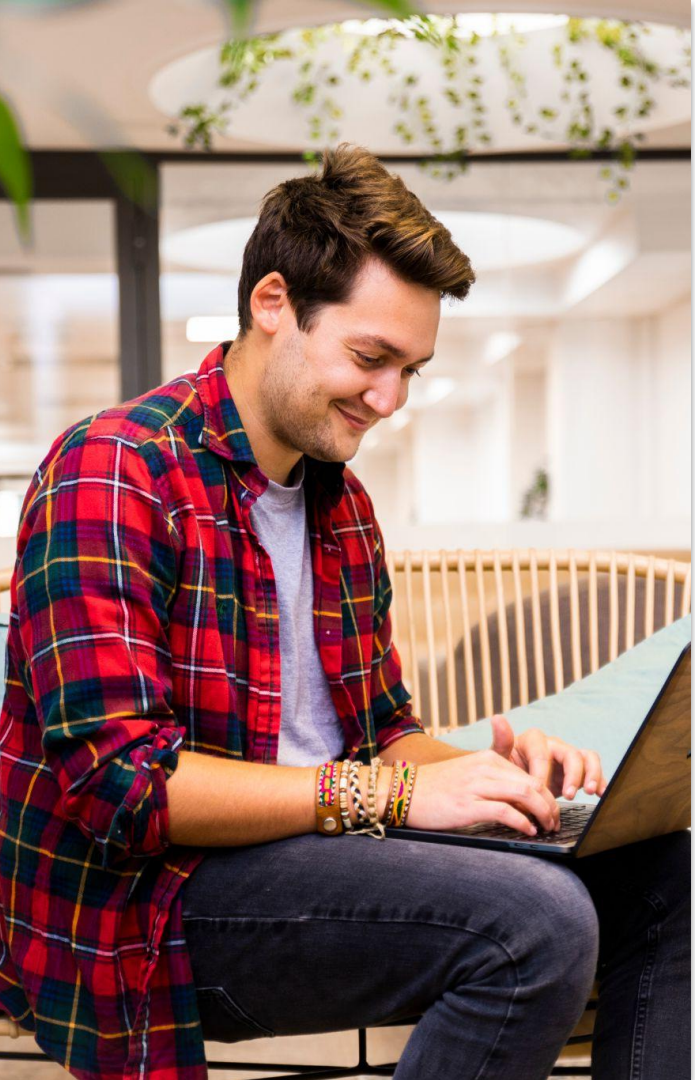


# Next.js Backend

# Contexte

## Le même principe de routes

Exactement comme **Express**, vous pouvez créer des routes avec l'API routes de **NextJS**

Et pour ce faire, vous devrez encore passer par ces 3 étapes :

| | |
|---|---|
| **1** | Créer la connexion à votre **BDD** (**MongoDB** via **Mongoose**) |
| **2** | Créer les fichiers Model (**Mongoose**) |
| **3** | Créer les routes |

# Step-by-step

# Contexte

## Créer la connexion à la BDD

```
yarn install mongoose
```

| | |
|---|---|
| 1 | Créer un fichier .env.local à la racine avec sa **CONNECTION_STRING** |
| 2 | Créer le dossier models à la racine et le fichier de connexion optimisé dedans |

# Contexte

## Créer les Modèles

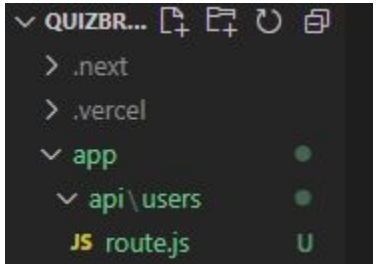
/models/User.js

```
import mongoose from "mongoose";

const User = mongoose.Schema(
    {
        name: String,
        age: Number,
        birthday: Date,
    }
);

export default mongoose.model("User", User);
```

# Contexte

## Créer les routes

| | |
|---|---|
| 1 | Dans le dossier **app**, créer un dossier **api** |
| 2 | Dans le dossier **api**, créer un dossier **users** par exemple |
| 3 | Dans le dossier **users** créer un fichier **route.js** |



# Contexte

## Préparer les routes

| | |
|---|---|
| 1 | Dans le fichier **route.js**, importez le model **User** et le fichier **connectDB** |
| 2 | Dans ce même fichier, importer **NextResponse** (remplacera le **res** d'Express) |

/api/users/route.js

```
import { NextResponse } from "next/server";
import User from "@/models/User";
import connectDB from "@/models/connectDB";
```

# Contexte

## Créer les routes

| | |
|---|---|
| 1 | Dans le fichier **route.js**, importez le model **User** et le fichier **connectDB** |
| 2 | Dans ce même fichier, importer **NextResponse** (remplacera le **res** d'Express) |

/api/users/route.js

```
import { NextResponse } from "next/server";
import User from "@/models/User";
import connectDB from "@/models/connectDB";
```

# Contexte

## Créer les routes

1. Créer une fonction par méthode que vous souhaitez (GET, POST, PUT, DELETE, PATCH…)

2. Rendez-là **asynchrone**, **exportez là** et ajoutez-y la connexion à la BDD si nécessaire

/api/users/route.js

```
export async function GET() {
    await connectDB();
    // ...
}

export async function POST(req) {
    await connectDB();
    // ...
}
```

## Contexte

## Exemple de Route GET

```
export async function GET() {
    await connectDB();
      try {
        const users = await User.find().sort({ age: "desc" });
        return NextResponse.json({ users });
    } catch (error) {
        console.error(error);
        return NextResponse.json(
            { message: 'Internal server error', error: true },
            { status: 500 }
        );
    }
}
```

👋 Vous venez de créer la route **GET** de **http://localhost:3000/api/users**

## Contexte

## Exemple de Route POST

```
export async function POST(req) {
    await connectDB();
    const data = await req.json();
    const newUser = new User({
        name: data.name,
        age: data.age,
        birthday: new Date()
    })
    const save = await newUser.save();
    return NextResponse.json({ result: true, message: User bien enregistré', save });
}
```

👋 Vous venez de créer la route **POST** de **http://localhost:3000/api/users**

brice-eliasse.com