**2**

**Étapes pour finaliser le backend**

**1. Organisation du projet**

Assure-toi d'avoir une bonne structure du backend :

bash

Copier le code

/ChatApp

├── server.js # Point d'entrée du serveur

├── /routes # Routes API

│ └── messages.js # Gestion des routes de messagerie

├── /models # Modèles pour la base de données

│ └── Message.js # Modèle Message

├── /middlewares # Middlewares (auth, logs...)

└── /config # Fichiers de configuration (DB, environnement)

**2. Configuration de la Base de Données (Exemple : MongoDB)**

**Installer Mongoose** :

bash

Copier le code

npm install mongoose

**Connexion à MongoDB dans server.js** :

javascript

Copier le code

const mongoose = require('mongoose');

mongoose.connect('mongodb://localhost:27017/chatApp', {

useNewUrlParser: true,

useUnifiedTopology: true,

})

.then(() => console.log('Connexion MongoDB réussie !'))

.catch((err) => console.error('Erreur de connexion à MongoDB :', err));

**3. Création du Modèle de Message**

Dans /models/Message.js :

javascript

Copier le code

const mongoose = require('mongoose');

const messageSchema = new mongoose.Schema({

sender: { type: String, required: true },

content: { type: String, required: true },

timestamp: { type: Date, default: Date.now },

});

module.exports = mongoose.model('Message', messageSchema);

**4. Création des Routes API**

**Installation d’Express** (si ce n’est pas déjà fait) :

bash

Copier le code

npm install express

Dans /routes/messages.js :

javascript

Copier le code

const express = require('express');

const Message = require('../models/Message');

const router = express.Router();

// Route pour récupérer tous les messages

router.get('/', async (req, res) => {

try {

const messages = await Message.find();

res.json(messages);

} catch (error) {

res.status(500).json({ message: 'Erreur serveur' });

}

});

// Route pour envoyer un message

router.post('/', async (req, res) => {

const { sender, content } = req.body;

const message = new Message({ sender, content });

try {

await message.save();

res.status(201).json(message);

} catch (error) {

res.status(500).json({ message: 'Erreur lors de l\'envoi du message' });

}

});

module.exports = router;

**5. Configuration du Serveur Node.js**

Dans server.js :

javascript

Copier le code

const express = require('express');

const http = require('http');

const socketIo = require('socket.io');

const mongoose = require('mongoose');

const messageRoutes = require('./routes/messages');

const cors = require('cors'); // Pour permettre l’accès au frontend

const app = express();

const server = http.createServer(app);

const io = socketIo(server);

// Middleware

app.use(cors());

app.use(express.json());

// Routes API

app.use('/api/messages', messageRoutes);

// Événements Socket.IO (communication en temps réel)

io.on('connection', (socket) => {

console.log('Nouvel utilisateur connecté');

socket.on('sendMessage', (message) => {

io.emit('receiveMessage', message); // Diffuser le message à tous les clients

});

socket.on('disconnect', () => {

console.log('Utilisateur déconnecté');

});

});

// Lancement du serveur

const PORT = process.env.PORT || 3000;

server.listen(PORT, () => console.log(`Serveur démarré sur le port ${PORT}`));

**6. Sécurisation du Backend**

* **Helmet** pour sécuriser les headers HTTP :

bash

Copier le code

npm install helmet

Ajoute-le dans server.js :

javascript

Copier le code

const helmet = require('helmet');

app.use(helmet());

* **bcrypt.js** pour hacher les mots de passe (si tu ajoutes des utilisateurs) :

bash

Copier le code

npm install bcrypt

Exemple d’utilisation :

javascript

Copier le code

const bcrypt = require('bcrypt');

const hashedPassword = await bcrypt.hash(password, 10);

* **Rate Limiter** pour prévenir les abus :

bash

Copier le code

npm install express-rate-limit

Exemple d’utilisation :

javascript

Copier le code

const rateLimit = require('express-rate-limit');

const limiter = rateLimit({

windowMs: 15 \* 60 \* 1000, // 15 minutes

max: 100, // Limite de 100 requêtes par IP

});

app.use(limiter);

**7. Tests du Backend**

* **Postman** ou **Insomnia** : Utilise ces outils pour tester tes routes (GET/POST).
* **Vérifie Socket.IO** : Ouvre plusieurs navigateurs ou onglets et teste l’envoi de messages en temps réel.

**8. Préparation pour le Déploiement**

* **Variables d'environnement** : Utilise un fichier .env pour stocker les informations sensibles :

bash

Copier le code

PORT=3000

MONGO\_URI=mongodb://localhost:27017/chatApp

Dans server.js :

javascript

Copier le code

require('dotenv').config();

const PORT = process.env.PORT;

const MONGO\_URI = process.env.MONGO\_URI;

mongoose.connect(MONGO\_URI, ...);

* **Déploiement sur Heroku** :
  1. Installe Heroku CLI :

bash

Copier le code

npm install -g heroku

* 1. Connecte-toi :

bash

Copier le code

heroku login

* 1. Initialise ton projet :

bash

Copier le code

heroku create chatapp-backend

git push heroku main

Avec tout ça, ton **backend sera complet et prêt à l’emploi** ! 🎯  
Si tu as des questions ou si tu rencontres des erreurs, je suis là pour t’aider ! 😊