# Export de code

## C:/Users/rescue123/Documents/DEV/CRM/backend\.env

DB\_NAME=project\_solar  
DB\_USER=postgres  
DB\_PASSWORD=password  
DB\_HOST=localhost  
DB\_PORT=5432  
JWT\_SECRET=supersecret  
EMAIL\_HOST=riz.o2switch.net  
EMAIL\_USER=projet@domsolaire.fr  
EMAIL\_PASSWORD=yazbZLKh?h?u  
EMAIL\_PORT=465  
PORT = 3000

## C:/Users/rescue123/Documents/DEV/CRM/backend\debug-index.txt

INDEX.JS EXECUTED !

## C:/Users/rescue123/Documents/DEV/CRM/backend\index.js

const fs = require('fs');  
fs.writeFileSync('debug-index.txt', 'INDEX.JS EXECUTED !');  
require('./src/app');

## C:/Users/rescue123/Documents/DEV/CRM/backend\package.json

{  
 "name": "carteprojet",  
 "version": "1.0.0",  
 "main": "index.js",  
 "scripts": {  
 "test": "echo \"Error: no test specified\" && exit 1"  
 },  
 "keywords": [],  
 "author": "",  
 "license": "ISC",  
 "description": "",  
 "dependencies": {  
 "axios": "^1.10.0",  
 "bcryptjs": "^3.0.2",  
 "cors": "^2.8.5",  
 "dotenv": "^17.2.0",  
 "express": "^5.1.0",  
 "jsonwebtoken": "^9.0.2",  
 "nodemailer": "^7.0.5",  
 "pg": "^8.16.3"  
 },  
 "devDependencies": {  
 "nodemon": "^3.1.10"  
 }  
}

## C:/Users/rescue123/Documents/DEV/CRM/backend\src\app.js

const express = require('express');  
const cors = require('cors');  
require('dotenv').config();  
  
const fs = require('fs');  
const path = require('path');  
  
// === Journalisation dans un fichier backend/logs/debug.log ===  
function logToFile(message) {  
 const logDir = path.join(\_\_dirname, '..', 'logs');  
 const logFile = path.join(logDir, 'debug.log');  
 try {  
 if (!fs.existsSync(logDir)) {  
 fs.mkdirSync(logDir);  
 }  
 fs.appendFileSync(logFile, `[${new Date().toISOString()}] ${message}\n`);  
 } catch (e) {  
 // On ignore les erreurs de log (ne doit pas bloquer l'app)  
 }  
}  
// Log de démarrage  
logToFile('==== BACKEND SOLAR APP DEMARRÉ ====');  
  
const userRoutes = require('./routes/userRoutes');  
const projectRoutes = require('./routes/projectRoutes');  
const publicRoutes = require('./routes/publicRoutes');  
const invitationRoutes = require('./routes/invitationRoutes');  
const projectStepRoutes = require('./routes/projectStepRoutes');  
const investorRoutes = require('./routes/investorRoutes');  
const dashboardRoutes = require('./routes/dashboardRoutes');  
const app = express();  
app.use(cors());  
app.use(express.json());  
  
// Log à chaque requête reçue  
app.use((req, res, next) => {  
 logToFile(`Reçu: ${req.method} ${req.originalUrl}`);  
 next();  
});  
  
// Routes  
app.use('/api/users', userRoutes);  
app.use('/api/projects', projectRoutes);  
app.use('/api/public', publicRoutes);  
app.use('/api/projects', invitationRoutes);  
app.use('/api/projects', projectStepRoutes);  
app.use('/api', investorRoutes); // une seule base /api pour ce module  
app.use("/api", dashboardRoutes);  
  
// Route de test pour le ping  
app.get('/api/ping', (req, res) => {  
 logToFile('/api/ping appelé');  
 res.json({ pong: true, now: new Date().toISOString() });  
});  
  
// Page d'accueil  
app.get('/', (req, res) => {  
 logToFile('Route / appelée');  
 res.send('API Project Solar ready!');  
});  
  
// Gestion des erreurs globales  
app.use((err, req, res, next) => {  
 logToFile('Erreur serveur: ' + err.stack);  
 res.status(500).json({ error: 'Erreur serveur : ' + err.message });  
});  
  
// Catch global des exceptions fatales (process)  
process.on('uncaughtException', err => {  
 logToFile('Uncaught Exception: ' + err.stack);  
});  
process.on('unhandledRejection', reason => {  
 logToFile('Unhandled Rejection: ' + reason);  
});  
  
const PORT = process.env.PORT || 'passenger';  
app.listen(PORT, () => {  
 const msg = `Backend running on port ${PORT}`;  
 console.log(msg);  
 logToFile(msg);  
});

## C:/Users/rescue123/Documents/DEV/CRM/backend\src\controllers\dashboardController.js

const pool = require("../utils/db");  
  
// Jalons à venir  
async function getUpcomingMilestones(req, res) {  
 try {  
 const today = new Date();  
  
 const { rows } = await pool.query(`  
 SELECT   
 m.id,  
 m.label,  
 m.due\_date,  
 m.status,  
 p.id AS project\_id,  
 p.name AS project\_name,  
 i.name AS investor\_name  
 FROM investor\_milestones m  
 LEFT JOIN projects p ON p.id = m.project\_id  
 LEFT JOIN investors i ON i.id = m.investor\_id  
 WHERE m.due\_date >= $1  
 ORDER BY m.due\_date ASC  
 LIMIT 10  
 `, [today]);  
  
 res.json(rows);  
 } catch (err) {  
 console.error("Erreur getUpcomingMilestones:", err);  
 res.status(500).json({ error: "Erreur lors de la récupération des jalons à venir" });  
 }  
}  
  
// Activité récente (projets modifiés récemment)  
async function getRecentActivity(req, res) {  
 try {  
 const { rows } = await pool.query(`  
 SELECT   
 'Mise à jour du projet "' || name || '"' AS text,  
 updated\_at AS date  
 FROM projects  
 ORDER BY updated\_at DESC  
 LIMIT 5  
 `);  
  
 res.json(rows);  
 } catch (err) {  
 console.error("Erreur getRecentActivity:", err);  
 res.status(500).json({ error: "Erreur lors de la récupération de l'activité récente" });  
 }  
}  
async function getOverdueMilestones(req, res) {  
 try {  
 const query = `  
 SELECT   
 m.id,   
 m.label,   
 m.due\_date,   
 p.id AS project\_id,   
 p.name AS project\_name,   
 i.name AS investor\_name  
 FROM investor\_milestones m  
 JOIN projects p ON p.id = m.project\_id  
 LEFT JOIN investors i ON i.id = m.investor\_id  
 WHERE m.due\_date < NOW()  
 ORDER BY m.due\_date ASC  
 LIMIT 20;  
 `;  
  
 const { rows } = await pool.query(query);  
 res.json(rows);  
 } catch (err) {  
 console.error("Erreur getOverdueMilestones:", err);  
 res.status(500).json({ error: "Erreur lors de la récupération des jalons en retard" });  
 }  
}  
module.exports = {  
 getUpcomingMilestones,  
 getRecentActivity,  
 getOverdueMilestones  
};

## C:/Users/rescue123/Documents/DEV/CRM/backend\src\middleware\auth.js

const jwt = require('jsonwebtoken');  
require('dotenv').config();  
  
function authenticate(req, res, next) {  
 const token = req.header('Authorization')?.replace('Bearer ', '');  
 if (!token) return res.status(401).json({ error: 'No token provided' });  
  
 try {  
 const decoded = jwt.verify(token, process.env.JWT\_SECRET);  
 req.user = decoded;  
 next();  
 } catch (e) {  
 res.status(401).json({ error: 'Token invalide' });  
 }  
}  
  
function authorizeRole(role) {  
 return (req, res, next) => {  
 if (req.user.role !== role) {  
 return res.status(403).json({ error: 'Accès interdit' });  
 }  
 next();  
 };  
}  
  
module.exports = { authenticate, authorizeRole };

## C:/Users/rescue123/Documents/DEV/CRM/backend\src\models\investorMilestoneModel.js

const pool = require('../utils/db');  
  
// ─── Helpers ───────────────────────────────────────────────────────────────────  
const ALLOWED\_STATUS = new Set(['Payé', 'Pas payé', 'Annulé', 'En cours']);  
  
function buildUpdateSet(data) {  
 // on autorise label, due\_date, status, comment  
 const allowed = ['label', 'due\_date', 'status', 'comment'];  
 const fields = {};  
 for (const k of allowed) {  
 if (Object.prototype.hasOwnProperty.call(data, k)) {  
 fields[k] = data[k] === '' ? null : data[k];  
 }  
 }  
 const keys = Object.keys(fields);  
 if (!keys.length) return { sql: '', values: [] };  
  
 const sets = [];  
 const values = [];  
 let i = 1;  
 for (const k of keys) {  
 sets.push(`${k} = $${i++}`);  
 values.push(fields[k]);  
 }  
 // updated\_at si ta table l’a  
 sets.push(`updated\_at = NOW()`);  
 return { sql: sets.join(', '), values };  
}  
  
// ─── Create ────────────────────────────────────────────────────────────────────  
// INSERT : retourne la ligne créée (pour MAJ instantanée côté front)  
const addMilestone = async (project\_id, investor\_id, { label, due\_date, status, comment }) => {  
 if (status && !ALLOWED\_STATUS.has(status)) {  
 const err = new Error('Statut de jalon invalide');  
 err.code = 'BAD\_STATUS';  
 throw err;  
 }  
  
 const { rows } = await pool.query(  
 `INSERT INTO investor\_milestones (project\_id, investor\_id, label, due\_date, status, comment)  
 VALUES ($1, $2, $3, $4, $5, $6)  
 RETURNING \*`,  
 [project\_id, investor\_id, label, due\_date || null, status, comment || null]  
 );  
 return rows[0];  
};  
  
// ─── Read ──────────────────────────────────────────────────────────────────────  
const listMilestones = async (project\_id, investor\_id) => {  
 const { rows } = await pool.query(  
 `SELECT \* FROM investor\_milestones  
 WHERE project\_id = $1 AND investor\_id = $2  
 ORDER BY COALESCE(due\_date, created\_at) DESC, id DESC`,  
 [project\_id, investor\_id]  
 );  
 return rows;  
};  
  
const getMilestoneById = async (id) => {  
 const { rows } = await pool.query(`SELECT \* FROM investor\_milestones WHERE id = $1`, [id]);  
 return rows[0] || null;  
};  
  
// ─── Update ────────────────────────────────────────────────────────────────────  
// Update "scopé" (sécurité: vérifie l’appartenance au projet & investisseur)  
const updateMilestoneScoped = async (project\_id, investor\_id, id, data) => {  
 if (data.status && !ALLOWED\_STATUS.has(data.status)) {  
 const err = new Error('Statut de jalon invalide');  
 err.code = 'BAD\_STATUS';  
 throw err;  
 }  
  
 const { sql, values } = buildUpdateSet(data);  
 if (!sql) {  
 // rien à mettre à jour -> renvoyer la ligne courante  
 const cur = await getMilestoneById(id);  
 return cur;  
 }  
  
 const params = [...values, id, project\_id, investor\_id];  
 const { rows } = await pool.query(  
 `UPDATE investor\_milestones  
 SET ${sql}  
 WHERE id = $${values.length + 1} AND project\_id = $${values.length + 2} AND investor\_id = $${values.length + 3}  
 RETURNING \*`,  
 params  
 );  
 return rows[0] || null;  
};  
  
// Update par id (route "flat")  
const updateMilestoneById = async (id, data) => {  
 if (data.status && !ALLOWED\_STATUS.has(data.status)) {  
 const err = new Error('Statut de jalon invalide');  
 err.code = 'BAD\_STATUS';  
 throw err;  
 }  
  
 const { sql, values } = buildUpdateSet(data);  
 if (!sql) {  
 const cur = await getMilestoneById(id);  
 return cur;  
 }  
  
 const params = [...values, id];  
 const { rows } = await pool.query(  
 `UPDATE investor\_milestones  
 SET ${sql}  
 WHERE id = $${values.length + 1}  
 RETURNING \*`,  
 params  
 );  
 return rows[0] || null;  
};  
  
// ─── Delete ────────────────────────────────────────────────────────────────────  
const deleteMilestone = async (id) => {  
 await pool.query('DELETE FROM investor\_milestones WHERE id = $1', [id]);  
};  
  
module.exports = {  
 addMilestone,  
 listMilestones,  
 getMilestoneById,  
 updateMilestoneScoped,  
 updateMilestoneById,  
 deleteMilestone  
};

## C:/Users/rescue123/Documents/DEV/CRM/backend\src\models\investorModel.js

const pool = require('../utils/db');  
  
const createInvestor = async ({ name, company, email, phone, notes }) => {  
 const { rows } = await pool.query(  
 `INSERT INTO investors (name, company, email, phone, notes)  
 VALUES ($1,$2,$3,$4,$5) RETURNING \*`,  
 [name, company, email, phone, notes]  
 );  
 return rows[0];  
};  
  
const listInvestors = async () => {  
 const { rows } = await pool.query(`SELECT \* FROM investors ORDER BY name`);  
 return rows;  
};  
  
const getInvestor = async (id) => {  
 const { rows } = await pool.query(`SELECT \* FROM investors WHERE id=$1`, [id]);  
 return rows[0];  
};  
  
const updateInvestor = async (id, payload) => {  
 const { name, company, email, phone, notes } = payload;  
 const { rows } = await pool.query(  
 `UPDATE investors SET name=$1, company=$2, email=$3, phone=$4, notes=$5  
 WHERE id=$6 RETURNING \*`,  
 [name, company, email, phone, notes, id]  
 );  
 return rows[0];  
};  
  
const deleteInvestor = async (id) => {  
 await pool.query(`DELETE FROM investors WHERE id=$1`, [id]);  
};  
  
const attachInvestorToProject = async ({ project\_id, investor\_id, role=null }) => {  
 const { rows } = await pool.query(  
 `INSERT INTO project\_investors (project\_id, investor\_id, role)  
 VALUES ($1,$2,$3)  
 ON CONFLICT (project\_id, investor\_id) DO UPDATE SET role=EXCLUDED.role  
 RETURNING \*`,  
 [project\_id, investor\_id, role]  
 );  
 return rows[0];  
};  
  
const detachInvestorFromProject = async ({ project\_id, investor\_id }) => {  
 await pool.query(  
 `DELETE FROM project\_investors WHERE project\_id=$1 AND investor\_id=$2`,  
 [project\_id, investor\_id]  
 );  
};  
  
const listProjectInvestors = async (project\_id) => {  
 const { rows } = await pool.query(  
 `SELECT pi.id as link\_id, i.\*, pi.role  
 FROM project\_investors pi  
 JOIN investors i ON i.id = pi.investor\_id  
 WHERE pi.project\_id = $1  
 ORDER BY i.name`,  
 [project\_id]  
 );  
 return rows;  
};  
  
module.exports = {  
 createInvestor, listInvestors, getInvestor, updateInvestor, deleteInvestor,  
 attachInvestorToProject, detachInvestorFromProject, listProjectInvestors  
};

## C:/Users/rescue123/Documents/DEV/CRM/backend\src\models\invitationModel.js

const pool = require('../utils/db');  
const crypto = require('crypto');  
  
const createInvitation = async (project\_id, email) => {  
 const token = crypto.randomBytes(32).toString('hex');  
 const expires\_at = new Date(Date.now() + 1000 \* 60 \* 60 \* 24 \* 7); // 7 jours  
 const { rows } = await pool.query(  
 'INSERT INTO invitations (project\_id, email, token, expires\_at) VALUES ($1, $2, $3, $4) RETURNING \*',  
 [project\_id, email, token, expires\_at]  
 );  
 return rows[0];  
};  
  
const getInvitation = async (token) => {  
 const { rows } = await pool.query('SELECT \* FROM invitations WHERE token = $1', [token]);  
 return rows[0];  
};  
  
module.exports = { createInvitation, getInvitation };

## C:/Users/rescue123/Documents/DEV/CRM/backend\src\models\projectModel.js

const pool = require('../utils/db');  
const { geocodeAddress } = require('../utils/geocode');  
  
// Récupérer tous les projets  
const getAllProjects = async () => {  
 const { rows } = await pool.query('SELECT \* FROM projects');  
 return rows;  
};  
  
// Récupérer les projets par owner  
const getProjectsByOwnerId = async (owner\_id) => {  
 const { rows } = await pool.query('SELECT \* FROM projects WHERE owner\_id = $1', [owner\_id]);  
 return rows;  
};  
  
// Créer un projet  
const createProject = async (project) => {  
 const { name, address, latitude, longitude, type, power, phone, email, comments, owner\_id, status } = project;  
 const fixedOwnerId = owner\_id === "" ? null : owner\_id;  
 console.log("CREATE project (status):", status); // DEBUG  
 const { rows } = await pool.query(  
 `INSERT INTO projects   
 (name, address, latitude, longitude, type, power, phone, email, comments, owner\_id, status)  
 VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11)  
 RETURNING \*`,  
 [name, address, latitude || null, longitude || null, type, power, phone, email, comments, fixedOwnerId, status || 'En cours']  
 );  
 return rows[0];  
};  
  
const addCustomField = async (project\_id, field\_name, field\_value, user\_id = null) => {  
 const { rows } = await pool.query(  
 'INSERT INTO project\_custom\_fields (project\_id, field\_name, field\_value, added\_by) VALUES ($1, $2, $3, $4) RETURNING \*',  
 [project\_id, field\_name, field\_value, user\_id]  
 );  
 return rows[0];  
};  
  
const getCustomFields = async (project\_id) => {  
 const { rows } = await pool.query(  
 'SELECT \* FROM project\_custom\_fields WHERE project\_id = $1',  
 [project\_id]  
 );  
 return rows;  
};  
  
const updateCustomField = async (field\_id, field\_name, field\_value) => {  
 const { rows } = await pool.query(  
 'UPDATE project\_custom\_fields SET field\_name = $1, field\_value = $2 WHERE id = $3 RETURNING \*',  
 [field\_name, field\_value, field\_id]  
 );  
 return rows[0];  
};  
  
const deleteCustomField = async (field\_id) => {  
 await pool.query('DELETE FROM project\_custom\_fields WHERE id = $1', [field\_id]);  
};  
  
// Mettre à jour un projet  
const updateProject = async (id, project) => {  
 // --- LOG DÉTAILLÉ  
 console.log("updateProject() : id =", id);  
 console.log("Projet reçu par updateProject :", project);  
  
 const { name, address, latitude, longitude, type, power, phone, email, comments, owner\_id, status } = project;  
 console.log("Champs utilisés pour l'UPDATE :");  
 console.log({ name, address, latitude, longitude, type, power, phone, email, comments, owner\_id, status });  
  
 const params = [  
 name, address, latitude || null, longitude || null,  
 type, power, phone, email, comments, owner\_id, status || 'En cours', id  
 ];  
 console.log("Paramètres SQL envoyés :", params);  
  
 const { rows } = await pool.query(  
 `UPDATE projects SET   
 name=$1, address=$2, latitude=$3, longitude=$4, type=$5, power=$6, phone=$7, email=$8, comments=$9, owner\_id=$10, status=$11   
 WHERE id=$12 RETURNING \*`,  
 params  
 );  
 console.log("Résultat UPDATE :", rows[0]);  
 return rows[0];  
};  
  
// Supprimer un projet  
const deleteProject = async (id) => {  
 await pool.query('DELETE FROM projects WHERE id=$1', [id]);  
};  
  
const getProjectById = async (id) => {  
 const { rows } = await pool.query('SELECT \* FROM projects WHERE id=$1', [id]);  
 return rows[0];  
};  
  
module.exports = {   
 getAllProjects,   
 getProjectsByOwnerId,  
 createProject,   
 addCustomField,  
 getCustomFields, // <- Ajouté pour le front  
 updateProject,   
 deleteProject,   
 getProjectById,  
 addCustomField,  
 getCustomFields,  
 updateCustomField,  
 deleteCustomField  
};

## C:/Users/rescue123/Documents/DEV/CRM/backend\src\models\projectStepModel.js

const pool = require('../utils/db');  
  
const addStep = async (project\_id, label, step\_date, step\_comment, status) => {  
 const { rows } = await pool.query(  
 'INSERT INTO project\_steps (project\_id, label, step\_date, step\_comment, status) VALUES ($1, $2, $3, $4, $5) RETURNING \*',  
 [project\_id, label, step\_date, step\_comment, status]  
 );  
 return rows[0];  
};  
  
const getSteps = async (project\_id) => {  
 const { rows } = await pool.query(  
 'SELECT \* FROM project\_steps WHERE project\_id = $1 ORDER BY step\_date DESC',  
 [project\_id]  
 );  
 return rows;  
};  
const deleteStep = async (stepId) => {  
 await pool.query('DELETE FROM project\_steps WHERE id = $1', [stepId]);  
};  
module.exports = { addStep, getSteps, deleteStep };

## C:/Users/rescue123/Documents/DEV/CRM/backend\src\models\userModel.js

const pool = require('../utils/db');  
  
const getUserByEmail = async (email) => {  
 const { rows } = await pool.query('SELECT \* FROM users WHERE email = $1', [email]);  
 return rows[0];  
};  
  
const createUser = async (name, email, hash, role = 'user') => {  
 const { rows } = await pool.query(  
 'INSERT INTO users (name, email, password\_hash, role) VALUES ($1, $2, $3, $4) RETURNING \*',  
 [name, email, hash, role]  
 );  
 return rows[0];  
};  
const getAllUsers = async () => {  
 const { rows } = await pool.query('SELECT id, name, email, role FROM users');  
 return rows;  
};  
  
module.exports = { getUserByEmail, createUser, getAllUsers };

