Parfait, on recommence avec une **architecture AdonisJS (backend API) + Next.js (frontend)** avec **MySQL** comme base de données.

**🧱 OBJECTIF GLOBAL**

Tu veux construire :

* 🔐 Authentification (Admin / User)
* 👤 Gestion des utilisateurs (admin & user)
* 📁 Gestion des clients
* 🧭 API en **AdonisJS**
* 🎨 Interface frontend en **Next.js**
* 💾 Base de données **MySQL**

**⚙️ ÉTAPE 1 : BACKEND AVEC ADONISJS + MYSQL**

**✅ 1. Créer le projet AdonisJS**

npm init adonis-ts-app@latest adonis-backend

cd adonis-backend

🟡 Choisis l’option **"API server"**

**✅ 2. Installer Auth et configurer MySQL**

npm install @adonisjs/auth

node ace configure @adonisjs/auth

Dans .env, configure MySQL :

DB\_CONNECTION=mysql

MYSQL\_HOST=127.0.0.1

MYSQL\_PORT=3306

MYSQL\_USER=root

MYSQL\_PASSWORD=motdepasse

MYSQL\_DB\_NAME=nom\_de\_ta\_base

**✅ 3. Créer les modèles et migrations**

**📌 a. Utilisateur (Admin ou User)**

node ace make:model User -m

Dans users migration :

table.increments('id')

table.string('email').notNullable().unique()

table.string('password').notNullable()

table.enum('role', ['admin', 'user']).notNullable().defaultTo('user')

table.timestamps()

Dans app/Models/User.ts :

import { BaseModel, column } from '@adonisjs/lucid/orm'

export default class User extends BaseModel {

@column({ isPrimary: true })

public id: number

@column()

public email: string

@column({ serializeAs: null })

public password: string

@column()

public role: 'admin' | 'user'

}

**📌 b. Client**

node ace make:model Client -m

Dans la migration :

table.increments('id')

table.string('name').notNullable()

table.string('company')

table.string('email')

table.string('phone')

table.text('description')

table.integer('user\_id').unsigned().references('id').inTable('users').onDelete('CASCADE')

table.timestamps()

Dans Client.ts :

import { BaseModel, column, belongsTo, BelongsTo } from '@adonisjs/lucid/orm'

import User from './User'

export default class Client extends BaseModel {

@column({ isPrimary: true })

public id: number

@column()

public name: string

@column()

public company: string

@column()

public email: string

@column()

public phone: string

@column()

public description: string

@column()

public userId: number

@belongsTo(() => User)

public user: BelongsTo<typeof User>

}

**✅ 4. Lancer la migration**

node ace migration:run

**✅ 5. Créer les contrôleurs**

**🔐 AuthController**

node ace make:controller AuthController

// start/routes.ts

Route.post('/register', 'AuthController.register')

Route.post('/login', 'AuthController.login')

Route.get('/me', 'AuthController.me').middleware('auth')

Route.post('/logout', 'AuthController.logout').middleware('auth')

Dans AuthController.ts :

import type { HttpContext } from '@adonisjs/core/http'

import User from '#models/user'

export default class AuthController {

async register({ request, auth }: HttpContext) {

const user = await User.create(request.only(['email', 'password', 'role']))

const token = await auth.use('api').login(user)

return { user, token }

}

async login({ request, auth }: HttpContext) {

const { email, password } = request.only(['email', 'password'])

const token = await auth.use('api').attempt(email, password)

return token

}

async me({ auth }: HttpContext) {

return auth.user

}

async logout({ auth }: HttpContext) {

await auth.logout()

return { message: 'Déconnecté' }

}

}

**👤 UsersController**

node ace make:controller UsersController

// start/routes.ts

Route.resource('/users', 'UsersController').apiOnly().middleware({

'\*': ['auth']

})

Dans UsersController.ts (exemple) :

import type { HttpContext } from '@adonisjs/core/http'

import User from '#models/user'

export default class UsersController {

async index({ response }: HttpContext) {

const users = await User.all()

return response.json(users)

}

async show({ params, response }: HttpContext) {

const user = await User.find(params.id)

return user ? response.json(user) : response.status(404).json({ error: 'Utilisateur non trouvé' })

}

async destroy({ params, response }: HttpContext) {

const user = await User.find(params.id)

if (!user) return response.status(404).json({ error: 'Utilisateur non trouvé' })

await user.delete()

return { message: 'Utilisateur supprimé' }

}

}

**📁 ClientsController**

node ace make:controller ClientsController

// start/routes.ts

Route.resource('/clients', 'ClientsController').apiOnly().middleware({

'\*': ['auth']

})

import type { HttpContext } from '@adonisjs/core/http'

import Client from '#models/client'

export default class ClientsController {

async index({ auth }: HttpContext) {

return await Client.query().where('userId', auth.user!.id)

}

async store({ request, auth }: HttpContext) {

const data = request.only(['name', 'company', 'email', 'phone', 'description'])

return await Client.create({ ...data, userId: auth.user!.id })

}

async show({ params, auth, response }: HttpContext) {

const client = await Client.query()

.where('id', params.id)

.where('userId', auth.user!.id)

.first()

return client ? client : response.status(404).json({ error: 'Client non trouvé' })

}

async update({ request, params, auth, response }: HttpContext) {

const client = await Client.query()

.where('id', params.id)

.where('userId', auth.user!.id)

.first()

if (!client) return response.status(404).json({ error: 'Client non trouvé' })

client.merge(request.only(['name', 'company', 'email', 'phone', 'description']))

await client.save()

return client

}

async destroy({ params, auth, response }: HttpContext) {

const client = await Client.query()

.where('id', params.id)

.where('userId', auth.user!.id)

.first()

if (!client) return response.status(404).json({ error: 'Client non trouvé' })

await client.delete()

return { message: 'Client supprimé' }

}

}

**⚛️ ÉTAPE 2 : FRONTEND AVEC NEXT.JS**

**✅ 1. Créer le projet Next.js**

npx create-next-app@latest frontend-app

cd frontend-app

**✅ 2. Installer Axios**

npm install axios

**✅ 3. Configuration Axios**

// lib/api.ts

import axios from 'axios'

const api = axios.create({

baseURL: 'http://localhost:3333',

withCredentials: true,

})

export default api

**✅ 4. Exemple de page Login**

// pages/login.tsx

import { useState } from 'react'

import api from '@/lib/api'

export default function Login() {

const [email, setEmail] = useState('')

const [password, setPassword] = useState('')

const handleLogin = async () => {

try {

const response = await api.post('/login', { email, password })

localStorage.setItem('token', response.data.token.token)

alert('Connecté avec succès')

} catch (err) {

alert('Erreur de connexion')

}

}

return (

<div>

<h1>Connexion</h1>

<input type="email" value={email} onChange={(e) => setEmail(e.target.value)} />

<input type="password" value={password} onChange={(e) => setPassword(e.target.value)} />

<button onClick={handleLogin}>Se connecter</button>

</div>

)

}

**🧪 Tu veux aller plus loin ?**

* Ajoute les middlewares adminOnly si tu veux protéger des routes (ex : UsersController accessible que par admin).
* Utilise JWT pour sécuriser les requêtes côté frontend.
* Connecte Next.js à Adonis

en injectant le token dans l’en-tête Authorization: Bearer <token>.

Souhaites-tu que je t’aide à faire un **squelette complet du projet** (avec fichiers et code pré-rempli) ?