## C:/Users/rescue123/Documents/DEV/CRM/backend\src\routes\dashboardRoutes.js

const express = require("express");  
const { getUpcomingMilestones, getRecentActivity, getOverdueMilestones } = require("../controllers/dashboardController");  
  
const router = express.Router();  
  
router.get("/milestones/upcoming", getUpcomingMilestones);  
router.get("/activity/recent", getRecentActivity);  
router.get('/milestones/overdue', getOverdueMilestones);  
  
module.exports = router;

## C:/Users/rescue123/Documents/DEV/CRM/backend\src\routes\investorRoutes.js

// backend/src/routes/investorRoutes.js  
const express = require('express');  
const { authenticate } = require('../middleware/auth');  
const pool = require('../utils/db');  
  
// ⚠️ Assure-toi d'avoir mis à jour le model comme je t'ai donné (+update)  
const {  
 addMilestone,  
 listMilestones,  
 deleteMilestone,  
 updateMilestoneScoped,  
 updateMilestoneById,  
} = require('../models/investorMilestoneModel');  
  
const router = express.Router();  
  
/\*\*  
 \* NOTE : ce fichier suppose que tu as déjà :  
 \* - table investors (id, name, company, email, phone, notes, ...)  
 \* - table project\_investors (project\_id, investor\_id, role, unique)  
 \* - table investor\_milestones (id, project\_id, investor\_id, label, due\_date, status, comment, created\_at, updated\_at)  
 \* - type ENUM milestone\_status qui contient EXACTEMENT :  
 \* 'Payé', 'Pas payé', 'Annulé', 'En cours'  
 \*/  
  
// ---------------- Investors master ----------------  
router.get('/investors', authenticate, async (\_req, res, next) => {  
 try {  
 const { rows } = await pool.query('SELECT \* FROM investors ORDER BY name ASC');  
 res.json(rows);  
 } catch (e) { next(e); }  
});  
  
router.post('/investors', authenticate, async (req, res, next) => {  
 try {  
 const { name, company, email, phone, notes } = req.body;  
 const { rows } = await pool.query(  
 `INSERT INTO investors (name, company, email, phone, notes)  
 VALUES ($1, $2, $3, $4, $5)  
 RETURNING \*`,  
 [name, company || null, email || null, phone || null, notes || null]  
 );  
 res.status(201).json(rows[0]);  
 } catch (e) { next(e); }  
});  
  
// ---------------- Link / Unlink investor ↔ project ----------------  
router.get('/projects/:project\_id/investors', authenticate, async (req, res, next) => {  
 try {  
 const { project\_id } = req.params;  
 const { rows } = await pool.query(  
 `SELECT i.\*, pi.role  
 FROM project\_investors pi  
 JOIN investors i ON i.id = pi.investor\_id  
 WHERE pi.project\_id = $1  
 ORDER BY i.name ASC`,  
 [project\_id]  
 );  
 res.json(rows);  
 } catch (e) { next(e); }  
});  
  
router.post('/projects/:project\_id/investors', authenticate, async (req, res, next) => {  
 try {  
 const { project\_id } = req.params;  
 const { investor\_id, role } = req.body;  
 await pool.query(  
 `INSERT INTO project\_investors (project\_id, investor\_id, role)  
 VALUES ($1, $2, $3)  
 ON CONFLICT (project\_id, investor\_id) DO UPDATE SET role = EXCLUDED.role`,  
 [project\_id, investor\_id, role || null]  
 );  
 res.status(201).json({ success: true });  
 } catch (e) { next(e); }  
});  
  
router.delete('/projects/:project\_id/investors/:investor\_id', authenticate, async (req, res, next) => {  
 try {  
 const { project\_id, investor\_id } = req.params;  
 await pool.query(  
 `DELETE FROM project\_investors WHERE project\_id = $1 AND investor\_id = $2`,  
 [project\_id, investor\_id]  
 );  
 res.json({ success: true });  
 } catch (e) { next(e); }  
});  
  
// ---------------- Milestones (investor-specific) ----------------  
  
// LIST  
router.get('/projects/:project\_id/investors/:investor\_id/milestones', authenticate, async (req, res, next) => {  
 try {  
 const { project\_id, investor\_id } = req.params;  
 const rows = await listMilestones(project\_id, investor\_id);  
 res.json(rows);  
 } catch (e) { next(e); }  
});  
  
// CREATE  
router.post('/projects/:project\_id/investors/:investor\_id/milestones', authenticate, async (req, res, next) => {  
 try {  
 const { project\_id, investor\_id } = req.params;  
 // Les libellés FR exacts ("Payé", "Pas payé", ...) doivent être envoyés par le front.  
 const created = await addMilestone(project\_id, investor\_id, req.body || {});  
 res.status(201).json(created);  
 } catch (e) { next(e); }  
});  
  
// UPDATE (nested) — accepte PUT et PATCH  
router.put('/projects/:project\_id/investors/:investor\_id/milestones/:id', authenticate, async (req, res, next) => {  
 try {  
 const { project\_id, investor\_id, id } = req.params;  
 const updated = await updateMilestoneScoped(project\_id, investor\_id, id, req.body || {});  
 if (!updated) return res.status(404).json({ error: 'Jalon introuvable' });  
 res.json(updated);  
 } catch (e) { next(e); }  
});  
  
router.patch('/projects/:project\_id/investors/:investor\_id/milestones/:id', authenticate, async (req, res, next) => {  
 try {  
 const { project\_id, investor\_id, id } = req.params;  
 const updated = await updateMilestoneScoped(project\_id, investor\_id, id, req.body || {});  
 if (!updated) return res.status(404).json({ error: 'Jalon introuvable' });  
 res.json(updated);  
 } catch (e) { next(e); }  
});  
  
// DELETE  
router.delete('/projects/:project\_id/investors/:investor\_id/milestones/:id', authenticate, async (req, res, next) => {  
 try {  
 const { id } = req.params;  
 await deleteMilestone(id);  
 res.json({ success: true });  
 } catch (e) { next(e); }  
});  
  
// ---------------- Milestones (flat fallback) ----------------  
// Permet au front de faire PUT/PATCH /api/milestones/:id si la route nested n'existe pas sur certaines envs  
  
router.put('/milestones/:id', authenticate, async (req, res, next) => {  
 try {  
 const { id } = req.params;  
 const updated = await updateMilestoneById(id, req.body || {});  
 if (!updated) return res.status(404).json({ error: 'Jalon introuvable' });  
 res.json(updated);  
 } catch (e) { next(e); }  
});  
  
router.patch('/milestones/:id', authenticate, async (req, res, next) => {  
 try {  
 const { id } = req.params;  
 const updated = await updateMilestoneById(id, req.body || {});  
 if (!updated) return res.status(404).json({ error: 'Jalon introuvable' });  
 res.json(updated);  
 } catch (e) { next(e); }  
});  
  
router.get('/milestones/overdue', authenticate, async (\_req, res, next) => {  
 try {  
 const { rows } = await pool.query(`  
 SELECT m.\*, p.name AS project\_name, i.name AS investor\_name  
 FROM investor\_milestones m  
 JOIN projects p ON p.id = m.project\_id  
 LEFT JOIN investors i ON i.id = m.investor\_id  
 WHERE m.due\_date IS NOT NULL  
 AND m.due\_date::date < CURRENT\_DATE  
 AND (m.status IS NULL OR m.status NOT IN ('Payé','Annulé'))  
 ORDER BY m.due\_date ASC, m.id ASC  
 LIMIT 100  
 `);  
 res.json(rows);  
 } catch (e) { next(e); }  
});  
  
// --- DASHBOARD: Jalons à venir ---  
router.get('/milestones/upcoming', authenticate, async (\_req, res, next) => {  
 try {  
 const { rows } = await pool.query(`  
 SELECT m.\*, p.name AS project\_name, i.name AS investor\_name  
 FROM investor\_milestones m  
 JOIN projects p ON p.id = m.project\_id  
 LEFT JOIN investors i ON i.id = m.investor\_id  
 WHERE m.due\_date IS NOT NULL  
 AND m.due\_date::date >= CURRENT\_DATE  
 AND m.status IN ('Pas payé', 'En cours') -- exclut Payé / Annulé  
 ORDER BY m.due\_date ASC, m.id ASC  
 LIMIT 100  
 `);  
 res.json(rows);  
 } catch (e) { next(e); }  
});  
  
module.exports = router;

## C:/Users/rescue123/Documents/DEV/CRM/backend\src\routes\invitationRoutes.js

const express = require('express');  
const { authenticate } = require('../middleware/auth');  
const { createInvitation } = require('../models/invitationModel');  
const nodemailer = require('nodemailer');  
const router = express.Router();  
  
router.post('/:project\_id/invite', authenticate, async (req, res) => {  
 const { email } = req.body;  
 const invitation = await createInvitation(req.params.project\_id, email);  
  
 // Envoi email  
 const transporter = nodemailer.createTransport({  
 host: process.env.EMAIL\_HOST,  
 port: process.env.EMAIL\_PORT,  
 secure: process.env.EMAIL\_PORT == 465, // true for 465, false for other ports  
 auth: {  
 user: process.env.EMAIL\_USER,  
 pass: process.env.EMAIL\_PASSWORD,  
 },  
});  
  
  
 const url = `https://projets.domsolaire.fr/public/suivi/${invitation.token}`;  
 await transporter.sendMail({  
 from: `"Projet Solaire" <${process.env.EMAIL\_USER}>`,  
 to: email,  
 subject: 'Lien de suivi projet solaire',  
 text: `Bonjour !\n\nVoici le lien de suivi de votre projet : ${url}\n\nCe lien expirera le ${invitation.expires\_at.toLocaleString()}`,  
 html: `<p>Bonjour !<br><br>Voici le lien de suivi de votre projet : <a href="${url}">${url}</a><br><br>Ce lien expirera le <b>${invitation.expires\_at.toLocaleString()}</b></p>`  
});  
  
  
 res.json({ message: 'Invitation envoyée', url });  
});  
  
module.exports = router;

## C:/Users/rescue123/Documents/DEV/CRM/backend\src\routes\projectRoutes.js

const express = require('express');  
const { authenticate } = require('../middleware/auth');  
const projectModel = require('../models/projectModel');  
const router = express.Router();  
  
// Liste des projets  
router.get('/', authenticate, async (req, res) => {  
 let projects;  
 if (req.user.role === 'admin') {  
 projects = await projectModel.getAllProjects();  
 } else {  
 projects = await projectModel.getProjectsByOwnerId(req.user.id);  
 }  
 res.json(projects);  
});  
  
// Créer un projet (admin seulement)  
router.post('/', authenticate, async (req, res) => {  
 if (req.user.role !== 'admin') return res.status(403).json({ error: 'Accès interdit' });  
 const project = await projectModel.createProject(req.body);  
 res.json(project);  
});  
  
router.put('/:id', authenticate, async (req, res) => {  
 if (req.user.role !== 'admin') return res.status(403).json({ error: 'Accès interdit' });  
 const updated = await projectModel.updateProject(req.params.id, req.body);  
 res.json(updated);  
});  
  
router.delete('/:id', authenticate, async (req, res) => {  
 if (req.user.role !== 'admin') return res.status(403).json({ error: 'Accès interdit' });  
 await projectModel.deleteProject(req.params.id);  
 res.json({ success: true });  
});  
  
// --- Champs personnalisés (Custom Fields) ---  
  
// Ajouter un champ personnalisé  
router.post('/:id/custom\_fields', authenticate, async (req, res) => {  
 const projectId = parseInt(req.params.id, 10);  
 if (isNaN(projectId)) return res.status(400).json({ error: 'Project id invalide' });  
 const { field\_name, field\_value, user\_id } = req.body;  
 try {  
 const newField = await projectModel.addCustomField(projectId, field\_name, field\_value, user\_id);  
 res.status(201).json(newField);  
 } catch (err) {  
 res.status(500).json({ error: err.message });  
 }  
});  
  
// Récupérer les champs personnalisés  
router.get('/:id/custom\_fields', authenticate, async (req, res) => {  
 const projectId = parseInt(req.params.id, 10);  
 if (isNaN(projectId)) return res.status(400).json({ error: 'Project id invalide' });  
 try {  
 const fields = await projectModel.getCustomFields(projectId);  
 res.json(fields);  
 } catch (err) {  
 res.status(500).json({ error: err.message });  
 }  
});  
  
// Modifier un champ personnalisé  
router.put('/custom\_fields/:field\_id', authenticate, async (req, res) => {  
 const fieldId = parseInt(req.params.field\_id, 10);  
 if (isNaN(fieldId)) return res.status(400).json({ error: 'Field id invalide' });  
 const { field\_name, field\_value } = req.body;  
 try {  
 const updated = await projectModel.updateCustomField(fieldId, field\_name, field\_value);  
 res.json(updated);  
 } catch (err) {  
 res.status(500).json({ error: err.message });  
 }  
});  
  
// Supprimer un champ personnalisé  
router.delete('/custom\_fields/:field\_id', authenticate, async (req, res) => {  
 const fieldId = parseInt(req.params.field\_id, 10);  
 if (isNaN(fieldId)) return res.status(400).json({ error: 'Field id invalide' });  
 try {  
 await projectModel.deleteCustomField(fieldId);  
 res.json({ success: true });  
 } catch (err) {  
 res.status(500).json({ error: err.message });  
 }  
});  
  
// --- Détail d'un projet par ID ---  
router.get('/:id', authenticate, async (req, res) => {  
 const { id } = req.params;  
 const project = await projectModel.getProjectById(id);  
 if (!project) return res.status(404).json({ error: 'Projet non trouvé' });  
  
 if (req.user.role !== 'admin' && project.owner\_id !== req.user.id) {  
 return res.status(403).json({ error: 'Accès interdit' });  
 }  
 res.json(project);  
});  
  
module.exports = router;

## C:/Users/rescue123/Documents/DEV/CRM/backend\src\routes\projectStepRoutes.js

const express = require('express');  
const { authenticate } = require('../middleware/auth');  
const { addStep, getSteps, deleteStep } = require('../models/projectStepModel');  
const router = express.Router();  
  
router.post('/:project\_id/steps', authenticate, async (req, res) => {  
 const { label, step\_date, step\_comment, status } = req.body;  
 const step = await addStep(req.params.project\_id, label, step\_date, step\_comment, status);  
 res.json(step);  
});  
  
// Récupérer les étapes  
router.get('/:project\_id/steps', authenticate, async (req, res) => {  
 const steps = await getSteps(req.params.project\_id);  
 res.json(steps);  
});  
// Route pour supprimer une étape  
router.delete('/:project\_id/steps/:step\_id', authenticate, async (req, res) => {  
 try {  
 const { project\_id, step\_id } = req.params;  
 await deleteStep(step\_id); // Assurez-vous que cette fonction est définie dans votre modèle  
 res.status(200).send({ message: 'Étape supprimée avec succès' });  
 } catch (error) {  
 console.error("Erreur lors de la suppression de l'étape:", error);  
 res.status(500).send({ message: 'Erreur lors de la suppression de l\'étape' });  
 }  
});  
  
module.exports = router;

## C:/Users/rescue123/Documents/DEV/CRM/backend\src\routes\publicRoutes.js

const express = require('express');  
const pool = require('../utils/db');  
const router = express.Router();  
  
router.get('/projects/:token', async (req, res) => {  
 const { token } = req.params;  
 const { rows } = await pool.query(  
 `SELECT p.\*, i.token, i.expires\_at  
 FROM projects p  
 JOIN invitations i ON p.id = i.project\_id  
 WHERE i.token = $1 AND i.expires\_at > NOW()`, [token]  
 );  
 if (rows.length === 0) return res.status(404).json({ error: 'Lien expiré ou invalide' });  
 const project = rows[0];  
 const { rows: steps } = await pool.query(  
 'SELECT \* FROM project\_steps WHERE project\_id = $1 ORDER BY step\_date DESC',  
 [project.id]  
 );  
 const { rows: custom\_fields } = await pool.query(  
 'SELECT \* FROM project\_custom\_fields WHERE project\_id = $1',  
 [project.id]  
 );  
 res.json({  
 ...project,  
 steps,  
 custom\_fields,  
 });  
});  
  
module.exports = router;

## C:/Users/rescue123/Documents/DEV/CRM/backend\src\routes\userRoutes.js

const express = require('express');  
const bcrypt = require('bcryptjs');  
const jwt = require('jsonwebtoken');  
const { getUserByEmail, createUser } = require('../models/userModel');  
const { authenticate } = require('../middleware/auth');  
const router = express.Router();  
const { getAllUsers } = require('../models/userModel'); // ajoute ce require en haut  
  
// Inscription  
// Inscription (admin only via API, ou public via front login)  
router.post('/register', authenticate, async (req, res) => {  
 if (req.user.role !== 'admin') return res.status(403).json({ error: 'Accès interdit' });  
 const { name, email, password } = req.body;  
 const hash = await bcrypt.hash(password, 10);  
 try {  
 const user = await createUser(name, email, hash, 'user');  
 res.json(user);  
 } catch (e) {  
 res.status(400).json({ error: 'Email déjà utilisé' });  
 }  
});  
  
// Connexion  
router.post('/login', async (req, res) => {  
 console.log('BODY:', req.body); // AJOUTE CETTE LIGNE !  
  
 const { email, password } = req.body;  
 const user = await getUserByEmail(email);  
 if (!user) return res.status(401).json({ error: 'Identifiants invalides' });  
  
 const valid = await bcrypt.compare(password, user.password\_hash);  
 if (!valid) return res.status(401).json({ error: 'Identifiants invalides' });  
  
 const token = jwt.sign(  
 { id: user.id, role: user.role, email: user.email },  
 process.env.JWT\_SECRET,  
 { expiresIn: '7d' }  
 );  
 res.json({ token, user: { id: user.id, name: user.name, email: user.email, role: user.role } });  
});  
  
// Exemple route protégée  
router.get('/me', authenticate, (req, res) => {  
 res.json(req.user);  
});  
// Nouvelle route : Liste tous les utilisateurs (admin only)  
router.get('/', authenticate, (req, res, next) => {  
 if (req.user.role !== 'admin') return res.status(403).json({ error: 'Accès interdit' });  
 getAllUsers()  
 .then(users => res.json(users))  
 .catch(next);  
});  
module.exports = router;

## C:/Users/rescue123/Documents/DEV/CRM/backend\src\utils\db.js

const { Pool } = require('pg');  
require('dotenv').config();  
  
const pool = new Pool({  
   
 user: process.env.DB\_USER,  
 host: process.env.DB\_HOST,  
 database: process.env.DB\_NAME,  
 password: process.env.DB\_PASSWORD,  
 port: process.env.DB\_PORT || 5432,  
});  
console.log('DB en cours :', process.env.DB\_NAME);  
  
module.exports = pool;

## C:/Users/rescue123/Documents/DEV/CRM/backend\src\utils\geocode.js

const axios = require('axios');  
  
async function geocodeAddress(address) {  
 const url = `https://nominatim.openstreetmap.org/search?format=json&q=${encodeURIComponent(address)}`;  
 const { data } = await axios.get(url, { headers: { 'User-Agent': 'carteprojet/1.0' } });  
 if (data && data.length > 0) {  
 return { latitude: parseFloat(data[0].lat), longitude: parseFloat(data[0].lon) };  
 }  
 return null;  
}  
  
module.exports = { geocodeAddress };

## C:/Users/rescue123/Documents/DEV/CRM/backend\tmp\restart.txt

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\.env

VITE\_APP\_NAME=Dom Solaire CRM

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\.gitignore

# Logs  
logs  
\*.log  
npm-debug.log\*  
yarn-debug.log\*  
yarn-error.log\*  
pnpm-debug.log\*  
lerna-debug.log\*  
  
node\_modules  
dist  
dist-ssr  
\*.local  
  
# Editor directories and files  
.vscode/\*  
!.vscode/extensions.json  
.idea  
.DS\_Store  
\*.suo  
\*.ntvs\*  
\*.njsproj  
\*.sln  
\*.sw?

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\eslint.config.js

import js from '@eslint/js'  
import globals from 'globals'  
import reactHooks from 'eslint-plugin-react-hooks'  
import reactRefresh from 'eslint-plugin-react-refresh'  
import { defineConfig, globalIgnores } from 'eslint/config'  
  
export default defineConfig([  
 globalIgnores(['dist']),  
 {  
 files: ['\*\*/\*.{js,jsx}'],  
 extends: [  
 js.configs.recommended,  
 reactHooks.configs['recommended-latest'],  
 reactRefresh.configs.vite,  
 ],  
 languageOptions: {  
 ecmaVersion: 2020,  
 globals: globals.browser,  
 parserOptions: {  
 ecmaVersion: 'latest',  
 ecmaFeatures: { jsx: true },  
 sourceType: 'module',  
 },  
 },  
 rules: {  
 'no-unused-vars': ['error', { varsIgnorePattern: '^[A-Z\_]' }],  
 },  
 },  
])

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\index.html

<!doctype html>  
<html lang="en">  
 <head>  
 <meta charset="UTF-8" />  
 <link rel="icon" type="image/svg+xml" href="/vite.svg" />  
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />  
 <title>Vite + React</title>  
 </head>  
 <body>  
 <div id="root"></div>  
 <script type="module" src="/src/main.jsx"></script>  
 </body>  
</html>

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\package.json

{  
 "name": "crm-frontend",  
 "private": true,  
 "version": "0.0.0",  
 "type": "module",  
 "scripts": {  
 "dev": "vite",  
 "build": "vite build",  
 "lint": "eslint .",  
 "preview": "vite preview"  
 },  
 "dependencies": {  
 "@date-io/date-fns": "^3.2.1",  
 "@emotion/react": "^11.14.0",  
 "@emotion/styled": "^11.14.1",  
 "@mui/icons-material": "^7.3.1",  
 "@mui/lab": "^7.0.0-beta.16",  
 "@mui/material": "^7.3.1",  
 "@mui/x-charts": "^8.9.2",  
 "@mui/x-data-grid": "^8.9.2",  
 "axios": "^1.11.0",  
 "date-fns": "^4.1.0",  
 "leaflet": "^1.9.4",  
 "react": "^19.1.0",  
 "react-dom": "^19.1.0",  
 "react-leaflet": "^5.0.0",  
 "react-router-dom": "^7.7.1"  
 },  
 "devDependencies": {  
 "@eslint/js": "^9.30.1",  
 "@types/react": "^19.1.8",  
 "@types/react-dom": "^19.1.6",  
 "@vitejs/plugin-react": "^4.6.0",  
 "eslint": "^9.30.1",  
 "eslint-plugin-react-hooks": "^5.2.0",  
 "eslint-plugin-react-refresh": "^0.4.20",  
 "globals": "^16.3.0",  
 "vite": "^7.0.4"  
 }  
}

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\README.md

# React + Vite  
  
This template provides a minimal setup to get React working in Vite with HMR and some ESLint rules.  
  
Currently, two official plugins are available:  
  
- [@vitejs/plugin-react](https://github.com/vitejs/vite-plugin-react/blob/main/packages/plugin-react) uses [Babel](https://babeljs.io/) for Fast Refresh  
- [@vitejs/plugin-react-swc](https://github.com/vitejs/vite-plugin-react/blob/main/packages/plugin-react-swc) uses [SWC](https://swc.rs/) for Fast Refresh  
  
## Expanding the ESLint configuration  
  
If you are developing a production application, we recommend using TypeScript with type-aware lint rules enabled. Check out the [TS template](https://github.com/vitejs/vite/tree/main/packages/create-vite/template-react-ts) for information on how to integrate TypeScript and [`typescript-eslint`](https://typescript-eslint.io) in your project.

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\vite.config.js

import { defineConfig } from 'vite'  
import react from '@vitejs/plugin-react'  
  
// https://vite.dev/config/  
export default defineConfig({  
 plugins: [react()],  
})

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\public\vite.svg

<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" aria-hidden="true" role="img" class="iconify iconify--logos" width="31.88" height="32" preserveAspectRatio="xMidYMid meet" viewBox="0 0 256 257"><defs><linearGradient id="IconifyId1813088fe1fbc01fb466" x1="-.828%" x2="57.636%" y1="7.652%" y2="78.411%"><stop offset="0%" stop-color="#41D1FF"></stop><stop offset="100%" stop-color="#BD34FE"></stop></linearGradient><linearGradient id="IconifyId1813088fe1fbc01fb467" x1="43.376%" x2="50.316%" y1="2.242%" y2="89.03%"><stop offset="0%" stop-color="#FFEA83"></stop><stop offset="8.333%" stop-color="#FFDD35"></stop><stop offset="100%" stop-color="#FFA800"></stop></linearGradient></defs><path fill="url(#IconifyId1813088fe1fbc01fb466)" d="M255.153 37.938L134.897 252.976c-2.483 4.44-8.862 4.466-11.382.048L.875 37.958c-2.746-4.814 1.371-10.646 6.827-9.67l120.385 21.517a6.537 6.537 0 0 0 2.322-.004l117.867-21.483c5.438-.991 9.574 4.796 6.877 9.62Z"></path><path fill="url(#IconifyId1813088fe1fbc01fb467)" d="M185.432.063L96.44 17.501a3.268 3.268 0 0 0-2.634 3.014l-5.474 92.456a3.268 3.268 0 0 0 3.997 3.378l24.777-5.718c2.318-.535 4.413 1.507 3.936 3.838l-7.361 36.047c-.495 2.426 1.782 4.5 4.151 3.78l15.304-4.649c2.372-.72 4.652 1.36 4.15 3.788l-11.698 56.621c-.732 3.542 3.979 5.473 5.943 2.437l1.313-2.028l72.516-144.72c1.215-2.423-.88-5.186-3.54-4.672l-25.505 4.922c-2.396.462-4.435-1.77-3.759-4.114l16.646-57.705c.677-2.35-1.37-4.583-3.769-4.113Z"></path></svg>

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\App.css

#root {  
 max-width: 1280px;  
 margin: 0 auto;  
 padding: 2rem;  
 text-align: center;  
}  
  
.logo {  
 height: 6em;  
 padding: 1.5em;  
 will-change: filter;  
 transition: filter 300ms;  
}  
.logo:hover {  
 filter: drop-shadow(0 0 2em #646cffaa);  
}  
.logo.react:hover {  
 filter: drop-shadow(0 0 2em #61dafbaa);  
}  
  
@keyframes logo-spin {  
 from {  
 transform: rotate(0deg);  
 }  
 to {  
 transform: rotate(360deg);  
 }  
}  
  
@media (prefers-reduced-motion: no-preference) {  
 a:nth-of-type(2) .logo {  
 animation: logo-spin infinite 20s linear;  
 }  
}  
  
.card {  
 padding: 2em;  
}  
  
.read-the-docs {  
 color: #888;  
}

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\App.jsx

// src/App.jsx  
import { BrowserRouter, Routes, Route, Navigate } from "react-router-dom";  
import LoginPage from "./pages/LoginPage";  
import Dashboard from "./pages/Dashboard";  
import ProjectDetail from "./pages/ProjectDetail";  
import { useContext } from "react";  
import { AuthContext } from "./contexts/AuthContext";  
import { Box } from "@mui/material";  
import CarteDesProjets from "./pages/CarteDesProjets";  
import ProjectList from "./pages/ProjectList";  
 import PublicSuiviPage from './pages/PublicSuiviPage';  
import InvestorList from "./pages/InvestorList";  
  
function App() {  
 const { token } = useContext(AuthContext);  
  
 return (  
 <BrowserRouter>  
 {/\* Décale tout le contenu sous le header (AppBar) \*/}  
 <Routes>  
 <Route path="/login" element={<LoginPage />} />  
 <Route path="/projects" element={token ? <Dashboard /> : <Navigate to="/login" />} />  
 <Route path="/projects/:id" element={token ? <ProjectDetail /> : <Navigate to="/login" />} />  
 <Route path="\*" element={<Navigate to="/projects" />} />  
 <Route path="/carte" element={token ? <CarteDesProjets /> : <Navigate to="/login" />} />  
 <Route path="/projects-list" element={<ProjectList />} />  
 <Route path="/investors-list" element={<InvestorList />} />  
 <Route path="/public/suivi/:token" element={<PublicSuiviPage />} />  
  
  
 </Routes>  
 </BrowserRouter>  
 );  
}  
  
export default App;

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\index.css

:root {  
 font-family: system-ui, Avenir, Helvetica, Arial, sans-serif;  
 line-height: 1.5;  
 font-weight: 400;  
  
 color-scheme: light dark;  
 color: rgba(255, 255, 255, 0.87);  
 background-color: #242424;  
  
 font-synthesis: none;  
 text-rendering: optimizeLegibility;  
 -webkit-font-smoothing: antialiased;  
 -moz-osx-font-smoothing: grayscale;  
}  
  
a {  
 font-weight: 500;  
 color: #646cff;  
 text-decoration: inherit;  
}  
a:hover {  
 color: #535bf2;  
}  
  
html, body, #root {  
 width: 100%;  
 height: 100%;  
}  
body {  
 margin: 0;  
 min-width: 320px;  
 min-height: 100vh;  
 /\* pas de flex ici, on laisse le layout gérer la largeur \*/  
}  
  
h1 {  
 font-size: 3.2em;  
 line-height: 1.1;  
}  
  
button {  
 border-radius: 8px;  
 border: 1px solid transparent;  
 padding: 0.6em 1.2em;  
 font-size: 1em;  
 font-weight: 500;  
 font-family: inherit;  
 background-color: #1a1a1a;  
 cursor: pointer;  
 transition: border-color 0.25s;  
}  
button:hover {  
 border-color: #646cff;  
}  
button:focus,  
button:focus-visible {  
 outline: 4px auto -webkit-focus-ring-color;  
}  
  
@media (prefers-color-scheme: light) {  
 :root {  
 color: #213547;  
 background-color: #ffffff;  
 }  
 a:hover {  
 color: #747bff;  
 }  
 button {  
 background-color: #f9f9f9;  
 }  
}

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\leaflet-fix.js

import L from "leaflet";  
import markerIcon2x from "leaflet/dist/images/marker-icon-2x.png";  
import markerIcon from "leaflet/dist/images/marker-icon.png";  
import markerShadow from "leaflet/dist/images/marker-shadow.png";  
  
// Corrige les chemins des images pour Vite  
L.Icon.Default.mergeOptions({  
 iconRetinaUrl: markerIcon2x,  
 iconUrl: markerIcon,  
 shadowUrl: markerShadow,  
});

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\main.jsx

import React from 'react';  
import ReactDOM from 'react-dom/client';  
import App from './App.jsx';  
import { AuthProvider } from './contexts/AuthContext.jsx';  
import './index.css';  
import 'leaflet/dist/leaflet.css';  
import './leaflet-fix';  
import { APP\_NAME } from './config/app';  
document.title = APP\_NAME;  
  
ReactDOM.createRoot(document.getElementById('root')).render(  
 <React.StrictMode>  
 <AuthProvider>  
 <App />  
 </AuthProvider>  
 </React.StrictMode>  
);

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\api\axios.jsx

// src/api/axios.jsx  
import axios from "axios";  
  
const api = axios.create({  
 baseURL: "/api", // <- plus "localhost"  
});  
  
export function setAuthToken(token) {  
 if (token) {  
 api.defaults.headers.common["Authorization"] = `Bearer ${token}`;  
 } else {  
 delete api.defaults.headers.common["Authorization"];  
 }  
}  
  
// 👉 initialise l’Authorization dès le chargement du module  
const saved = localStorage.getItem("token");  
if (saved) setAuthToken(saved);  
  
export default api;

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\api\investors.js

import api from './axios';  
  
/\*\*  
 \* Mapping UI -> valeurs internes (sans accents) utilisées dans le front.  
 \* Les libellés FR (gauche) sont ceux affichés dans les selects.  
 \* Les valeurs internes (droite) sont celles qu'on stocke côté UI.  
 \*/  
export const STATUS\_MAP = {  
 'Payé': 'payee',  
 'Pas payé': 'pas\_payee',  
 'Annulé': 'annule',  
 'En cours': 'en\_cours',  
};  
  
// Inverse: valeur interne -> libellé FR exact attendu par l'ENUM Postgres  
export const REVERSE\_STATUS\_MAP = Object.fromEntries(  
 Object.entries(STATUS\_MAP).map(([label, value]) => [value, label])  
);  
  
// Optionnel: set des libellés FR valides (ENUM DB)  
export const DB\_STATUS = new Set(['Payé', 'Pas payé', 'Annulé', 'En cours']);  
  
export const listInvestors = () =>  
 api.get('/investors').then((r) => r.data);  
  
export const listProjectInvestors = (projectId) =>  
 api.get(`/projects/${projectId}/investors`).then((r) => r.data);  
  
export const attachInvestor = (projectId, investor\_id, role) =>  
 api.post(`/projects/${projectId}/investors`, { investor\_id, role }).then((r) => r.data);  
  
export const detachInvestor = (projectId, investor\_id) =>  
 api.delete(`/projects/${projectId}/investors/${investor\_id}`).then((r) => r.data);  
  
export const listMilestones = (projectId, investor\_id) =>  
 api.get(`/projects/${projectId}/investors/${investor\_id}/milestones`).then((r) => r.data);  
  
export const deleteMilestone = (projectId, investorId, milestoneId) =>  
 api.delete(`/projects/${projectId}/investors/${investorId}/milestones/${milestoneId}`).then(r => r.data);  
  
/\*\*  
 \* Ajout jalon : accepte un body avec status interne ("en\_cours") ou libellé FR ("En cours").  
 \* On convertit toujours vers le libellé FR exact pour la DB.  
 \*/  
export async function addMilestone(projectId, investorId, body) {  
 const statusFR = DB\_STATUS.has(body.status) ? body.status : (REVERSE\_STATUS\_MAP[body.status] ?? body.status);  
 const payload = {  
 ...body,  
 status: statusFR, // "Payé" | "Pas payé" | "Annulé" | "En cours"  
 };  
 const { data } = await api.post(  
 `/projects/${projectId}/investors/${investorId}/milestones`,  
 payload  
 );  
 return data; // 201 + objet créé  
}  
  
/\*\*  
 \* Mise à jour jalon (ex: changer statut).  
 \* `body.status` peut être interne ("payee") ou FR ("Payé"), on mappe vers FR pour la DB.  
 \*/  
export async function updateMilestone(projectId, investorId, milestoneId, body) {  
 const statusFR = body.status != null  
 ? (DB\_STATUS.has(body.status) ? body.status : (REVERSE\_STATUS\_MAP[body.status] ?? body.status))  
 : undefined;  
  
 const payload = {  
 ...body,  
 ...(statusFR != null ? { status: statusFR } : {}), // n'envoie status que si présent  
 };  
  
 const { data } = await api.put(  
 `/projects/${projectId}/investors/${investorId}/milestones/${milestoneId}`,  
 payload  
 );  
 return data;  
}  
  
export const createInvestor = (payload) =>  
 api.post('/investors', payload).then((r) => r.data);

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\assets\react.svg

<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" aria-hidden="true" role="img" class="iconify iconify--logos" width="35.93" height="32" preserveAspectRatio="xMidYMid meet" viewBox="0 0 256 228"><path fill="#00D8FF" d="M210.483 73.824a171.49 171.49 0 0 0-8.24-2.597c.465-1.9.893-3.777 1.273-5.621c6.238-30.281 2.16-54.676-11.769-62.708c-13.355-7.7-35.196.329-57.254 19.526a171.23 171.23 0 0 0-6.375 5.848a155.866 155.866 0 0 0-4.241-3.917C100.759 3.829 77.587-4.822 63.673 3.233C50.33 10.957 46.379 33.89 51.995 62.588a170.974 170.974 0 0 0 1.892 8.48c-3.28.932-6.445 1.924-9.474 2.98C17.309 83.498 0 98.307 0 113.668c0 15.865 18.582 31.778 46.812 41.427a145.52 145.52 0 0 0 6.921 2.165a167.467 167.467 0 0 0-2.01 9.138c-5.354 28.2-1.173 50.591 12.134 58.266c13.744 7.926 36.812-.22 59.273-19.855a145.567 145.567 0 0 0 5.342-4.923a168.064 168.064 0 0 0 6.92 6.314c21.758 18.722 43.246 26.282 56.54 18.586c13.731-7.949 18.194-32.003 12.4-61.268a145.016 145.016 0 0 0-1.535-6.842c1.62-.48 3.21-.974 4.76-1.488c29.348-9.723 48.443-25.443 48.443-41.52c0-15.417-17.868-30.326-45.517-39.844Zm-6.365 70.984c-1.4.463-2.836.91-4.3 1.345c-3.24-10.257-7.612-21.163-12.963-32.432c5.106-11 9.31-21.767 12.459-31.957c2.619.758 5.16 1.557 7.61 2.4c23.69 8.156 38.14 20.213 38.14 29.504c0 9.896-15.606 22.743-40.946 31.14Zm-10.514 20.834c2.562 12.94 2.927 24.64 1.23 33.787c-1.524 8.219-4.59 13.698-8.382 15.893c-8.067 4.67-25.32-1.4-43.927-17.412a156.726 156.726 0 0 1-6.437-5.87c7.214-7.889 14.423-17.06 21.459-27.246c12.376-1.098 24.068-2.894 34.671-5.345a134.17 134.17 0 0 1 1.386 6.193ZM87.276 214.515c-7.882 2.783-14.16 2.863-17.955.675c-8.075-4.657-11.432-22.636-6.853-46.752a156.923 156.923 0 0 1 1.869-8.499c10.486 2.32 22.093 3.988 34.498 4.994c7.084 9.967 14.501 19.128 21.976 27.15a134.668 134.668 0 0 1-4.877 4.492c-9.933 8.682-19.886 14.842-28.658 17.94ZM50.35 144.747c-12.483-4.267-22.792-9.812-29.858-15.863c-6.35-5.437-9.555-10.836-9.555-15.216c0-9.322 13.897-21.212 37.076-29.293c2.813-.98 5.757-1.905 8.812-2.773c3.204 10.42 7.406 21.315 12.477 32.332c-5.137 11.18-9.399 22.249-12.634 32.792a134.718 134.718 0 0 1-6.318-1.979Zm12.378-84.26c-4.811-24.587-1.616-43.134 6.425-47.789c8.564-4.958 27.502 2.111 47.463 19.835a144.318 144.318 0 0 1 3.841 3.545c-7.438 7.987-14.787 17.08-21.808 26.988c-12.04 1.116-23.565 2.908-34.161 5.309a160.342 160.342 0 0 1-1.76-7.887Zm110.427 27.268a347.8 347.8 0 0 0-7.785-12.803c8.168 1.033 15.994 2.404 23.343 4.08c-2.206 7.072-4.956 14.465-8.193 22.045a381.151 381.151 0 0 0-7.365-13.322Zm-45.032-43.861c5.044 5.465 10.096 11.566 15.065 18.186a322.04 322.04 0 0 0-30.257-.006c4.974-6.559 10.069-12.652 15.192-18.18ZM82.802 87.83a323.167 323.167 0 0 0-7.227 13.238c-3.184-7.553-5.909-14.98-8.134-22.152c7.304-1.634 15.093-2.97 23.209-3.984a321.524 321.524 0 0 0-7.848 12.897Zm8.081 65.352c-8.385-.936-16.291-2.203-23.593-3.793c2.26-7.3 5.045-14.885 8.298-22.6a321.187 321.187 0 0 0 7.257 13.246c2.594 4.48 5.28 8.868 8.038 13.147Zm37.542 31.03c-5.184-5.592-10.354-11.779-15.403-18.433c4.902.192 9.899.29 14.978.29c5.218 0 10.376-.117 15.453-.343c-4.985 6.774-10.018 12.97-15.028 18.486Zm52.198-57.817c3.422 7.8 6.306 15.345 8.596 22.52c-7.422 1.694-15.436 3.058-23.88 4.071a382.417 382.417 0 0 0 7.859-13.026a347.403 347.403 0 0 0 7.425-13.565Zm-16.898 8.101a358.557 358.557 0 0 1-12.281 19.815a329.4 329.4 0 0 1-23.444.823c-7.967 0-15.716-.248-23.178-.732a310.202 310.202 0 0 1-12.513-19.846h.001a307.41 307.41 0 0 1-10.923-20.627a310.278 310.278 0 0 1 10.89-20.637l-.001.001a307.318 307.318 0 0 1 12.413-19.761c7.613-.576 15.42-.876 23.31-.876H128c7.926 0 15.743.303 23.354.883a329.357 329.357 0 0 1 12.335 19.695a358.489 358.489 0 0 1 11.036 20.54a329.472 329.472 0 0 1-11 20.722Zm22.56-122.124c8.572 4.944 11.906 24.881 6.52 51.026c-.344 1.668-.73 3.367-1.15 5.09c-10.622-2.452-22.155-4.275-34.23-5.408c-7.034-10.017-14.323-19.124-21.64-27.008a160.789 160.789 0 0 1 5.888-5.4c18.9-16.447 36.564-22.941 44.612-18.3ZM128 90.808c12.625 0 22.86 10.235 22.86 22.86s-10.235 22.86-22.86 22.86s-22.86-10.235-22.86-22.86s10.235-22.86 22.86-22.86Z"></path></svg>

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\components\AutocompleteAdresse.jsx

import { Autocomplete, TextField, CircularProgress } from "@mui/material";  
import { useState } from "react";  
import axios from "axios";  
  
export default function AutocompleteAdresse({ value, onChange, onCoords }) {  
 const [options, setOptions] = useState([]);  
 const [loading, setLoading] = useState(false);  
  
 // Quand on tape dans le champ :  
 const handleInputChange = async (event, inputValue) => {  
 if (!inputValue || inputValue.length < 3) {  
 setOptions([]);  
 return;  
 }  
 setLoading(true);  
 try {  
 const res = await axios.get("https://api-adresse.data.gouv.fr/search", {  
 params: { q: inputValue, limit: 10 }  
 });  
 setOptions(res.data.features);  
 } catch {  
 setOptions([]);  
 }  
 setLoading(false);  
 };  
  
 // Quand on sélectionne une adresse :  
 const handleChange = (event, newValue) => {  
 if (newValue) {  
 // Récupère l'adresse, la lat/lon  
 onChange(newValue.properties.label);  
 if (onCoords) {  
 onCoords({  
 latitude: newValue.geometry.coordinates[1],  
 longitude: newValue.geometry.coordinates[0]  
 });  
 }  
 } else {  
 onChange("");  
 if (onCoords) onCoords({ latitude: null, longitude: null });  
 }  
 };  
  
 return (  
 <Autocomplete  
 freeSolo  
 options={options}  
 loading={loading}  
 getOptionLabel={option =>  
 typeof option === "string" ? option : option.properties.label  
 }  
 onInputChange={handleInputChange}  
 onChange={handleChange}  
 filterOptions={x => x} // pas de filtrage côté client  
 renderInput={params => (  
 <TextField  
 {...params}  
 label="Adresse \*"  
 value={value}  
 fullWidth  
 sx={{ mb: 2 }}  
 InputProps={{  
 ...params.InputProps,  
 endAdornment: (  
 <>  
 {loading ? <CircularProgress color="inherit" size={20} /> : null}  
 {params.InputProps.endAdornment}  
 </>  
 ),  
 }}  
 />  
 )}  
 />  
 );  
}

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\components\Header.jsx

import { useContext } from "react";  
import { AppBar, Toolbar, Typography, Button, Box } from "@mui/material";  
import { Link, useNavigate, useLocation } from "react-router-dom";  
import { AuthContext } from "../contexts/AuthContext";  
  
export default function Header() {  
 const { token, user, logout } = useContext(AuthContext);  
 const navigate = useNavigate();  
 const location = useLocation();  
  
 const handleLogout = () => {  
 logout();  
 navigate("/login");  
 };  
  
 // Logo clickable : va toujours sur le dashboard si connecté, sinon accueil  
 const handleLogoClick = () => {  
 if (token) navigate("/projects");  
 else navigate("/login");  
 };  
  
 return (  
 <AppBar position="fixed" sx={{ background: "#0288d1", zIndex: 1200 }}>  
 <Toolbar sx={{ display: "flex", justifyContent: "space-between", minHeight: 64 }}>  
 {/\* Logo / App Name \*/}  
 <Box sx={{ display: "flex", alignItems: "center", cursor: "pointer" }} onClick={handleLogoClick}>  
 <Typography variant="h6" fontWeight={800} sx={{ color: "white", mr: 1 }}>  
 🌞 Gestion des Projets Solaires  
 </Typography>  
 </Box>  
 {/\* Navigation boutons \*/}  
 <Box sx={{ display: "flex", alignItems: "center", gap: 1 }}>  
 {token ? (  
 <>  
 <Button  
 color={location.pathname === "/projects" ? "success" : "inherit"}  
 variant="text"  
 component={Link}  
 to="/projects"  
 sx={{ fontWeight: 600 }}  
 >  
 Dashboard  
 </Button>  
 <Button  
 color={location.pathname === "/carte" ? "success" : "inherit"}  
 variant="text"  
 component={Link}  
 to="/carte"  
 sx={{ fontWeight: 600 }}  
>  
 Carte des projets  
</Button>  
 <Typography sx={{ mx: 2, fontWeight: 600, color: "#fff" }}>  
 {user?.name}  
 {user?.role === "admin" && (  
 <span style={{  
 background: "#06c270", color: "#fff",  
 fontSize: "12px", padding: "2px 8px", borderRadius: "1em",  
 marginLeft: 8, fontWeight: 600  
 }}>  
 Admin  
 </span>  
 )}  
 </Typography>  
 <Button variant="outlined" sx={{  
 color: "#fff", borderColor: "#fff",  
 "&:hover": { borderColor: "#fff", background: "rgba(255,255,255,0.05)" }  
 }} onClick={handleLogout}>Déconnexion</Button>  
 </>  
 ) : (  
 <Button  
 variant="contained"  
 color="success"  
 component={Link}  
 to="/login"  
 sx={{ fontWeight: 600 }}  
 >  
 Se connecter  
 </Button>  
 )}  
 </Box>  
 </Toolbar>  
 </AppBar>  
 );  
}

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\components\dashboard\OverdueMilestones.jsx

import { useEffect, useState } from "react";  
import { Paper, Typography, List, ListItem, ListItemText, Chip, Box } from "@mui/material";  
import api from "../../api/axios";  
  
function normalize(s) {  
 return String(s ?? "")  
 .toLowerCase()  
 .normalize("NFD").replace(/\p{Diacritic}/gu, "")  
 .replace(/\_/g, " ")  
 .replace(/\s+/g, " ")  
 .trim();  
}  
function isClosedStatus(status) {  
 const n = normalize(status);  
 return n === "paye" || n === "payee" || n === "annule";  
}  
function startOfDay(d) {  
 const x = new Date(d);  
 x.setHours(0, 0, 0, 0);  
 return x;  
}  
  
export default function OverdueMilestones() {  
 const [items, setItems] = useState([]);  
  
 useEffect(() => {  
 (async () => {  
 try {  
 const { data } = await api.get("/milestones/overdue");  
 const arr = Array.isArray(data) ? data : [];  
 const today0 = startOfDay(new Date());  
 const filtered = arr  
 .filter(m => m?.due\_date)  
 .filter(m => startOfDay(m.due\_date) < today0)  
 .filter(m => !isClosedStatus(m?.status))  
 .sort((a, b) => new Date(a.due\_date) - new Date(b.due\_date));  
 setItems(filtered.slice(0, 8));  
 } catch (e) {  
 console.error("Erreur chargement jalons en retard", e);  
 setItems([]);  
 }  
 })();  
 }, []);  
  
 const count = items.length;  
  
 return (  
 <Paper  
 sx={{  
 p: { xs: 1.5, md: 2 },  
 borderRadius: 3,  
 boxShadow: { xs: 1, md: 3 },  
 height: "100%",  
 display: "flex",  
 flexDirection: "column",  
 gap: 1,  
 m: { xs: 0, md: 2 },  
 minWidth: { xs: 260, md: 'auto' },  
 }}  
 >  
 <Typography fontWeight={700}>⏳ Jalons en retard</Typography>  
 <Typography variant="caption" color="text.secondary" sx={{ mb: 0.5 }}>  
 {count === 0 ? "Aucun jalon en retard 🎉" : ""}  
 </Typography>  
 <Box sx={{ flex: 1, minHeight: 0, overflowY: "auto", maxHeight: { xs: 180, md: 'none' } }}>  
 <List dense>  
 {items.map(m => {  
 const d = m.due\_date ? new Date(m.due\_date) : null;  
 return (  
 <ListItem  
 key={m.id}  
 sx={{ py: 0.5, cursor: "pointer", "&:hover": { bgcolor: "#fafafa" }, borderRadius: 1 }}  
 onClick={() => window.location.assign(`/projects/${m.project\_id}`)}  
 secondaryAction={  
 d ? <Chip size="small" color="error" label={d.toLocaleDateString("fr-FR")} /> : null  
 }  
 >  
 <ListItemText  
 primary={<b>{m.label || "Jalon"}</b>}  
 secondary={`${m.project\_name ?? "Projet"}${m.investor\_name ? " • " + m.investor\_name : ""}`}  
 />  
 </ListItem>  
 );  
 })}  
 </List>  
 </Box>  
 </Paper>  
 );  
}

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\components\dashboard\ProjectStatusDonut.jsx

// components/dashboard/ProjectStatusDonut.jsx  
import \* as React from 'react';  
import { Box, Paper, Typography, useMediaQuery } from '@mui/material';  
import { useTheme } from '@mui/material/styles';  
import { PieChart, pieArcLabelClasses } from '@mui/x-charts/PieChart';  
  
export default function ProjectStatusDonut({ counts }) {  
 const theme = useTheme();  
 const isMobile = useMediaQuery(theme.breakpoints.down('sm'));  
 const size = isMobile ? { width: 260, height: 200 } : { width: 400, height: 240 };  
  
 const safeCounts = counts ?? { enCours: 150, terminee: 300, annulee: 50 };  
  
 const data = [  
 { id: 0, label: 'En cours', value: safeCounts.enCours ?? 0 },  
 { id: 1, label: 'Terminée', value: safeCounts.terminee ?? 0 },  
 { id: 2, label: 'Annulée', value: safeCounts.annulee ?? 0 },  
 ];  
  
 const total = data.reduce((s, d) => s + d.value, 0);  
  
 const colors = [  
 theme.palette.info.main,  
 theme.palette.success.main,  
 theme.palette.error.main,  
 ];  
  
 const valueFormatter = (item) => `${item.value}`;  
  
 return (  
 <Paper  
 sx={{  
 p: { xs: 1.5, md: 2 },  
 borderRadius: 3,  
 boxShadow: { xs: 1, md: 3 },  
 height: '100%',  
 m: { xs: 0, md: 2 },  
 display: 'flex',  
 flexDirection: 'column',  
 alignItems: 'center',  
 gap: 1,  
 minWidth: { xs: 260, md: 0 },  
 }}  
 >  
 <Typography fontWeight={700}>📊 Répartition des statuts</Typography>  
  
 <Box sx={{ position: 'relative', ...size }}>  
 <PieChart  
 colors={colors}  
 series={[  
 {  
 data,  
 innerRadius: '65%',  
 outerRadius: '100%',  
 arcLabelMinAngle: 35,  
 arcLabelRadius: '60%',  
 valueFormatter,  
 paddingAngle: 2,  
 cornerRadius: 4,  
 },  
 ]}  
 sx={{  
 [`& .${pieArcLabelClasses.root}`]: { fontWeight: 'bold' },  
 }}  
 {...size}  
 slotProps={{  
 legend: isMobile  
 ? { hidden: true }  
 : {  
 position: { vertical: 'bottom', horizontal: 'middle' },  
 direction: 'row',  
 },  
 }}  
 />  
  
 <Box  
 sx={{  
 position: 'absolute',  
 inset: 0,  
 display: 'flex',  
 alignItems: 'center',  
 justifyContent: 'center',  
 pointerEvents: 'none',  
 fontWeight: 800,  
 fontSize: 22,  
 }}  
 >  
 {total}  
 </Box>  
 </Box>  
 </Paper>  
 );  
}

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\components\dashboard\RecentActivity.jsx

// components/dashboard/RecentActivity.jsx  
import { useEffect, useState } from "react";  
import { Paper, Typography, List, ListItem, ListItemText, Box } from "@mui/material";  
import api from "../../api/axios";  
  
export default function RecentActivity() {  
 const [activities, setActivities] = useState([]);  
  
 useEffect(() => {  
 (async () => {  
 try {  
 const { data } = await api.get("/activity/recent");  
 setActivities(data || []);  
 } catch (e) {  
 console.error("Erreur activité récente", e);  
 setActivities([]);  
 }  
 })();  
 }, []);  
  
 return (  
 <Paper  
 sx={{  
 p: { xs: 1.5, md: 2 },  
 borderRadius: 3,  
 boxShadow: { xs: 1, md: 3 },  
 height: "100%",  
 m: { xs: 0, md: 2 },  
 minWidth: { xs: 260, md: 'auto' },  
 }}  
 >  
 <Typography fontWeight={700} sx={{ mb: 1 }}>🕒 Activité récente</Typography>  
 <Box sx={{ maxHeight: { xs: 180, md: 240 }, overflowY: "auto" }}>  
 <List dense>  
 {activities.slice(0, 8).map((a, idx) => (  
 <ListItem  
 key={idx}  
 sx={{ py: 0.5, borderRadius: 1, "&:hover": { bgcolor: "#fafafa" }, cursor: a.project\_id ? "pointer" : "default" }}  
 onClick={() => a.project\_id && window.location.assign(`/projects/${a.project\_id}`)}  
 >  
 <ListItemText  
 primary={a.text}  
 secondary={a.date ? new Date(a.date).toLocaleString("fr-FR") : "—"}  
 />  
 </ListItem>  
 ))}  
 {activities.length === 0 && <Typography color="text.secondary">Aucune activité récente</Typography>}  
 </List>  
 </Box>  
 </Paper>  
 );  
}

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\components\dashboard\TopInvestors.jsx

// components/dashboard/TopInvestors.jsx  
import { useEffect, useState } from "react";  
import { Paper, Typography, Table, TableHead, TableRow, TableCell, TableBody, Box } from "@mui/material";  
import api from "../../api/axios";  
  
export default function TopInvestors() {  
 const [rows, setRows] = useState([]);  
  
 useEffect(() => {  
 (async () => {  
 try {  
 // Endpoint attendu: GET /api/analytics/top-investors  
 // qui retourne: [{ id, name, project\_count, upcoming\_milestones, last\_activity }]  
 const { data } = await api.get("/analytics/top-investors");  
 setRows(data || []);  
 } catch (e) {  
 console.error("Erreur top investisseurs", e);  
 setRows([]);  
 }  
 })();  
 }, []);  
  
 return (  
 <Paper sx={{ p: 2, borderRadius: 3, boxShadow: 3, height: "100%", display: "flex", flexDirection: "column", m: 2 }}>  
 <Typography fontWeight={700} sx={{ mb: 1 }}>🏆 Top investisseurs</Typography>  
 <Box sx={{ flex: 1, minHeight: 0, overflow: "auto" }}>  
 <Table size="small" stickyHeader>  
 <TableHead>  
 <TableRow>  
 <TableCell>Investisseur</TableCell>  
 <TableCell align="right">Projets</TableCell>  
 <TableCell align="right">Jalons à venir</TableCell>  
 <TableCell align="right">Dernière activité</TableCell>  
 </TableRow>  
 </TableHead>  
 <TableBody>  
 {rows.slice(0, 6).map((r) => (  
 <TableRow  
 key={r.id}  
 hover  
 sx={{ cursor: "pointer" }}  
 onClick={() => window.location.assign(`/investors/${r.id}`)}  
 >  
 <TableCell>{r.name}</TableCell>  
 <TableCell align="right">{r.project\_count ?? 0}</TableCell>  
 <TableCell align="right">{r.upcoming\_milestones ?? 0}</TableCell>  
 <TableCell align="right">  
 {r.last\_activity ? new Date(r.last\_activity).toLocaleDateString("fr-FR") : "—"}  
 </TableCell>  
 </TableRow>  
 ))}  
 {rows.length === 0 && (  
 <TableRow><TableCell colSpan={4} align="center">Aucune donnée</TableCell></TableRow>  
 )}  
 </TableBody>  
 </Table>  
 </Box>  
 </Paper>  
 );  
}

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\components\dashboard\UpcomingMilestones.jsx

import { useEffect, useState } from "react";  
import { Paper, Typography, Chip, List, ListItem, ListItemText, Box } from "@mui/material";  
import api from "../../api/axios";  
  
function normalize(s) {  
 return String(s || "")  
 .toLowerCase()  
 .normalize("NFD").replace(/\p{Diacritic}/gu, "")  
 .replace(/\_/g, " ")  
 .trim();  
}  
function isClosedStatus(status) {  
 const n = normalize(status);  
 return n === "paye" || n === "payee" || n === "annule";  
}  
function startOfDay(d) {  
 const x = new Date(d);  
 x.setHours(0, 0, 0, 0);  
 return x;  
}  
  
export default function UpcomingMilestones() {  
 const [items, setItems] = useState([]);  
  
 useEffect(() => {  
 (async () => {  
 try {  
 const { data } = await api.get("/milestones/upcoming");  
 const arr = Array.isArray(data) ? data : [];  
 const today0 = startOfDay(new Date());  
 const filtered = arr  
 .filter(m => m?.due\_date)  
 .filter(m => startOfDay(m.due\_date) >= today0)  
 .filter(m => !isClosedStatus(m?.status))  
 .sort((a, b) => new Date(a.due\_date) - new Date(b.due\_date));  
 setItems(filtered.slice(0, 8));  
 } catch (e) {  
 console.error("Erreur chargement jalons", e);  
 setItems([]);  
 }  
 })();  
 }, []);  
  
 const today0 = startOfDay(new Date());  
  
 return (  
 <Paper  
 sx={{  
 p: { xs: 1.5, md: 2 },  
 borderRadius: 3,  
 boxShadow: { xs: 1, md: 3 },  
 height: "100%",  
 m: { xs: 0, md: 2 },  
 minWidth: { xs: 260, md: 'auto' },  
 }}  
 >  
 <Typography fontWeight={700} sx={{ mb: 1 }}>📅 Échéances à venir</Typography>  
 {items.length > 0 ? (  
 <List dense sx={{ maxHeight: { xs: 180, md: 'none' }, overflowY: { xs: 'auto', md: 'visible' } }}>  
 {items.map(m => {  
 const d = m.due\_date ? new Date(m.due\_date) : null;  
 const d0 = d ? startOfDay(d) : null;  
 const overdue = d0 && d0 < today0;  
 return (  
 <ListItem  
 key={m.id}  
 sx={{ py: 0.5, cursor: "pointer", "&:hover": { bgcolor: "#f5f5f5" } }}  
 onClick={() => window.location.assign(`/projects/${m.project\_id}`)}  
 secondaryAction={  
 d ? (  
 <Chip  
 size="small"  
 color={overdue ? "error" : "default"}  
 label={d.toLocaleDateString("fr-FR")}  
 />  
 ) : null  
 }  
 >  
 <ListItemText  
 primary={<b>{m.label}</b>}  
 secondary={`${m.project\_name ?? "Projet"} • ${m.investor\_name ?? "Investisseur"}`}  
 />  
 </ListItem>  
 );  
 })}  
 </List>  
 ) : (  
 <Box sx={{ mt: 1, color: "text.secondary" }}>Aucun jalon à venir 👍</Box>  
 )}  
 </Paper>  
 );  
}

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\components\layout\SidebarLayout.jsx

// src/components/layout/SidebarLayout.jsx  
import \* as React from "react";  
import { styled, useTheme } from "@mui/material/styles";  
import {  
 Box, CssBaseline, Toolbar, Typography, Divider, IconButton,  
 List, ListItem, ListItemButton, ListItemIcon, ListItemText,  
 useMediaQuery  
} from "@mui/material";  
import MuiDrawer from "@mui/material/Drawer";  
import MuiAppBar from "@mui/material/AppBar";  
import MenuIcon from "@mui/icons-material/Menu";  
import ChevronLeftIcon from "@mui/icons-material/ChevronLeft";  
import ChevronRightIcon from "@mui/icons-material/ChevronRight";  
import DashboardIcon from "@mui/icons-material/Dashboard";  
import ListAltIcon from "@mui/icons-material/ListAlt";  
import GroupsIcon from "@mui/icons-material/Groups";  
import { Link, useLocation, NavLink } from "react-router-dom";  
// import { AuthContext } from "../../contexts/AuthContext"; // si tu veux un bouton Déconnexion  
import MapIcon from "@mui/icons-material/Map";  
  
const drawerWidth = 220;  
  
const openedMixin = (theme) => ({  
 width: drawerWidth,  
 transition: theme.transitions.create("width", {  
 easing: theme.transitions.easing.sharp,  
 duration: theme.transitions.duration.enteringScreen,  
 }),  
 overflowX: "hidden",  
});  
  
const closedMixin = (theme) => ({  
 transition: theme.transitions.create("width", {  
 easing: theme.transitions.easing.sharp,  
 duration: theme.transitions.duration.leavingScreen,  
 }),  
 overflowX: "hidden",  
 width: `calc(${theme.spacing(7)} + 1px)`,  
 [theme.breakpoints.up("sm")]: {  
 width: `calc(${theme.spacing(8)} + 1px)`,  
 },  
});  
  
const DrawerHeader = styled("div")(({ theme }) => ({  
 display: "flex",  
 alignItems: "center",  
 justifyContent: "flex-end",  
 padding: theme.spacing(0, 1),  
 ...theme.mixins.toolbar,  
}));  
  
const AppBar = styled(MuiAppBar, {  
 shouldForwardProp: (prop) => prop !== "open" && prop !== "isMobile",  
})(({ theme, open, isMobile }) => ({  
 zIndex: theme.zIndex.drawer + 1,  
 transition: theme.transitions.create(["width", "margin"], {  
 easing: theme.transitions.easing.sharp,  
 duration: theme.transitions.duration.leavingScreen,  
 }),  
 ...(!isMobile && open && {  
 marginLeft: drawerWidth,  
 width: `calc(100% - ${drawerWidth}px)`,  
 transition: theme.transitions.create(["width", "margin"], {  
 easing: theme.transitions.easing.sharp,  
 duration: theme.transitions.duration.enteringScreen,  
 }),  
 }),  
}));  
  
const Drawer = styled(MuiDrawer, { shouldForwardProp: (prop) => prop !== "open" })(  
 ({ theme, open }) => ({  
 width: drawerWidth,  
 flexShrink: 0,  
 whiteSpace: "nowrap",  
 boxSizing: "border-box",  
 ...(open && {  
 ...openedMixin(theme),  
 "& .MuiDrawer-paper": openedMixin(theme),  
 }),  
 ...(!open && {  
 ...closedMixin(theme),  
 "& .MuiDrawer-paper": closedMixin(theme),  
 }),  
 })  
);  
  
export default function SidebarLayout({ children }) {  
 const theme = useTheme();  
 const isMobile = useMediaQuery(theme.breakpoints.down("md"));  
 const location = useLocation();  
 // const { logout, user } = React.useContext(AuthContext); // si besoin  
  
 // état pour desktop (mini-variant) et mobile (temporary)  
 const [openDesktop, setOpenDesktop] = React.useState(true);  
 const [openMobile, setOpenMobile] = React.useState(false);  
  
 const toggleDrawer = () => {  
 if (isMobile) setOpenMobile((s) => !s);  
 else setOpenDesktop((s) => !s);  
 };  
  
 const navItems = [  
 { to: "/projects", label: "Dashboard", icon: <DashboardIcon /> },  
 { to: "/projects-list", label: "Projets", icon: <ListAltIcon /> },  
 { to: "/investors-list", label: "Investisseurs", icon: <GroupsIcon /> },  
 { to: "/carte", label: "Carte des Projets", icon: <MapIcon /> },  
 ];  
  
 // marge gauche du contenu pour ne pas passer sous le drawer (desktop uniquement)  
 const contentMarginLeft = (theme) =>  
 isMobile  
 ? 0  
 : openDesktop  
 ? `${drawerWidth}px`  
 : `calc(${theme.spacing(8)} + 1px)`; // largeur mini fermée (sm et +)  
  
 return (  
 <Box sx={{ display: "flex", minHeight: "100vh", bgcolor: "#f6f8fb" }}>  
 <CssBaseline />  
  
 <AppBar position="fixed" open={openDesktop} isMobile={isMobile} color="primary">  
 <Toolbar sx={{ minHeight: 64 }}>  
 <IconButton  
 color="inherit"  
 aria-label="open drawer"  
 onClick={toggleDrawer}  
 edge="start"  
 sx={{ mr: 2 }}  
 >  
 <MenuIcon />  
 </IconButton>  
 <Typography variant="h6" noWrap sx={{ fontWeight: 800, flex: 1 }}>  
 🌞 Gestion des Projets Solaires  
 </Typography>  
  
 {/\* Exemple: bouton logout à droite (décommente si nécessaire)  
 <Button color="inherit" onClick={logout}>Déconnexion</Button>  
 \*/}  
 </Toolbar>  
 </AppBar>  
  
 {/\* Drawer permanent en desktop (mini variant), temporaire en mobile \*/}  
 {isMobile ? (  
 <MuiDrawer  
 variant="temporary"  
 open={openMobile}  
 onClose={() => setOpenMobile(false)}  
 ModalProps={{ keepMounted: true }}  
 PaperProps={{ sx: { width: drawerWidth } }}  
 >  
 <DrawerHeader>  
 <IconButton onClick={() => setOpenMobile(false)}>  
 {theme.direction === "rtl" ? <ChevronRightIcon /> : <ChevronLeftIcon />}  
 </IconButton>  
 </DrawerHeader>  
 <Divider />  
 <List>  
 {navItems.map((item) => (  
 <ListItem key={item.to} disablePadding>  
 <ListItemButton  
 component={Link}  
 to={item.to}  
 selected={location.pathname === item.to}  
 onClick={() => setOpenMobile(false)}  
 >  
 <ListItemIcon>{item.icon}</ListItemIcon>  
 <ListItemText primary={item.label} />  
 </ListItemButton>  
 </ListItem>  
 ))}  
 </List>  
 </MuiDrawer>  
 ) : (  
 <Drawer variant="permanent" open={openDesktop}>  
 <DrawerHeader>  
 <IconButton onClick={toggleDrawer}>  
 {theme.direction === "rtl" ? <ChevronRightIcon /> : <ChevronLeftIcon />}  
 </IconButton>  
 </DrawerHeader>  
 <Divider />  
 <List  
 sx={{  
 // style pour l’état actif  
 "& .MuiButtonBase-root.active": {  
 bgcolor: "action.selected",  
 "&:hover": { bgcolor: "action.selected" },  
 },  
 }}  
 >  
 {navItems.map(({ to, label, icon }) => (  
 <ListItem key={to} disablePadding sx={{ display: "block" }}>  
 <ListItemButton  
 component={NavLink}  
 to={to}  
 // NavLink -> ajoute la classe "active" automatiquement  
 sx={[  
 { minHeight: 48, px: 2.5 },  
 open ? { justifyContent: "initial" } : { justifyContent: "center" },  
 ]}  
 >  
 <ListItemIcon  
 sx={[  
 { minWidth: 0, justifyContent: "center" },  
 open ? { mr: 3 } : { mr: "auto" },  
 ]}  
 >  
 {icon}  
 </ListItemIcon>  
 <ListItemText  
 primary={label}  
 sx={[open ? { opacity: 1 } : { opacity: 0 }]}  
 />  
 </ListItemButton>  
 </ListItem>  
 ))}  
 </List>  
 </Drawer>  
 )}  
  
 {/\* Contenu \*/}  
 <Box  
 component="main"  
 sx={(theme) => ({  
 flexGrow: 1,  
 width: "100%",  
 minWidth: 0,  
 boxSizing: "border-box",  
 p: { xs: 2, md: 3 },  
 ml: isMobile  
 ? 0  
 : openDesktop  
 ? `0px`  
 : `(0px + 1px)`, // largeur rail mini  
 transition: theme.transitions.create("margin-left", {  
 easing: theme.transitions.easing.sharp,  
 duration: theme.transitions.duration.shorter,  
 }),  
 })}  
 >  
 <Toolbar />  
 {children}  
 </Box>  
  
 </Box>  
 );  
}

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\components\project\CustomFieldsCard.jsx

import { Paper, Typography, Table, TableHead, TableRow, TableCell, TableBody, IconButton, TextField } from "@mui/material";  
import EditIcon from "@mui/icons-material/Edit";  
import DeleteIcon from "@mui/icons-material/Delete";  
import AddIcon from "@mui/icons-material/Add";  
import { useState } from "react";  
  
/\*\*  
 \* Props:  
 \* - dense?: boolean (par défaut true) -> compresse les lignes au max  
 \* - truncate?: boolean (par défaut true) -> tronque les longues valeurs avec ellipsis  
 \*/  
export default function CustomFieldsCard({ fields, onAdd, onEdit, onDelete, dense = true, truncate = true }) {  
 const [form, setForm] = useState({ name: "", value: "" });  
 const [editId, setEditId] = useState(null);  
  
 async function submit() {  
 if (!form.name) return;  
 if (editId) {  
 await onEdit(editId, form.name, form.value);  
 setEditId(null);  
 } else {  
 await onAdd(form.name, form.value);  
 }  
 setForm({ name: "", value: "" });  
 }  
  
 const compactSx = dense  
 ? {  
 // Réduit la hauteur et les paddings partout dans le tableau  
 "& .MuiTableCell-root": { py: 0.25, px: 0.75, lineHeight: 1.25 },  
 "& .MuiTableRow-root": { height: 34 }, // ~ compact  
 "& .MuiInputBase-input": { py: 0.3 }, // amincit les TextField  
 "& .MuiIconButton-root": { p: 0.25 }, // icônes plus serrées  
 }  
 : undefined;  
  
 const cellTruncateSx = truncate  
 ? {  
 maxWidth: 240,  
 whiteSpace: "nowrap",  
 overflow: "hidden",  
 textOverflow: "ellipsis",  
 }  
 : undefined;  
  
 return (  
 <Paper elevation={1} sx={{ p: 2, borderRadius: 2 }}>  
 <Typography variant="subtitle2" sx={{ mb: 1, fontSize: 13 }}>Infos personnalisées</Typography>  
  
 <Table size="small" sx={compactSx}>  
 <TableHead>  
 <TableRow>  
 <TableCell sx={{ fontWeight: 700, fontSize: 12 }}>Nom</TableCell>  
 <TableCell sx={{ fontWeight: 700, fontSize: 12 }}>Valeur</TableCell>  
 <TableCell sx={{ width: 40 }} />  
 </TableRow>  
 </TableHead>  
  
 <TableBody>  
 {fields.map((f) => (  
 <TableRow key={f.id} hover>  
 <TableCell sx={cellTruncateSx} title={truncate ? f.field\_name : undefined}>  
 {f.field\_name}  
 </TableCell>  
 <TableCell sx={cellTruncateSx} title={truncate ? f.field\_value : undefined}>  
 {f.field\_value}  
 </TableCell>  
 <TableCell align="right">  
 <IconButton  
 size="small"  
 onClick={() => {  
 setEditId(f.id);  
 setForm({ name: f.field\_name, value: f.field\_value });  
 }}  
 >  
 <EditIcon fontSize="inherit" />  
 </IconButton>  
 <IconButton size="small" onClick={() => onDelete(f.id)}>  
 <DeleteIcon fontSize="inherit" />  
 </IconButton>  
 </TableCell>  
 </TableRow>  
 ))}  
  
 {/\* Ligne d'édition/ajout \*/}  
 <TableRow>  
 <TableCell>  
 <TextField  
 size="small"  
 placeholder="Nom du champ"  
 fullWidth  
 value={form.name}  
 onChange={(e) => setForm((s) => ({ ...s, name: e.target.value }))}  
 inputProps={{ "aria-label": "Nom du champ" }}  
 />  
 </TableCell>  
 <TableCell>  
 <TextField  
 size="small"  
 placeholder="Valeur"  
 fullWidth  
 value={form.value}  
 onChange={(e) => setForm((s) => ({ ...s, value: e.target.value }))}  
 inputProps={{ "aria-label": "Valeur" }}  
 />  
 </TableCell>  
 <TableCell align="right">  
 <IconButton size="small" color="primary" onClick={submit} aria-label={editId ? "Enregistrer" : "Ajouter"}>  
 <AddIcon fontSize="inherit" />  
 </IconButton>  
 </TableCell>  
 </TableRow>  
 </TableBody>  
 </Table>  
 </Paper>  
 );  
}

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\components\project\InvestorsCard.jsx

import { useEffect, useState } from "react";  
import {  
 Paper, Typography, Box, TextField, Button, Card, Stack, IconButton, Divider, Chip,  
 Dialog, DialogTitle, DialogContent, DialogActions  
} from "@mui/material";  
import DeleteIcon from "@mui/icons-material/Delete";  
import RefreshIcon from "@mui/icons-material/Refresh";  
import AddIcon from "@mui/icons-material/Add";  
import {  
 listInvestors, listProjectInvestors, attachInvestor, detachInvestor,  
 listMilestones, addMilestone, createInvestor, deleteMilestone,  
 STATUS\_MAP, REVERSE\_STATUS\_MAP  
} from "../../api/investors";  
import api from "../../api/axios"; // pour les fallback PUT/PATCH  
  
// rôle fixe  
const FIXED\_ROLE = "investisseurs";  
  
/\*\* Normalise une chaine: minuscule, sans accents, remplace \_ par espace \*/  
function normalize(s) {  
 return String(s || "")  
 .toLowerCase()  
 .normalize("NFD").replace(/\p{Diacritic}/gu, "") // retire accents  
 .replace(/\_/g, " ")  
 .replace(/\s+/g, " ")  
 .trim();  
}  
  
// Couleur basée sur le statut (accepte formats internes ET libellés FR)  
function getStepColor(status) {  
 const n = normalize(status);  
 if (n === "payee" || n === "paye") return "success";  
 if (n === "annule") return "error";  
 if (n === "en cours") return "info";  
 if (n === "pas payee" || n === "pas paye") return "warning";  
 return "default";  
}  
  
// Convertit n'importe quelle forme vers la valeur interne si possible  
// "Payé" -> "payee", "payee" -> "payee"  
function internalFromAny(s) {  
 return STATUS\_MAP[s] || s;  
}  
  
// Envoie le statut attendu par la DB (libellé FR exact)  
function toDbStatus(internalOrFr) {  
 return REVERSE\_STATUS\_MAP[internalOrFr] || internalOrFr; // "en\_cours" -> "En cours"  
}  
  
/\*\* Fallback intelligent pour update jalon quand le back n'a pas la route nested en PUT \*/  
async function updateMilestoneSmart(projectId, investorId, milestoneId, statusInternal) {  
 const status = toDbStatus(statusInternal); // toujours libellé FR pour l'ENUM DB  
  
 // 1) Essai route nested en PUT  
 try {  
 await api.put(`/projects/${projectId}/investors/${investorId}/milestones/${milestoneId}`, { status });  
 return;  
 } catch (e) {  
 const statusCode = e?.response?.status;  
 const body = e?.response?.data;  
 const bodyText = typeof body === "string" ? body : JSON.stringify(body || {});  
 const isRouteMissing = statusCode === 404 || bodyText.includes("Cannot PUT");  
 if (!isRouteMissing) {  
 // peut être 400 (validation), etc. -> rethrow  
 throw e;  
 }  
 }  
  
 // 2) Essai route flat en PUT  
 try {  
 await api.put(`/milestones/${milestoneId}`, { status, project\_id: projectId, investor\_id: investorId });  
 return;  
 } catch (e) {  
 const statusCode = e?.response?.status;  
 const body = e?.response?.data;  
 const bodyText = typeof body === "string" ? body : JSON.stringify(body || {});  
 const isRouteMissing = statusCode === 404 || bodyText.includes("Cannot PUT");  
 if (!isRouteMissing) {  
 throw e;  
 }  
 }  
  
 // 3) Essai nested en PATCH  
 try {  
 await api.patch(`/projects/${projectId}/investors/${investorId}/milestones/${milestoneId}`, { status });  
 return;  
 } catch (e) {  
 const statusCode = e?.response?.status;  
 const body = e?.response?.data;  
 const bodyText = typeof body === "string" ? body : JSON.stringify(body || {});  
 const isRouteMissing = statusCode === 404 || bodyText.includes("Cannot PATCH");  
 if (!isRouteMissing) {  
 throw e;  
 }  
 }  
  
 // 4) Essai flat en PATCH  
 await api.patch(`/milestones/${milestoneId}`, { status, project\_id: projectId, investor\_id: investorId });  
}  
  
export default function InvestorsCard({ projectId }) {  
 const [allInvestors, setAllInvestors] = useState([]);  
 const [projInvestors, setProjInvestors] = useState([]);  
 const [selectedInvestorId, setSelectedInvestorId] = useState("");  
 const [msByInvestor, setMsByInvestor] = useState({});  
  
 // Dialog création investisseur  
 const [createOpen, setCreateOpen] = useState(false);  
 const [newInv, setNewInv] = useState({ name: "", company: "", email: "", phone: "", notes: "" });  
  
 // Dialog ajout jalon  
 const [msDialogOpenFor, setMsDialogOpenFor] = useState(null); // id investisseur ou null  
 const [msForm, setMsForm] = useState({  
 label: "",  
 due\_date: "",  
 status: STATUS\_MAP["En cours"], // valeur interne  
 comment: ""  
 });  
  
 // Charger investisseurs du compte + investisseurs du projet  
 useEffect(() => {  
 (async () => {  
 setAllInvestors(await listInvestors());  
 const proj = await listProjectInvestors(projectId);  
 setProjInvestors(proj);  
 })();  
 }, [projectId]);  
  
 // Précharger tous les jalons dès qu'on a la liste des investisseurs du projet  
 useEffect(() => {  
 if (!projInvestors.length) return;  
 let cancelled = false;  
 (async () => {  
 try {  
 const entries = await Promise.all(  
 projInvestors.map(async (inv) => {  
 try {  
 const items = await listMilestones(projectId, inv.id);  
 return [inv.id, items];  
 } catch (e) {  
 console.error("Préchargement jalons échoué pour", inv.id, e);  
 return [inv.id, []];  
 }  
 })  
 );  
 if (!cancelled) {  
 setMsByInvestor((prev) => {  
 const next = { ...prev };  
 for (const [id, items] of entries) next[id] = items;  
 return next;  
 });  
 }  
 } catch (e) {  
 console.error("Préchargement jalons (global) échoué:", e);  
 }  
 })();  
 return () => { cancelled = true; };  
 }, [projectId, projInvestors]);  
  
 async function refreshMilestones(invId) {  
 const items = await listMilestones(projectId, invId);  
 setMsByInvestor(s => ({ ...s, [invId]: items }));  
 }  
  
 function openMsDialog(invId) {  
 setMsDialogOpenFor(invId);  
 setMsForm({  
 label: "",  
 due\_date: "",  
 status: STATUS\_MAP["En cours"], // interne  
 comment: ""  
 });  
 }  
  
 async function submitMilestone() {  
 if (!msDialogOpenFor || !msForm.label) return;  
 const invId = msDialogOpenFor; // capture avant fermeture  
 try {  
 await addMilestone(projectId, invId, msForm); // mapping vers libellé FR fait côté API  
 setMsDialogOpenFor(null);  
 await refreshMilestones(invId);  
 } catch (err) {  
 console.error("Erreur ajout jalon:", err);  
 alert("Impossible d'ajouter ce jalon.");  
 }  
 }  
  
 return (  
 <Paper elevation={1} sx={{ p: 2, borderRadius: 2 }}>  
 {/\* Sélection + bouton + pour ouvrir le dialog de création \*/}  
 <Box sx={{ display: "flex", gap: 1, flexWrap: "wrap", mb: 1, alignItems: "center" }}>  
 <TextField  
 select size="small" label="Investisseur" value={selectedInvestorId}  
 onChange={e => setSelectedInvestorId(e.target.value)} sx={{ minWidth: 260 }} SelectProps={{ native: true }}  
 >  
 <option value=""></option>  
 {allInvestors.map(i => (  
 <option key={i.id} value={i.id}>  
 {i.name}{i.company ? ` (${i.company})` : ""}  
 </option>  
 ))}  
 </TextField>  
  
 <IconButton  
 aria-label="Ajouter un investisseur"  
 size="small"  
 color="primary"  
 onClick={() => setCreateOpen(true)}  
 sx={{ border: '1px solid', borderColor: 'primary.main', borderRadius: 1 }}  
 >  
 <AddIcon fontSize="small" />  
 </IconButton>  
  
 <Button  
 size="small" variant="contained"  
 onClick={async () => {  
 if (!selectedInvestorId) return;  
 await attachInvestor(projectId, selectedInvestorId, FIXED\_ROLE);  
 const fresh = await listProjectInvestors(projectId);  
 setProjInvestors(fresh);  
 // précharger les jalons de l’investisseur attaché  
 try { await refreshMilestones(selectedInvestorId); } catch {}  
 setSelectedInvestorId("");  
 }}  
 >  
 Attribuer  
 </Button>  
 </Box>  
  
 {/\* --- Dialog création investisseur --- \*/}  
 <Dialog open={createOpen} onClose={() => setCreateOpen(false)} fullWidth maxWidth="sm">  
 <DialogTitle>Créer un investisseur</DialogTitle>  
 <DialogContent sx={{ pt: 2, display: "grid", gap: 2 }}>  
 <TextField  
 autoFocus  
 label="Nom \*"  
 required  
 size="small"  
 value={newInv.name}  
 onChange={e => setNewInv(v => ({ ...v, name: e.target.value }))}  
 />  
 <TextField  
 label="Société"  
 size="small"  
 value={newInv.company}  
 onChange={e => setNewInv(v => ({ ...v, company: e.target.value }))}  
 />  
 <TextField  
 label="Email"  
 size="small"  
 type="email"  
 value={newInv.email}  
 onChange={e => setNewInv(v => ({ ...v, email: e.target.value }))}  
 />  
 <TextField  
 label="Téléphone"  
 size="small"  
 value={newInv.phone}  
 onChange={e => setNewInv(v => ({ ...v, phone: e.target.value }))}  
 />  
 <TextField  
 label="Notes"  
 size="small"  
 multiline  
 minRows={2}  
 value={newInv.notes}  
 onChange={e => setNewInv(v => ({ ...v, notes: e.target.value }))}  
 />  
 </DialogContent>  
 <DialogActions sx={{ px: 3, pb: 2 }}>  
 <Button onClick={() => setCreateOpen(false)}>Annuler</Button>  
 <Button  
 variant="contained"  
 onClick={async () => {  
 if (!newInv.name) return;  
 const created = await createInvestor(newInv);  
 const fresh = await listInvestors();  
 setAllInvestors(fresh);  
 const newId = created?.id ?? fresh.find(i => i.email === newInv.email && i.name === newInv.name)?.id;  
 if (newId) setSelectedInvestorId(String(newId));  
 setNewInv({ name: "", company: "", email: "", phone: "", notes: "" });  
 setCreateOpen(false);  
 }}  
 >  
 Créer  
 </Button>  
 </DialogActions>  
 </Dialog>  
  
 {/\* Liste investisseurs + jalons \*/}  
 <Stack spacing={1}>  
 {projInvestors.map(inv => (  
 <Card key={inv.id} sx={{ p: 1.5 }}>  
 <Box sx={{ display: "flex", justifyContent: "space-between", alignItems: "center" }}>  
 <Box>  
 <Typography fontWeight={700}>{inv.name}{inv.company && ` — ${inv.company}`}</Typography>  
 <Typography variant="caption">  
 {inv.email || "—"} • {inv.phone || "—"} • rôle: {FIXED\_ROLE}  
 </Typography>  
 </Box>  
 <Button color="error" size="small" onClick={async () => {  
 await detachInvestor(projectId, inv.id);  
 setProjInvestors(await listProjectInvestors(projectId));  
 // enlever ses jalons du state  
 setMsByInvestor(s => {  
 const n = { ...s };  
 delete n[inv.id];  
 return n;  
 });  
 }}>  
 Retirer  
 </Button>  
 </Box>  
  
 <Divider sx={{ my: 1 }} />  
  
 <Box sx={{ display: "flex", alignItems: "center", gap: 1 }}>  
 <Typography variant="subtitle2">Jalons</Typography>  
 <IconButton size="small" onClick={() => refreshMilestones(inv.id)}>  
 <RefreshIcon fontSize="small" />  
 </IconButton>  
 <Box sx={{ flex: 1 }} />  
 <Button size="small" startIcon={<AddIcon />} onClick={() => openMsDialog(inv.id)}>  
 Ajouter un jalon  
 </Button>  
 </Box>  
  
 <Stack spacing={0.5} sx={{ my: 1 }}>  
 {(msByInvestor[inv.id] || []).map(m => {  
 const internal = internalFromAny(m.status); // interne pour select/couleur  
 const labelFR = REVERSE\_STATUS\_MAP[internal] || m.status; // libellé pour chip  
  
 return (  
 <Box key={m.id} sx={{ display: "flex", gap: 8, alignItems: "center", flexWrap: "wrap" }}>  
 <Box sx={{ display: "flex", alignItems: "center", gap: 1, minWidth: 260 }}>  
 <Chip size="small" label={labelFR} color={getStepColor(internal)} />  
 <Typography variant="body2" fontWeight={600}>{m.label}</Typography>  
 <Typography variant="caption" sx={{ color: "#777" }}>  
 {m.due\_date ? new Date(m.due\_date).toLocaleDateString('fr-FR') : ''}  
 </Typography>  
 {m.comment && <Typography variant="caption" sx={{ fontStyle: "italic" }}>{m.comment}</Typography>}  
 </Box>  
  
 {/\* Select pour modifier le statut (optimistic UI + fallback routes) \*/}  
 <TextField  
 label="Statut"  
 select  
 size="small"  
 value={internal}  
 onChange={async (e) => {  
 const newInternal = e.target.value;  
  
 // optimistic UI  
 setMsByInvestor(prev => {  
 const next = { ...prev };  
 next[inv.id] = (next[inv.id] || []).map(x =>  
 x.id === m.id ? { ...x, status: newInternal } : x  
 );  
 return next;  
 });  
  
 try {  
 await updateMilestoneSmart(projectId, inv.id, m.id, newInternal);  
 // sécurise: recharge après succès  
 await refreshMilestones(inv.id);  
 } catch (err) {  
 console.error("Maj statut jalon:", err);  
 alert("Impossible de modifier le statut de ce jalon.");  
 // rollback si erreur  
 setMsByInvestor(prev => {  
 const next = { ...prev };  
 next[inv.id] = (next[inv.id] || []).map(x =>  
 x.id === m.id ? { ...x, status: internal } : x  
 );  
 return next;  
 });  
 }  
 }}  
 SelectProps={{ native: true }}  
 sx={{ minWidth: 160 }}  
 >  
 {Object.entries(STATUS\_MAP).map(([label, value]) => (  
 <option key={value} value={value}>{label}</option>  
 ))}  
 </TextField>  
  
 <IconButton  
 size="small"  
 color="error"  
 onClick={async () => {  
 try {  
 await deleteMilestone(projectId, inv.id, m.id);  
 refreshMilestones(inv.id);  
 } catch (err) {  
 console.error("Erreur suppression jalon:", err);  
 alert("Impossible de supprimer ce jalon.");  
 }  
 }}  
 >  
 <DeleteIcon fontSize="small" />  
 </IconButton>  
 </Box>  
 );  
 })}  
 </Stack>  
 </Card>  
 ))}  
 </Stack>  
  
 {/\* Dialog ajout jalon \*/}  
 <Dialog  
 open={!!msDialogOpenFor}  
 onClose={() => setMsDialogOpenFor(null)}  
 fullWidth  
 maxWidth="sm"  
 >  
 <DialogTitle>Ajouter un jalon</DialogTitle>  
 <DialogContent sx={{ pt: 2, display: "grid", gap: 2 }}>  
 <TextField  
 label="Libellé \*"  
 required  
 size="small"  
 value={msForm.label}  
 onChange={e => setMsForm(f => ({ ...f, label: e.target.value }))}  
 />  
 <TextField  
 label="Échéance"  
 type="date"  
 size="small"  
 value={msForm.due\_date}  
 onChange={e => setMsForm(f => ({ ...f, due\_date: e.target.value }))}  
 InputLabelProps={{ shrink: true }}  
 />  
 <TextField  
 label="Statut"  
 select  
 size="small"  
 value={msForm.status} // interne  
 onChange={e => setMsForm(f => ({ ...f, status: e.target.value }))}  
 SelectProps={{ native: true }}  
 >  
 {Object.entries(STATUS\_MAP).map(([label, value]) => (  
 <option key={value} value={value}>{label}</option>  
 ))}  
 </TextField>  
 <TextField  
 label="Commentaire"  
 size="small"  
 value={msForm.comment}  
 onChange={e => setMsForm(f => ({ ...f, comment: e.target.value }))}  
 />  
 </DialogContent>  
 <DialogActions sx={{ px: 3, pb: 2 }}>  
 <Button onClick={() => setMsDialogOpenFor(null)}>Annuler</Button>  
 <Button variant="contained" color="success" onClick={submitMilestone}>  
 Ajouter  
 </Button>  
 </DialogActions>  
 </Dialog>  
 </Paper>  
 );  
}

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\components\project\ProjectHeaderBar.jsx

import { Paper, Box, Typography, Chip, Select, MenuItem, Button, IconButton, Tooltip } from "@mui/material";  
import ShareIcon from "@mui/icons-material/Share";  
  
export default function ProjectHeaderBar({  
 project,  
 user,  
 onEditClick,  
 onStatusChange,  
 statusUpdating,  
 onShareClick, // ⬅️ nouvelle prop  
}) {  
 const statusList = ["En cours", "Terminée", "Annulée"];  
  
 return (  
 <Paper elevation={1} sx={{ p: 2, mb: 2, borderRadius: 2 }}>  
 <Box  
 sx={{  
 display: "flex",  
 alignItems: "center",  
 justifyContent: "space-between",  
 gap: 2,  
 flexWrap: "wrap",  
 }}  
 >  
 {/\* Titre + statut \*/}  
 <Box sx={{ display: "flex", alignItems: "center", gap: 1, flexWrap: "wrap" }}>  
 <Typography variant="h6" fontWeight={800}>{project.name}</Typography>  
 {project.type && <Chip label={project.type} color="info" sx={{ fontWeight: 700 }} />}  
  
 <Select  
 value={project.status || "En cours"}  
 onChange={onStatusChange}  
 size="small"  
 sx={{ ml: 0.5, fontWeight: 700, minWidth: 120, bgcolor: "#f6f7fa" }}  
 disabled={statusUpdating}  
 >  
 {statusList.map((s) => (  
 <MenuItem key={s} value={s}>{s}</MenuItem>  
 ))}  
 </Select>  
 </Box>  
  
 {/\* Actions à droite \*/}  
 <Box sx={{ display: "flex", alignItems: "center", gap: 1 }}>  
 <Tooltip title="Partager le projet">  
 <IconButton  
 size="small"  
 onClick={onShareClick}  
 aria-label="Partager le projet"  
 >  
 <ShareIcon />  
 </IconButton>  
 </Tooltip>  
  
 {user?.role === "admin" && (  
 <Button variant="outlined" onClick={onEditClick} size="small">  
 Modifier  
 </Button>  
 )}  
 </Box>  
 </Box>  
  
 {/\* Infos clés \*/}  
 <Box  
 sx={{  
 mt: 1,  
 display: "grid",  
 gridTemplateColumns: { xs: "1fr", sm: "1fr 1fr" },  
 columnGap: 3,  
 rowGap: 0.5,  
 }}  
 >  
 <Typography variant="body2"><b>Adresse :</b> {project.address || "—"}</Typography>  
 <Typography variant="body2"><b>Puissance :</b> {project.power ? `${project.power} kWc` : "—"}</Typography>  
 <Typography variant="body2"><b>Téléphone :</b> {project.phone || "—"}</Typography>  
 <Typography variant="body2"><b>Email :</b> {project.email || "—"}</Typography>  
 </Box>  
  
 {project.comments && (  
 <Typography variant="body2" sx={{ mt: 1 }}>  
 <b>Commentaires :</b><br />  
 <span style={{ whiteSpace: "pre-line" }}>{project.comments}</span>  
 </Typography>  
 )}  
 </Paper>  
 );  
}

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\components\project\ShareCard.jsx

// src/components/project/ShareCard.jsx  
import { useState } from "react";  
import { Paper, Typography, Box, TextField, Button } from "@mui/material";  
  
export default function ShareCard({ onInvite }) {  
 const [email, setEmail] = useState("");  
  
 async function handleInvite(e) {  
 e.preventDefault();  
 if (!email) return;  
 await onInvite(email);  
 setEmail("");  
 }  
  
 return (  
 <Paper sx={{ p: 2, borderRadius: 2 }}>  
 <Typography variant="subtitle2" sx={{ mb: 1 }}>Partager le projet</Typography>  
  
 <Box component="form" onSubmit={handleInvite} sx={{ display: "flex", gap: 1 }}>  
 <TextField  
 size="small"  
 label="Email"  
 value={email}  
 onChange={(e) => setEmail(e.target.value)}  
 type="email"  
 />  
 <Button type="submit" size="small" variant="contained">Inviter</Button>  
 </Box>  
 </Paper>  
 );  
}

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\components\project\StepsCard.jsx

import { Paper, Typography, Box, Stack, Chip, IconButton, Button, Dialog, DialogTitle, DialogContent, DialogActions, TextField, MenuItem } from "@mui/material";  
import DeleteIcon from "@mui/icons-material/Delete";  
import AddIcon from "@mui/icons-material/Add";  
import { useState } from "react";  
  
function chipColor(s) {  
 const v = (s || "").toLowerCase();  
 if (v.includes("ok")) return "success";  
 if (v.includes("annul")) return "error";  
 if (v.includes("cours")) return "info";  
 return "default";  
}  
  
export default function StepsCard({ steps, onAdd, onDelete }) {  
 const [open, setOpen] = useState(false);  
 const [form, setForm] = useState({ label: "", date: "", status: "En Cours", comment: "" });  
  
 async function submit(e) {  
 e.preventDefault();  
 await onAdd({ label: form.label, step\_date: form.date, status: form.status, step\_comment: form.comment });  
 setForm({ label: "", date: "", status: "En Cours", comment: "" });  
 setOpen(false);  
 }  
  
 return (  
 <Paper elevation={1} sx={{ p: 2, borderRadius: 2 }}>  
 <Typography variant="subtitle2" sx={{ mb: 1 }}>Suivi du projet</Typography>  
 <Box sx={{ bgcolor: "#f6f7fb", borderRadius: 1, p: 1, mb: 1, maxHeight: 220, overflowY: "auto" }}>  
 <Stack spacing={1}>  
 {steps.map(step => (  
 <Box key={step.id} sx={{ display: "flex", alignItems: "center", gap: 1 }}>  
 <Chip size="small" label={step.status || "En Cours"} color={chipColor(step.status)} />  
 <Box sx={{ flex: 1 }}>  
 <Typography variant="body2" fontWeight={600} sx={{ display: "inline", mr: 1 }}>{step.label}</Typography>  
 <Typography variant="caption" sx={{ color: "#888", ml: 1 }}>  
 {step.step\_date ? new Date(step.step\_date).toLocaleDateString("fr-FR") : ""}  
 </Typography>  
 {step.step\_comment && <Typography variant="caption" sx={{ ml: 2, fontStyle: "italic", color: "#888" }}>{step.step\_comment}</Typography>}  
 </Box>  
 <IconButton size="small" color="error" onClick={() => onDelete(step.id)}>  
 <DeleteIcon fontSize="small" />  
 </IconButton>  
 </Box>  
 ))}  
 </Stack>  
 </Box>  
 <Button size="small" variant="contained" startIcon={<AddIcon />} onClick={() => setOpen(true)}>  
 Ajouter une étape  
 </Button>  
  
 <Dialog open={open} onClose={() => setOpen(false)} fullWidth maxWidth="sm">  
 <DialogTitle>Ajouter une étape</DialogTitle>  
 <Box component="form" onSubmit={submit}>  
 <DialogContent sx={{ display: "grid", gap: 2 }}>  
 <TextField label="Libellé \*" value={form.label} onChange={e => setForm(f => ({ ...f, label: e.target.value }))} required />  
 <TextField label="Date \*" type="date" value={form.date} onChange={e => setForm(f => ({ ...f, date: e.target.value }))} InputLabelProps={{ shrink: true }} required />  
 <TextField select label="Statut" value={form.status} onChange={e => setForm(f => ({ ...f, status: e.target.value }))}>  
 <MenuItem value="OK">OK</MenuItem>  
 <MenuItem value="En Cours">En Cours</MenuItem>  
 <MenuItem value="Annulé">Annulé</MenuItem>  
 </TextField>  
 <TextField label="Commentaire" value={form.comment} onChange={e => setForm(f => ({ ...f, comment: e.target.value }))} />  
 </DialogContent>  
 <DialogActions>  
 <Button onClick={() => setOpen(false)}>Annuler</Button>  
 <Button type="submit" variant="contained" color="success">Ajouter</Button>  
 </DialogActions>  
 </Box>  
 </Dialog>  
 </Paper>  
 );  
}

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\config\app.js

export const APP\_NAME = import.meta.env.VITE\_APP\_NAME || 'Gestion des Projets Solaires';

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\contexts\AuthContext.jsx

import { createContext, useState, useEffect } from "react";  
import { setAuthToken } from "../api/axios";  
  
export const AuthContext = createContext();  
  
export function AuthProvider({ children }) {  
 const [token, setToken] = useState(localStorage.getItem("token") || null);  
 const [user, setUser] = useState(() => {  
 const saved = localStorage.getItem("user");  
 return saved ? JSON.parse(saved) : null;  
 });  
  
 useEffect(() => {  
 setAuthToken(token);  
 }, [token]);  
  
 const login = (token, user) => {  
 setToken(token);  
 setUser(user);  
 setAuthToken(token);  
 localStorage.setItem("token", token);  
 localStorage.setItem("user", JSON.stringify(user));  
 };  
  
 const logout = () => {  
 setToken(null);  
 setUser(null);  
 setAuthToken(null);  
 localStorage.removeItem("token");  
 localStorage.removeItem("user");  
 };  
  
 return (  
 <AuthContext.Provider value={{ token, user, login, logout }}>  
 {children}  
 </AuthContext.Provider>  
 );  
}

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\pages\CarteDesProjets.jsx

// src/pages/CarteDesProjets.jsx  
import { useEffect, useState } from "react";  
import { Box, Typography, Paper } from "@mui/material";  
import { MapContainer, TileLayer, Marker, Popup, useMap } from "react-leaflet";  
import api from "../api/axios";  
import SidebarLayout from "../components/layout/SidebarLayout";  
  
// Ajuste la vue automatiquement  
function FitBounds({ projects }) {  
 const map = useMap();  
 useEffect(() => {  
 const points = projects  
 .filter((p) => p.latitude && p.longitude)  
 .map((p) => [parseFloat(p.latitude), parseFloat(p.longitude)]);  
 if (points.length === 1) map.setView(points[0], 13);  
 else if (points.length > 1) map.fitBounds(points, { padding: [30, 30] });  
 else map.setView([46.603354, 1.888334], 6);  
 }, [projects, map]);  
 return null;  
}  
  
// Force Leaflet à recalculer la taille quand le layout change  
function AutoResize() {  
 const map = useMap();  
 useEffect(() => {  
 const tick = () => map.invalidateSize();  
 // petit délai pour laisser l'animation du drawer finir  
 const timeout = setTimeout(tick, 300);  
 window.addEventListener("resize", tick);  
 return () => {  
 clearTimeout(timeout);  
 window.removeEventListener("resize", tick);  
 };  
 }, [map]);  
 return null;  
}  
  
export default function CarteDesProjets() {  
 const [projects, setProjects] = useState([]);  
  
 useEffect(() => {  
 api.get("/projects").then((res) => setProjects(res.data));  
 }, []);  
  
 return (  
 <SidebarLayout>  
 <MapContainer  
 center={[46.603354, 1.888334]}  
 zoom={6}  
 scrollWheelZoom  
 style={{ width: "100%", height: "90vh" }}  
 >  
 <TileLayer  
 attribution="&copy; OpenStreetMap"  
 url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"  
 />  
  
 {projects.map(  
 (p) =>  
 p.latitude &&  
 p.longitude && (  
 <Marker  
 key={p.id}  
 position={[parseFloat(p.latitude), parseFloat(p.longitude)]}  
 >  
 <Popup>  
 <Typography variant="subtitle1" fontWeight={700}>  
 {p.name}  
 </Typography>  
 <Typography variant="body2">  
 <b>Type&nbsp;:</b> {p.type}  
 </Typography>  
 <Typography variant="body2">  
 <b>Adresse&nbsp;:</b> {p.address}  
 </Typography>  
 <Typography variant="body2">  
 <b>Puissance&nbsp;:</b> {p.power} kWc  
 </Typography>  
 <Typography variant="body2">  
 <b>Statut&nbsp;:</b> {p.status}  
 </Typography>  
 <Typography variant="body2">  
 <b>Tél&nbsp;:</b> {p.phone}  
 <br />  
 <b>Email&nbsp;:</b> {p.email}  
 </Typography>  
 </Popup>  
 </Marker>  
 )  
 )}  
  
 <FitBounds projects={projects} />  
 <AutoResize />  
 </MapContainer>  
   
 </SidebarLayout>  
 );  
}

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\pages\Dashboard.jsx

import { useState, useEffect } from "react";  
import {  
 Box,  
 Button,  
 Typography,  
 Grid,  
 Dialog,  
 DialogTitle,  
 DialogContent,  
 DialogActions,  
 TextField,  
 MenuItem,  
 Chip,  
 IconButton,  
 Paper,  
 CircularProgress,  
 InputAdornment,  
} from "@mui/material";  
import AddIcon from "@mui/icons-material/Add";  
import DeleteIcon from "@mui/icons-material/Delete";  
import SearchIcon from "@mui/icons-material/Search";  
import { MapContainer, TileLayer, Marker, useMap } from "react-leaflet";  
import { useNavigate } from "react-router-dom";  
import api from "../api/axios";  
import AutocompleteAdresse from "../components/AutocompleteAdresse";  
import SidebarLayout from "../components/layout/SidebarLayout";  
  
// Widgets  
import ProjectStatusDonut from "../components/dashboard/ProjectStatusDonut";  
import UpcomingMilestones from "../components/dashboard/UpcomingMilestones";  
import OverdueMilestones from "../components/dashboard/OverdueMilestones";  
import RecentActivity from "../components/dashboard/RecentActivity";  
  
function FitBounds({ projects }) {  
 const map = useMap();  
 useEffect(() => {  
 const pts = (projects || [])  
 .filter((p) => p.latitude && p.longitude)  
 .map((p) => [Number(p.latitude), Number(p.longitude)]);  
 if (pts.length === 1) map.setView(pts[0], 13);  
 else if (pts.length > 1) map.fitBounds(pts, { padding: [30, 30] });  
 else map.setView([46.603354, 1.888334], 6);  
 }, [projects, map]);  
 return null;  
}  
  
export default function Dashboard() {  
 const navigate = useNavigate();  
  
 const [projects, setProjects] = useState([]);  
 const [loading, setLoading] = useState(true);  
 const [modalOpen, setModalOpen] = useState(false);  
 const [form, setForm] = useState({  
 name: "",  
 address: "",  
 type: "",  
 power: "",  
 phone: "",  
 email: "",  
 comments: "",  
 owner\_id: "",  
 latitude: null,  
 longitude: null,  
 status: "En cours",  
 });  
 const [statusFilter, setStatusFilter] = useState("");  
 const [search, setSearch] = useState("");  
  
 useEffect(() => {  
 let mounted = true;  
 (async () => {  
 try {  
 const res = await api.get("/projects");  
 const data = Array.isArray(res.data) ? res.data : [];  
 if (mounted) {  
 setProjects(data);  
 setLoading(false);  
 }  
 } catch (err) {  
 if (mounted) {  
 setProjects([]);  
 setLoading(false);  
 }  
 if (err?.response?.status === 401) {  
 navigate("/login");  
 }  
 }  
 })();  
 return () => {  
 mounted = false;  
 };  
 }, [navigate]);  
  
 const counts = {  
 enCours: (projects || []).filter((p) => p.status === "En cours").length,  
 terminee: (projects || []).filter((p) => p.status === "Terminée").length,  
 annulee: (projects || []).filter((p) => p.status === "Annulée").length,  
 };  
  
 const lowerSearch = search.toLowerCase().trim();  
 const filteredProjects = (projects || []).filter((p) => {  
 if (statusFilter && p.status !== statusFilter) return false;  
 if (!lowerSearch) return true;  
 return Object.values(p).join(" ").toLowerCase().includes(lowerSearch);  
 });  
  
 const openModal = () => setModalOpen(true);  
 const closeModal = () => setModalOpen(false);  
 const onChange = (e) =>  
 setForm((f) => ({ ...f, [e.target.name]: e.target.value }));  
  
 async function submit(e) {  
 e.preventDefault();  
 if (!form.name || !form.address || !form.type || !form.power)  
 return alert("Champs obligatoires manquants.");  
 if (!form.latitude || !form.longitude)  
 return alert("Sélectionne une adresse (coordonnées manquantes).");  
 try {  
 await api.post("/projects", form);  
 closeModal();  
 const { data } = await api.get("/projects");  
 setProjects(Array.isArray(data) ? data : []);  
 } catch (err) {  
 alert("Erreur lors de l'ajout du projet.");  
 }  
 }  
  
 return (  
 <SidebarLayout>  
 <Box  
 sx={{  
 width: "100%",  
 minWidth: 0,  
 boxSizing: "border-box",  
 px: { xs: 1.5, md: 4 },  
 pt: 2,  
 pb: 6,  
 }}  
 >  
 {loading ? (  
 <Box sx={{ display: "flex", justifyContent: "center", mt: 8 }}>  
 <CircularProgress />  
 </Box>  
 ) : (  
 <>  
 {/\* ROW 1 — Widgets \*/}  
 {/\* Mobile: horizontal scroll, on utilise directement les widgets (pas de Paper supplémentaire) \*/}  
 <Box  
 sx={{  
 display: { xs: "flex", md: "none" },  
 gap: 1.5,  
 mb: 2,  
 overflowX: "auto",  
 pb: 1,  
 scrollSnapType: "x mandatory",  
 "& > \*": { scrollSnapAlign: "start" },  
 "&::-webkit-scrollbar": { display: "none" },  
 }}  
 >  
 <ProjectStatusDonut counts={counts} />  
 <UpcomingMilestones />  
 <OverdueMilestones />  
 <RecentActivity />  
 </Box>  
  
 {/\* Desktop (inchangé) \*/}  
 <Grid  
 container  
 spacing={2}  
 sx={{ mb: 4, display: { xs: "none", md: "flex" } }}  
 >  
 <Grid size={3} item xs={12} md={3}>  
 <ProjectStatusDonut counts={counts} />  
 </Grid>  
 <Grid size={3} item xs={12} md={3}>  
 <UpcomingMilestones />  
 </Grid>  
 <Grid size={3} item xs={12} md={3}>  
 <OverdueMilestones />  
 </Grid>  
 <Grid size={3} item xs={12} md={3}>  
 <RecentActivity />  
 </Grid>  
 </Grid>  
  
 {/\* ROW 2 — Carte \*/}  
 <Paper sx={{ p: 2, borderRadius: 3, boxShadow: 2, mb: 3 }}>  
 <Typography fontWeight={700} sx={{ mb: 1 }}>  
 Carte des Projets  
 </Typography>  
 <Box  
 sx={{  
 height: { xs: 300, sm: 360, md: 420 },  
 borderRadius: 2,  
 overflow: "hidden",  
 }}  
 >  
 <MapContainer  
 center={[46.603354, 1.888334]}  
 zoom={6}  
 scrollWheelZoom  
 style={{ height: "100%", width: "100%" }}  
 >  
 <TileLayer url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png" />  
 {(filteredProjects || []).map((p) =>  
 p.latitude && p.longitude ? (  
 <Marker  
 key={p.id}  
 position={[Number(p.latitude), Number(p.longitude)]}  
 />  
 ) : null  
 )}  
 <FitBounds projects={filteredProjects} />  
 </MapContainer>  
 </Box>  
 </Paper>  
  
 {/\* ROW 3 — Filtres \*/}  
 <Paper  
 sx={{  
 p: 2,  
 borderRadius: 2,  
 display: "grid",  
 gap: 1.5,  
 alignItems: "center",  
 mb: 3,  
 gridTemplateColumns: {  
 xs: "1fr",  
 sm: "1fr 220px auto",  
 },  
 }}  
 >  
 <TextField  
 size="small"  
 fullWidth  
 placeholder="Rechercher..."  
 value={search}  
 onChange={(e) => setSearch(e.target.value)}  
 InputProps={{  
 startAdornment: (  
 <InputAdornment position="start">  
 <SearchIcon sx={{ color: "text.disabled" }} />  
 </InputAdornment>  
 ),  
 }}  
 />  
 <TextField  
 select  
 size="small"  
 label="Statut"  
 value={statusFilter}  
 onChange={(e) => setStatusFilter(e.target.value)}  
 sx={{ minWidth: { xs: "100%", sm: 180 } }}  
 >  
 {["", "En cours", "Terminée", "Annulée"].map((v) => (  
 <MenuItem key={v || "all"} value={v}>  
 {v || "Tous les statuts"}  
 </MenuItem>  
 ))}  
 </TextField>  
 <Button  
 variant="contained"  
 color="success"  
 startIcon={<AddIcon />}  
 onClick={openModal}  
 sx={{ width: { xs: "100%", sm: "auto" } }}  
 >  
 Ajouter un projet  
 </Button>  
 </Paper>  
  
 {/\* ROW 4 — Cartes projets \*/}  
 <Grid container spacing={2}>  
 {(filteredProjects || []).map((project) => (  
 <Grid item xs={12} sm={6} md={4} lg={3} key={project.id}>  
 <Paper  
 elevation={1}  
 sx={{  
 p: 2,  
 borderRadius: 3,  
 position: "relative",  
 cursor: "pointer",  
 bgcolor: "#fff",  
 minHeight: { xs: 148, md: 165 },  
 "&:hover": { boxShadow: 4, border: "2px solid #06C270" },  
 transition: "all .15s",  
 }}  
 onClick={() => navigate(`/projects/${project.id}`)}  
 >  
 <IconButton  
 size="small"  
 color="error"  
 onClick={async (e) => {  
 e.stopPropagation();  
 if (window.confirm("Supprimer ce projet ?")) {  
 await api.delete(`/projects/${project.id}`);  
 setProjects((prev) =>  
 prev.filter((p) => p.id !== project.id)  
 );  
 }  
 }}  
 sx={{ position: "absolute", top: 8, right: 8 }}  
 >  
 <DeleteIcon />  
 </IconButton>  
  
 <Box  
 sx={{  
 display: "flex",  
 alignItems: "center",  
 gap: 1,  
 mb: 0.5,  
 minWidth: 0,  
 }}  
 >  
 <Typography  
 variant="h6"  
 fontWeight={700}  
 noWrap  
 sx={{ flex: 1 }}  
 >  
 {project.name}  
 </Typography>  
 <Chip  
 size="small"  
 label={project.status}  
 color={  
 project.status === "Terminée"  
 ? "success"  
 : project.status === "Annulée"  
 ? "error"  
 : "info"  
 }  
 sx={{ fontWeight: 700, fontSize: 12 }}  
 />  
 </Box>  
 <Typography  
 variant="body2"  
 color="text.secondary"  
 noWrap  
 sx={{ mb: 0.5 }}  
 >  
 📍 {project.address}  
 </Typography>  
 <Typography variant="body2">  
 <b>Type :</b> {project.type}  
 </Typography>  
 <Typography variant="body2">  
 <b>Puissance :</b> {project.power} kWc  
 </Typography>  
 </Paper>  
 </Grid>  
 ))}  
 </Grid>  
 </>  
 )}  
 </Box>  
  
 <Dialog open={modalOpen} onClose={closeModal} fullWidth maxWidth="sm">  
 <DialogTitle>Nouveau Projet</DialogTitle>  
 <DialogContent>  
 <Box component="form" sx={{ mt: 1 }}>  
 <TextField  
 label="Nom du projet \*"  
 name="name"  
 value={form.name}  
 onChange={onChange}  
 fullWidth  
 sx={{ mb: 2 }}  
 />  
 <AutocompleteAdresse  
 value={form.address}  
 onChange={(val) => setForm((f) => ({ ...f, address: val }))}  
 onCoords={(c) =>  
 setForm((f) => ({  
 ...f,  
 latitude: c.latitude,  
 longitude: c.longitude,  
 }))  
 }  
 />  
 <TextField  
 label="Type \*"  
 name="type"  
 value={form.type}  
 onChange={onChange}  
 fullWidth  
 sx={{ mb: 2 }}  
 />  
 <TextField  
 label="Puissance (kWc) \*"  
 name="power"  
 value={form.power}  
 onChange={onChange}  
 type="number"  
 fullWidth  
 sx={{ mb: 2 }}  
 />  
 <TextField  
 label="Téléphone"  
 name="phone"  
 value={form.phone}  
 onChange={onChange}  
 fullWidth  
 sx={{ mb: 2 }}  
 />  
 <TextField  
 label="Email"  
 name="email"  
 value={form.email}  
 onChange={onChange}  
 fullWidth  
 sx={{ mb: 2 }}  
 />  
 <TextField  
 label="Commentaires"  
 name="comments"  
 value={form.comments}  
 onChange={onChange}  
 fullWidth  
 multiline  
 rows={2}  
 sx={{ mb: 2 }}  
 />  
 <TextField  
 select  
 label="Statut \*"  
 name="status"  
 value={form.status}  
 onChange={onChange}  
 fullWidth  
 sx={{ mb: 2 }}  
 >  
 <MenuItem value="En cours">En cours</MenuItem>  
 <MenuItem value="Terminée">Terminée</MenuItem>  
 <MenuItem value="Annulée">Annulée</MenuItem>  
 </TextField>  
 </Box>  
 </DialogContent>  
 <DialogActions>  
 <Button onClick={closeModal}>Annuler</Button>  
 <Button onClick={submit} variant="contained" color="success">  
 Enregistrer  
 </Button>  
 </DialogActions>  
 </Dialog>  
 </SidebarLayout>  
 );  
}

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\pages\InvestorList.jsx

import { useEffect, useState } from "react";  
import {  
 Box,  
 Button,  
 Typography,  
 Dialog,  
 DialogTitle,  
 DialogContent,  
 DialogActions,  
 TextField,  
 IconButton,  
 Paper,  
 Tooltip,  
} from "@mui/material";  
import { DataGrid } from "@mui/x-data-grid";  
import AddIcon from "@mui/icons-material/Add";  
import EditIcon from "@mui/icons-material/Edit";  
import DeleteIcon from "@mui/icons-material/Delete";  
import SearchIcon from "@mui/icons-material/Search";  
import api from "../api/axios";  
import SidebarLayout from "../components/layout/SidebarLayout";  
  
const defaultForm = {  
 name: "",  
 company: "",  
 email: "",  
 phone: "",  
 notes: "",  
};  
  
export default function InvestorList() {  
 const [investors, setInvestors] = useState([]);  
 const [modalOpen, setModalOpen] = useState(false);  
 const [editId, setEditId] = useState(null);  
 const [form, setForm] = useState(defaultForm);  
 const [search, setSearch] = useState("");  
  
 useEffect(() => {  
 loadInvestors();  
 }, []);  
  
 const loadInvestors = () => {  
 api.get("/investors").then((res) => setInvestors(res.data));  
 };  
  
 const handleOpen = (row) => {  
 if (row) {  
 setForm(row);  
 setEditId(row.id);  
 } else {  
 setForm(defaultForm);  
 setEditId(null);  
 }  
 setModalOpen(true);  
 };  
  
 const handleClose = () => {  
 setModalOpen(false);  
 setForm(defaultForm);  
 setEditId(null);  
 };  
  
 const handleChange = (e) => {  
 setForm((f) => ({ ...f, [e.target.name]: e.target.value }));  
 };  
  
 const handleSave = async () => {  
 try {  
 if (editId) {  
 await api.put(`/investors/${editId}`, form);  
 } else {  
 await api.post("/investors", form);  
 }  
 loadInvestors();  
 handleClose();  
 } catch (err) {  
 alert("Erreur lors de l'enregistrement !");  
 }  
 };  
  
 const handleDelete = async (id) => {  
 if (!window.confirm("Supprimer cet investisseur ?")) return;  
 await api.delete(`/investors/${id}`);  
 setInvestors((list) => list.filter((inv) => inv.id !== id));  
 };  
  
 // Recherche  
 const filteredRows = investors.filter((inv) =>  
 Object.values(inv).join(" ").toLowerCase().includes(search.toLowerCase())  
 );  
  
 const columns = [  
 { field: "id", headerName: "ID", width: 80 },  
 { field: "name", headerName: "Nom", flex: 1, minWidth: 160 },  
 { field: "company", headerName: "Entreprise", flex: 1, minWidth: 160 },  
 { field: "email", headerName: "Email", flex: 1, minWidth: 180 },  
 { field: "phone", headerName: "Téléphone", flex: 1, minWidth: 140 },  
 { field: "notes", headerName: "Notes", flex: 1, minWidth: 160 },  
 {  
 field: "actions",  
 headerName: "",  
 width: 100,  
 sortable: false,  
 filterable: false,  
 renderCell: (params) => (  
 <>  
 <Tooltip title="Modifier">  
 <IconButton  
 size="small"  
 color="primary"  
 onClick={() => handleOpen(params.row)}  
 >  
 <EditIcon fontSize="small" />  
 </IconButton>  
 </Tooltip>  
 <Tooltip title="Supprimer">  
 <IconButton  
 size="small"  
 color="error"  
 onClick={() => handleDelete(params.row.id)}  
 >  
 <DeleteIcon fontSize="small" />  
 </IconButton>  
 </Tooltip>  
 </>  
 ),  
 },  
 ];  
  
 return (  
 <SidebarLayout>  
 <Box  
 sx={{  
 width: "100%",  
 minWidth: 0,  
 boxSizing: "border-box",  
 px: { xs: 2, md: 4 },  
 pt: 2,  
 pb: 6,  
 }}  
 >  
 <Paper sx={{ width: "100%", p: { xs: 2, md: 3 }, borderRadius: 3, boxShadow: 2 }}>  
 <Box  
 sx={{  
 display: "flex",  
 alignItems: "center",  
 justifyContent: "space-between",  
 mb: 2,  
 }}  
 >  
 <Box>  
 <Typography variant="h5" fontWeight={700}>  
 Liste des Investisseurs  
 </Typography>  
 <Typography color="text.secondary" fontSize={15}>  
 Gestion de tous les investisseurs  
 </Typography>  
 </Box>  
 <Button  
 variant="contained"  
 startIcon={<AddIcon />}  
 onClick={() => handleOpen(null)}  
 sx={{ borderRadius: 3, fontWeight: 700 }}  
 >  
 Créer  
 </Button>  
 </Box>  
  
 {/\* Barre de recherche \*/}  
 <Box sx={{ mb: 2, display: "flex", alignItems: "center", gap: 1 }}>  
 <TextField  
 size="small"  
 placeholder="Recherche globale"  
 value={search}  
 onChange={(e) => setSearch(e.target.value)}  
 InputProps={{  
 startAdornment: (  
 <SearchIcon sx={{ color: "text.disabled", mr: 1 }} />  
 ),  
 }}  
 sx={{ maxWidth: 420, bgcolor: "#fff", borderRadius: 2 }}  
 />  
 </Box>  
  
 {/\* Tableau \*/}  
 <DataGrid  
 rows={filteredRows}  
 columns={columns}  
 getRowId={(row) => row.id}  
 autoHeight  
 pageSize={8}  
 rowsPerPageOptions={[8, 16, 32]}  
 disableSelectionOnClick  
 sx={{  
 bgcolor: "#fff",  
 borderRadius: 2,  
 fontSize: 16,  
 boxShadow: 1,  
 "& .MuiDataGrid-row": { cursor: "pointer" },  
 minHeight: 400,  
 maxHeight: 600,  
 }}  
 />  
  
 {/\* Modal ajout/modif \*/}  
 <Dialog open={modalOpen} onClose={handleClose} fullWidth maxWidth="sm">  
 <DialogTitle>  
 {editId ? "Modifier l'investisseur" : "Ajouter un investisseur"}  
 </DialogTitle>  
 <DialogContent  
 sx={{ display: "flex", flexDirection: "column", gap: 2, mt: 1 }}  
 >  
 <TextField  
 label="Nom \*"  
 name="name"  
 value={form.name}  
 onChange={handleChange}  
 fullWidth  
 />  
 <TextField  
 label="Entreprise"  
 name="company"  
 value={form.company}  
 onChange={handleChange}  
 fullWidth  
 />  
 <TextField  
 label="Email"  
 name="email"  
 value={form.email}  
 onChange={handleChange}  
 fullWidth  
 />  
 <TextField  
 label="Téléphone"  
 name="phone"  
 value={form.phone}  
 onChange={handleChange}  
 fullWidth  
 />  
 <TextField  
 label="Notes"  
 name="notes"  
 value={form.notes}  
 onChange={handleChange}  
 fullWidth  
 multiline  
 rows={2}  
 />  
 </DialogContent>  
 <DialogActions>  
 <Button onClick={handleClose}>Annuler</Button>  
 <Button onClick={handleSave} variant="contained" color="success">  
 {editId ? "Enregistrer" : "Créer"}  
 </Button>  
 </DialogActions>  
 </Dialog>  
 </Paper>  
 </Box>  
 </SidebarLayout>  
 );  
}

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\pages\LoginPage.jsx

import { useState, useContext } from 'react';  
import { AuthContext } from '../contexts/AuthContext';  
import api, { setAuthToken } from '../api/axios';  
import { Card, TextField, Button, Typography, Box } from '@mui/material';  
  
export default function LoginPage() {  
 const { login } = useContext(AuthContext);  
 const [email, setEmail] = useState('');  
 const [password, setPassword] = useState('');  
 const [error, setError] = useState('');  
  
 const handleSubmit = async (e) => {  
 e.preventDefault();  
 setError('');  
 try {  
 const res = await api.post('/users/login', { email, password });  
 setAuthToken(res.data.token);  
 login(res.data.token, res.data.user); // passe aussi le user si tu veux  
 window.location.href = '/projects';  
 } catch {  
 setError('Identifiants invalides');  
 }  
 };  
  
 return (  
 <Box sx={{ minHeight: "100vh", display: "flex", alignItems: "center", justifyContent: "center" }}>  
 <Card sx={{ p: 4, minWidth: 350 }}>  
 <Typography variant="h5" sx={{ mb: 2 }}>Connexion</Typography>  
 <form onSubmit={handleSubmit}>  
 <TextField  
 label="Email"  
 value={email}  
 onChange={e => setEmail(e.target.value)}  
 fullWidth  
 sx={{ mb: 2 }}  
 />  
 <TextField  
 label="Mot de passe"  
 type="password"  
 value={password}  
 onChange={e => setPassword(e.target.value)}  
 fullWidth  
 sx={{ mb: 2 }}  
 />  
 {error && <Typography color="error" sx={{ mb: 1 }}>{error}</Typography>}  
 <Button type="submit" variant="contained" fullWidth>  
 Se connecter  
 </Button>  
 </form>  
 </Card>  
 </Box>  
 );  
}

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\pages\ProjectDetail.jsx

import { useState, useEffect, useContext } from "react";  
import { useParams } from "react-router-dom";  
import {  
 Box,  
 Dialog, DialogTitle, DialogContent, DialogActions,  
 Button, TextField, MenuItem, Typography,  
 Card, CardHeader, CardContent,  
 Tabs, Tab, Accordion, AccordionSummary, AccordionDetails,  
 Stack  
} from "@mui/material";  
import ExpandMoreIcon from "@mui/icons-material/ExpandMore";  
import { MapContainer, TileLayer, Marker } from "react-leaflet";  
import api from "../api/axios";  
import { AuthContext } from "../contexts/AuthContext";  
import SidebarLayout from "../components/layout/SidebarLayout";  
  
// cards  
import ProjectHeaderBar from "../components/project/ProjectHeaderBar";  
import CustomFieldsCard from "../components/project/CustomFieldsCard";  
import InvestorsCard from "../components/project/InvestorsCard";  
import StepsCard from "../components/project/StepsCard";  
import AutocompleteAdresse from "../components/AutocompleteAdresse";  
  
const statusList = ["En cours", "Terminée", "Annulée"];  
  
export default function ProjectDetail() {  
 const { id } = useParams();  
 const { user } = useContext(AuthContext);  
  
 const [project, setProject] = useState(null);  
 const [customFields, setCustomFields] = useState([]);  
 const [steps, setSteps] = useState([]);  
 const [loading, setLoading] = useState(true);  
  
 // edit project dialog  
 const [editOpen, setEditOpen] = useState(false);  
 const [editForm, setEditForm] = useState(null);  
 const [statusUpdating, setStatusUpdating] = useState(false);  
  
 // share dialog  
 const [inviteDialogOpen, setInviteDialogOpen] = useState(false);  
 const [inviteEmail, setInviteEmail] = useState("");  
 const [inviteUrl, setInviteUrl] = useState("");  
  
 // tabs: 0 = Étapes, 1 = Investisseurs  
 const [tab, setTab] = useState(0);  
  
 // ===== LOAD =====  
 useEffect(() => {  
 (async () => {  
 setLoading(true);  
 const res = await api.get(`/projects/${id}`);  
 setProject(res.data);  
  
 const cf = await api.get(`/projects/${id}/custom\_fields`);  
 setCustomFields(cf.data);  
  
 const st = await api.get(`/projects/${id}/steps`);  
 setSteps(st.data);  
  
 setLoading(false);  
 })();  
 }, [id]);  
  
 // ===== Custom fields CRUD =====  
 async function refreshCustomFields() {  
 const cf = await api.get(`/projects/${id}/custom\_fields`);  
 setCustomFields(cf.data);  
 }  
 async function addCustomField(field\_name, field\_value, user\_id) {  
 await api.post(`/projects/${id}/custom\_fields`, { field\_name, field\_value, user\_id });  
 await refreshCustomFields();  
 }  
 async function updateCustomField(field\_id, field\_name, field\_value) {  
 await api.put(`/projects/custom\_fields/${field\_id}`, { field\_name, field\_value });  
 await refreshCustomFields();  
 }  
 async function deleteCustomField(field\_id) {  
 await api.delete(`/projects/custom\_fields/${field\_id}`);  
 await refreshCustomFields();  
 }  
  
 // ===== Steps =====  
 async function refreshSteps() {  
 const st = await api.get(`/projects/${id}/steps`);  
 setSteps(st.data);  
 }  
 async function addStep(payload) {  
 await api.post(`/projects/${id}/steps`, payload);  
 await refreshSteps();  
 }  
 async function deleteStep(stepId) {  
 await api.delete(`/projects/${id}/steps/${stepId}`);  
 await refreshSteps();  
 }  
  
 // ===== Project header actions =====  
 function openEdit() {  
 setEditForm({  
 name: project.name || "",  
 address: project.address || "",  
 type: project.type || "",  
 power: project.power || "",  
 phone: project.phone || "",  
 email: project.email || "",  
 comments: project.comments || "",  
 status: project.status || "En cours",  
 latitude: project.latitude || null,  
 longitude: project.longitude || null,  
 });  
 setEditOpen(true);  
 }  
 async function saveEdit(e) {  
 e && e.preventDefault();  
 await api.put(`/projects/${id}`, editForm);  
 setEditOpen(false);  
 const res = await api.get(`/projects/${id}`);  
 setProject(res.data);  
 }  
 async function changeStatus(e) {  
 const newStatus = e.target.value;  
 setStatusUpdating(true);  
 await api.put(`/projects/${id}`, {  
 name: project.name || "",  
 address: project.address || "",  
 type: project.type || "",  
 power: project.power || "",  
 phone: project.phone || "",  
 email: project.email || "",  
 comments: project.comments || "",  
 status: newStatus,  
 latitude: project.latitude,  
 longitude: project.longitude,  
 owner\_id: project.owner\_id || null,  
 });  
 setProject((p) => ({ ...p, status: newStatus }));  
 setStatusUpdating(false);  
 }  
  
 // ===== Public share (dialog) =====  
 async function doInvite(email) {  
 const res = await api.post(`/projects/${id}/invite`, { email });  
 setInviteUrl(res.data.url);  
 }  
 function openShareDialog() {  
 setInviteEmail("");  
 setInviteUrl("");  
 setInviteDialogOpen(true);  
 }  
 function closeShareDialog() {  
 setInviteDialogOpen(false);  
 setInviteEmail("");  
 setInviteUrl("");  
 }  
 async function submitShare(e) {  
 e.preventDefault();  
 if (!inviteEmail) return;  
 await doInvite(inviteEmail);  
 }  
  
 if (loading || !project) {  
 return (  
 <SidebarLayout>  
 <Box sx={{ p: 4 }}>Chargement…</Box>  
 </SidebarLayout>  
 );  
 }  
  
 return (  
 <SidebarLayout>  
 <Box sx={{ display: "flex", gap: 2 }}>  
 {/\* COLONNE GAUCHE (pile verticale) \*/}  
 <Box  
 sx={{  
 width: { xs: "100%", md: "55%" },  
 maxHeight: "80vh",  
 overflowY: "auto",  
 pr: 1,  
 }}  
 >  
 <ProjectHeaderBar  
 project={project}  
 user={user}  
 onEditClick={openEdit}  
 onStatusChange={changeStatus}  
 statusUpdating={statusUpdating}  
 onShareClick={openShareDialog}  
 />  
  
 <Stack spacing={2}>  
 {/\* Infos personnalisées en accordéon compact \*/}  
 <Accordion defaultExpanded disableGutters elevation={0} sx={{ borderRadius: 2, background: "transparent" }}>  
 <AccordionSummary expandIcon={<ExpandMoreIcon />} sx={{ px: 0 }}>  
 <Typography variant="subtitle2">Infos personnalisées</Typography>  
 </AccordionSummary>  
 <AccordionDetails sx={{ p: 0 }}>  
 <CustomFieldsCard  
 fields={customFields}  
 onAdd={(name, value) => addCustomField(name, value, user.id)}  
 onEdit={updateCustomField}  
 onDelete={deleteCustomField}  
 />  
 </AccordionDetails>  
 </Accordion>  
  
 {/\* Bloc Suivi avec onglets \*/}  
 <Card elevation={1} sx={{ borderRadius: 2 }}>  
 <CardHeader  
 sx={{  
 py: 0.5,  
 "& .MuiCardHeader-content": { m: 0 },  
 }}  
 title={  
 <Box sx={{ height: 40, display: "flex", alignItems: "center", overflow: "hidden" }}>  
 <Tabs  
 value={tab}  
 onChange={(\_, v) => setTab(v)}  
 variant="standard"  
 sx={{  
 minHeight: 40,  
 "& .MuiTabs-flexContainer": { gap: 1 },  
 "& .MuiTab-root": {  
 minHeight: 40,  
 px: 1.5,  
 py: 0.5,  
 minWidth: 0,  
 textTransform: "none",  
 fontWeight: 600,  
 lineHeight: 1.2,  
 },  
 "& .MuiTab-root.Mui-selected": {  
 px: 1.5,  
 py: 0.5,  
 fontWeight: 600,  
 lineHeight: 1.2,  
 },  
 }}  
 TabIndicatorProps={{ sx: { height: 2, bottom: 0 } }}  
 >  
 <Tab disableRipple label="Étapes" />  
 <Tab disableRipple label="Investisseurs" />  
 </Tabs>  
 </Box>  
 }  
 />  
 <CardContent sx={{ pt: 2 }}>  
 {tab === 0 && <StepsCard steps={steps} onAdd={addStep} onDelete={deleteStep} />}  
 {tab === 1 && <InvestorsCard projectId={id} />}  
 </CardContent>  
 </Card>  
 </Stack>  
 </Box>  
  
 {/\* COLONNE DROITE : CARTE \*/}  
 <Box  
 sx={{  
 width: { xs: "100%", md: "45%" },  
 display: { xs: "none", md: "block" },  
 height: "80vh",  
 position: "relative",  
 bgcolor: "#f0f4f7",  
 borderRadius: 2,  
 overflow: "hidden",  
 }}  
 >  
 {project.latitude && project.longitude ? (  
 <MapContainer  
 center={[project.latitude, project.longitude]}  
 zoom={15}  
 style={{ width: "100%", height: "100%" }}  
 scrollWheelZoom  
 dragging  
 doubleClickZoom  
 >  
 <TileLayer attribution="&copy; OpenStreetMap" url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png" />  
 <Marker position={[project.latitude, project.longitude]} />  
 </MapContainer>  
 ) : (  
 <Box sx={{ display: "grid", placeItems: "center", height: "100%" }}>  
 <Typography variant="body2" color="text.secondary">  
 Aucune position définie pour ce projet.  
 </Typography>  
 </Box>  
 )}  
 </Box>  
 </Box>  
  
 {/\* ==== DIALOG: Edit projet ==== \*/}  
 <Dialog open={editOpen} onClose={() => setEditOpen(false)} fullWidth maxWidth="sm">  
 <DialogTitle>Modifier le projet</DialogTitle>  
 <DialogContent>  
 {editForm && (  
 <Box component="form" sx={{ mt: 1 }}>  
 <TextField label="Nom" name="name" value={editForm.name} onChange={(e) => setEditForm((f) => ({ ...f, name: e.target.value }))} fullWidth sx={{ mb: 1 }} />  
 <AutocompleteAdresse  
 value={editForm?.address || ""}  
 onChange={(val) => setEditForm((f) => ({ ...f, address: val }))}  
 onCoords={(coords) => setEditForm((f) => ({ ...f, latitude: coords.latitude, longitude: coords.longitude }))}  
 />  
 <TextField label="Type" name="type" value={editForm.type} onChange={(e) => setEditForm((f) => ({ ...f, type: e.target.value }))} fullWidth sx={{ mb: 1 }} />  
 <TextField label="Puissance (kWc)" name="power" value={editForm.power} onChange={(e) => setEditForm((f) => ({ ...f, power: e.target.value }))} type="number" fullWidth sx={{ mb: 1 }} />  
 <TextField label="Téléphone" name="phone" value={editForm.phone} onChange={(e) => setEditForm((f) => ({ ...f, phone: e.target.value }))} fullWidth sx={{ mb: 1 }} />  
 <TextField label="Email" name="email" value={editForm.email} onChange={(e) => setEditForm((f) => ({ ...f, email: e.target.value }))} fullWidth sx={{ mb: 1 }} />  
 <TextField label="Commentaires" name="comments" value={editForm.comments} onChange={(e) => setEditForm((f) => ({ ...f, comments: e.target.value }))} fullWidth multiline rows={2} sx={{ mb: 1 }} />  
 <TextField select label="Statut" name="status" value={editForm.status || "En cours"} onChange={(e) => setEditForm((f) => ({ ...f, status: e.target.value }))} fullWidth sx={{ mb: 1 }}>  
 {statusList.map((s) => (  
 <MenuItem key={s} value={s}>{s}</MenuItem>  
 ))}  
 </TextField>  
 </Box>  
 )}  
 </DialogContent>  
 <DialogActions>  
 <Button onClick={() => setEditOpen(false)}>Annuler</Button>  
 <Button onClick={saveEdit} variant="contained" color="success">Enregistrer</Button>  
 </DialogActions>  
 </Dialog>  
  
 {/\* ==== DIALOG: Partager ==== \*/}  
 <Dialog open={inviteDialogOpen} onClose={closeShareDialog} fullWidth maxWidth="xs">  
 <DialogTitle>Partager le projet</DialogTitle>  
 <DialogContent>  
 {!inviteUrl ? (  
 <Box component="form" onSubmit={submitShare} sx={{ mt: 1, display: "flex", gap: 1 }}>  
 <TextField  
 size="small"  
 label="Email"  
 type="email"  
 fullWidth  
 value={inviteEmail}  
 onChange={(e) => setInviteEmail(e.target.value)}  
 required  
 />  
 <Button type="submit" variant="contained">Inviter</Button>  
 </Box>  
 ) : (  
 <Typography sx={{ mt: 1 }}>  
 Lien généré :{" "}  
 <a href={inviteUrl} target="\_blank" rel="noreferrer">  
 {inviteUrl}  
 </a>  
 </Typography>  
 )}  
 </DialogContent>  
 <DialogActions>  
 <Button onClick={closeShareDialog}>Fermer</Button>  
 </DialogActions>  
 </Dialog>  
 </SidebarLayout>  
 );  
}

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\pages\ProjectList.jsx

import { useEffect, useState } from "react";  
import {  
 Box,  
 Button,  
 Typography,  
 Dialog,  
 DialogTitle,  
 DialogContent,  
 DialogActions,  
 TextField,  
 IconButton,  
 Paper,  
 Tooltip,  
} from "@mui/material";  
import { DataGrid } from "@mui/x-data-grid";  
import AddIcon from "@mui/icons-material/Add";  
import EditIcon from "@mui/icons-material/Edit";  
import DeleteIcon from "@mui/icons-material/Delete";  
import SearchIcon from "@mui/icons-material/Search";  
import api from "../api/axios";  
import SidebarLayout from "../components/layout/SidebarLayout";  
  
const defaultForm = {  
 name: "",  
 address: "",  
 type: "",  
 power: "",  
 phone: "",  
 email: "",  
 comments: "",  
 status: "En cours",  
};  
  
export default function ProjectList() {  
 const [projects, setProjects] = useState([]);  
 const [modalOpen, setModalOpen] = useState(false);  
 const [editId, setEditId] = useState(null);  
 const [form, setForm] = useState(defaultForm);  
 const [search, setSearch] = useState("");  
  
 // Charger les projets  
 useEffect(() => {  
 api.get("/projects").then((res) => setProjects(res.data));  
 }, []);  
  
 // CRUD  
 const handleOpen = (row) => {  
 if (row) {  
 setForm(row);  
 setEditId(row.id);  
 } else {  
 setForm(defaultForm);  
 setEditId(null);  
 }  
 setModalOpen(true);  
 };  
  
 const handleClose = () => {  
 setModalOpen(false);  
 setForm(defaultForm);  
 setEditId(null);  
 };  
  
 const handleChange = (e) => {  
 setForm((f) => ({ ...f, [e.target.name]: e.target.value }));  
 };  
  
 const handleSave = async () => {  
 try {  
 if (editId) {  
 await api.put(`/projects/${editId}`, form);  
 } else {  
 await api.post("/projects", form);  
 }  
 const res = await api.get("/projects");  
 setProjects(res.data);  
 handleClose();  
 } catch (err) {  
 alert("Erreur lors de l'enregistrement !");  
 }  
 };  
  
 const handleDelete = async (id) => {  
 if (!window.confirm("Supprimer ce projet ?")) return;  
 await api.delete(`/projects/${id}`);  
 setProjects((p) => p.filter((proj) => proj.id !== id));  
 };  
  
 // Recherche multi-champs  
 const filteredRows = projects.filter((p) =>  
 Object.values(p).join(" ").toLowerCase().includes(search.toLowerCase())  
 );  
  
 // Colonnes DataGrid  
 const columns = [  
 { field: "id", headerName: "ID", width: 80 },  
 { field: "name", headerName: "Nom", flex: 1, minWidth: 160 },  
 { field: "address", headerName: "Adresse", flex: 2, minWidth: 220 },  
 { field: "type", headerName: "Type", width: 120 },  
 { field: "power", headerName: "Puissance", width: 110 },  
 { field: "status", headerName: "Statut", width: 110 },  
 {  
 field: "actions",  
 headerName: "",  
 width: 100,  
 sortable: false,  
 filterable: false,  
 renderCell: (params) => (  
 <>  
 <Tooltip title="Modifier">  
 <IconButton  
 size="small"  
 color="primary"  
 onClick={() => handleOpen(params.row)}  
 >  
 <EditIcon fontSize="small" />  
 </IconButton>  
 </Tooltip>  
 <Tooltip title="Supprimer">  
 <IconButton  
 size="small"  
 color="error"  
 onClick={() => handleDelete(params.row.id)}  
 >  
 <DeleteIcon fontSize="small" />  
 </IconButton>  
 </Tooltip>  
 </>  
 ),  
 },  
 ];  
  
 return (  
 <SidebarLayout>  
 {/\* Contenu principal — pleine largeur \*/}  
 <Box  
 sx={{  
 width: "100%",  
 minWidth: 0,  
 boxSizing: "border-box",  
 px: { xs: 2, md: 4 },  
 pt: 2,  
 pb: 6,  
 }}  
 >  
 <Paper sx={{ width: "100%", p: { xs: 2, md: 3 }, borderRadius: 3, boxShadow: 2 }}>  
 <Box  
 sx={{  
 display: "flex",  
 alignItems: "center",  
 justifyContent: "space-between",  
 mb: 2,  
 }}  
 >  
 <Box>  
 <Typography variant="h5" fontWeight={700}>  
 Liste des Projets  
 </Typography>  
 <Typography color="text.secondary" fontSize={15}>  
 Gestion de tous les projets  
 </Typography>  
 </Box>  
 <Button  
 variant="contained"  
 startIcon={<AddIcon />}  
 onClick={() => handleOpen(null)}  
 sx={{ borderRadius: 3, fontWeight: 700 }}  
 >  
 Créer  
 </Button>  
 </Box>  
  
 {/\* Barre de recherche \*/}  
 <Box sx={{ mb: 2, display: "flex", alignItems: "center", gap: 1 }}>  
 <TextField  
 size="small"  
 placeholder="Recherche globale"  
 value={search}  
 onChange={(e) => setSearch(e.target.value)}  
 InputProps={{  
 startAdornment: (  
 <SearchIcon sx={{ color: "text.disabled", mr: 1 }} />  
 ),  
 }}  
 sx={{ maxWidth: 420, bgcolor: "#fff", borderRadius: 2 }}  
 />  
 </Box>  
  
 <DataGrid  
 rows={filteredRows}  
 columns={columns}  
 getRowId={(row) => row.id}  
 autoHeight  
 pageSize={8}  
 rowsPerPageOptions={[8, 16, 32]}  
 disableSelectionOnClick  
 sx={{  
 bgcolor: "#fff",  
 borderRadius: 2,  
 fontSize: 16,  
 boxShadow: 1,  
 "& .MuiDataGrid-row": { cursor: "pointer" },  
 minHeight: 400,  
 maxHeight: 600,  
 }}  
 />  
  
 {/\* Modal Ajout/Édition \*/}  
 <Dialog open={modalOpen} onClose={handleClose} fullWidth maxWidth="sm">  
 <DialogTitle>  
 {editId ? "Modifier le projet" : "Ajouter un projet"}  
 </DialogTitle>  
 <DialogContent  
 sx={{ display: "flex", flexDirection: "column", gap: 2, mt: 1 }}  
 >  
 <TextField  
 label="Nom \*"  
 name="name"  
 value={form.name}  
 onChange={handleChange}  
 fullWidth  
 />  
 <TextField  
 label="Adresse \*"  
 name="address"  
 value={form.address}  
 onChange={handleChange}  
 fullWidth  
 />  
 <TextField  
 label="Type \*"  
 name="type"  
 value={form.type}  
 onChange={handleChange}  
 fullWidth  
 />  
 <TextField  
 label="Puissance \*"  
 name="power"  
 value={form.power}  
 onChange={handleChange}  
 fullWidth  
 />  
 <TextField  
 label="Téléphone"  
 name="phone"  
 value={form.phone}  
 onChange={handleChange}  
 fullWidth  
 />  
 <TextField  
 label="Email"  
 name="email"  
 value={form.email}  
 onChange={handleChange}  
 fullWidth  
 />  
 <TextField  
 label="Commentaires"  
 name="comments"  
 value={form.comments}  
 onChange={handleChange}  
 fullWidth  
 multiline  
 rows={2}  
 />  
 <TextField  
 select  
 label="Statut"  
 name="status"  
 value={form.status}  
 onChange={handleChange}  
 fullWidth  
 SelectProps={{ native: true }}  
 >  
 <option value="En cours">En cours</option>  
 <option value="Terminée">Terminée</option>  
 <option value="Annulée">Annulée</option>  
 </TextField>  
 </DialogContent>  
 <DialogActions>  
 <Button onClick={handleClose}>Annuler</Button>  
 <Button onClick={handleSave} variant="contained" color="success">  
 {editId ? "Enregistrer" : "Créer"}  
 </Button>  
 </DialogActions>  
 </Dialog>  
 </Paper>  
 </Box>  
 </SidebarLayout>  
 );  
}

## C:/Users/rescue123/Documents/DEV/CRM/crm-frontend\src\pages\PublicSuiviPage.jsx

// src/pages/PublicSuiviPage.jsx  
import { useEffect, useState } from "react";  
import { useParams, Navigate } from "react-router-dom";  
import api from "../api/axios";  
import {  
 Box,  
 Typography,  
 Drawer,  
 List,  
 ListItem,  
 ListItemText,  
 Chip,  
 Divider,  
 useTheme,  
 useMediaQuery,  
 CircularProgress,  
 Paper,  
} from "@mui/material";  
import { MapContainer, TileLayer, Marker, Popup } from "react-leaflet";  
  
const drawerWidth = 300;  
  
export default function PublicSuiviPage() {  
 const { token } = useParams();  
 const [projectData, setProjectData] = useState(null);  
 const [error, setError] = useState(null);  
 const [loading, setLoading] = useState(true);  
  
 const theme = useTheme();  
 const isMobile = useMediaQuery(theme.breakpoints.down("md"));  
  
 useEffect(() => {  
 api  
 .get(`/public/projects/${token}`)  
 .then((res) => setProjectData(res.data))  
 .catch(() => setError("Lien invalide ou expiré"))  
 .finally(() => setLoading(false));  
 }, [token]);  
  
 if (loading) {  
 return (  
 <Box sx={{ display: "flex", height: "100vh", justifyContent: "center", alignItems: "center" }}>  
 <CircularProgress />  
 </Box>  
 );  
 }  
 if (error) return <Typography color="error" align="center" sx={{ mt: 4 }}>{error}</Typography>;  
 if (!projectData) return <Navigate to="/" replace />;  
  
 const {  
 name,  
 address,  
 latitude,  
 longitude,  
 type,  
 power,  
 phone,  
 email,  
 status,  
 comments,  
 steps = [],  
 custom\_fields = [],  
 } = projectData;  
  
 const InfoPanel = (  
 <Box sx={{ p: 2 }}>  
 <Typography variant="h5" gutterBottom>{name}</Typography>  
 <Divider sx={{ mb: 2 }} />  
  
 <List dense>  
 <ListItem>  
 <ListItemText primary="Adresse" secondary={address} />  
 </ListItem>  
 <ListItem>  
 <ListItemText primary="Type" secondary={type} />  
 </ListItem>  
 <ListItem>  
 <ListItemText primary="Puissance" secondary={`${power} kWc`} />  
 </ListItem>  
 <ListItem>  
 <ListItemText  
 primary="Statut"  
 secondary={  
 <Chip  
 label={status}  
 size="small"  
 sx={{ bgcolor: theme.palette.info.light, color: theme.palette.info.contrastText }}  
 />  
 }  
 />  
 </ListItem>  
  
 <Divider sx={{ my: 1.5 }} />  
  
 <ListItem>  
 <ListItemText primary="Commentaires" secondary={comments || "— Aucun commentaire —"} />  
 </ListItem>  
  
 <Divider sx={{ my: 1.5 }} />  
  
 <ListItem>  
 <ListItemText  
 primary="Contact"  
 secondary={  
 <>  
 <Typography variant="body2">Tél. : {phone || "—"}</Typography>  
 <Typography variant="body2">Email : {email || "—"}</Typography>  
 </>  
 }  
 />  
 </ListItem>  
  
 <Divider sx={{ my: 1.5 }} />  
  
 <Typography variant="subtitle1" sx={{ px: 2, mb: 1 }}>  
 Champs personnalisés  
 </Typography>  
 {custom\_fields.length > 0 ? (  
 custom\_fields.map((f) => (  
 <Chip  
 key={f.id}  
 label={`${f.field\_name} : ${f.field\_value}`}  
 variant="outlined"  
 sx={{ mb: 1, mx: 2, width: "calc(100% - 32px)" }}  
 />  
 ))  
 ) : (  
 <Typography variant="body2" color="text.secondary" sx={{ px: 2 }}>  
 Aucun champ personnalisé.  
 </Typography>  
 )}  
 </List>  
 </Box>  
 );  
  
 return (  
 <Box  
 sx={{  
 display: { xs: "block", md: "flex" },  
 width: "100%",  
 minHeight: "100vh",  
 bgcolor: "#fafafa",  
 }}  
 >  
 {/\* Panneau d’info \*/}  
 {isMobile ? (  
 <Paper elevation={1} sx={{ m: 2, borderRadius: 2 }}>{InfoPanel}</Paper>  
 ) : (  
 <Drawer  
 variant="permanent"  
 anchor="left"  
 sx={{  
 width: drawerWidth,  
 flexShrink: 0,  
 "& .MuiDrawer-paper": {  
 width: drawerWidth,  
 boxSizing: "border-box",  
 p: 0,  
 },  
 }}  
 >  
 {InfoPanel}  
 </Drawer>  
 )}  
  
 {/\* Contenu principal \*/}  
 <Box  
 component="main"  
 sx={{  
 flexGrow: 1,  
 p: { xs: 2, md: 4 },  
 ml: { md: `${drawerWidth}px` }, // espace pris par le drawer en desktop  
 }}  
 >  
 {/\* Carte \*/}  
 {latitude && longitude && (  
 <Paper elevation={2} sx={{ borderRadius: 2, overflow: "hidden", mb: 3 }}>  
 <Typography fontWeight={700} sx={{ p: 2, pb: 0 }}>  
 Carte du projet  
 </Typography>  
 <Box sx={{ height: { xs: 260, sm: 320, md: 400 }, mt: 1 }}>  
 <MapContainer  
 center={[latitude, longitude]}  
 zoom={13}  
 style={{ width: "100%", height: "100%" }}  
 scrollWheelZoom={false}  
 >  
 <TileLayer url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png" />  
 <Marker position={[latitude, longitude]}>  
 <Popup>{name}</Popup>  
 </Marker>  
 </MapContainer>  
 </Box>  
 </Paper>  
 )}  
  
 {/\* Timeline \*/}  
 <Paper elevation={1} sx={{ borderRadius: 2, p: 2 }}>  
 <Typography variant="h6" gutterBottom>  
 Historique des étapes  
 </Typography>  
 {steps.length > 0 ? (  
 <List dense>  
 {steps.map((s) => (  
 <ListItem  
 key={s.id}  
 sx={{ mb: 1, bgcolor: theme.palette.background.paper, borderRadius: 1, boxShadow: 0 }}  
 >  
 <ListItemText  
 primary={  
 <Box sx={{ display: "flex", alignItems: "center", gap: 1 }}>  
 <Typography variant="subtitle2" sx={{ fontWeight: 600 }}>  
 {s.label}  
 </Typography>  
 <Chip  
 label={s.status}  
 size="small"  
 color={  
 (s.status || "").toLowerCase().includes("ok")  
 ? "success"  
 : (s.status || "").toLowerCase().includes("annul")  
 ? "error"  
 : "default"  
 }  
 />  
 </Box>  
 }  
 secondary={  
 <Box>  
 <Typography variant="caption" color="text.secondary">  
 {new Date(s.step\_date).toLocaleDateString("fr-FR")}  
 </Typography>  
 {s.step\_comment && (  
 <Typography variant="body2" sx={{ fontStyle: "italic", mt: 0.5 }}>  
 {s.step\_comment}  
 </Typography>  
 )}  
 </Box>  
 }  
 />  
 </ListItem>  
 ))}  
 </List>  
 ) : (  
 <Typography sx={{ fontStyle: "italic", color: theme.palette.text.disabled }}>  
 Aucune étape enregistrée.  
 </Typography>  
 )}  
 </Paper>  
 </Box>  
 </Box>  
 );  
}