Build/build-config.json : { "appId": "com.bertotracker.app", "productName": "BERTO Tracker", "directories": { "output": "dist" }, "win": { "target": "nsis" }, "mac": { "target": "dmg" }, "linux": { "target": "AppImage" } }

Config/app.json : { "appName": "Berto Tracker", "version": "1.0.0", "environment": "development", "port": 3000 }

Config/database.json : { "development": { "driver": "sqlite3", "database": "./data/employee\_tracker.db" }, "production": { "driver": "sqlite3", "database": "./data/employee\_tracker.db" } }

Config/security.json : { "jwtSecret": "secret\_key", "bcryptRounds": 10 }

Check-db.js : const sqlite3 = require('sqlite3').verbose(); const db = new sqlite3.Database('./data/employee\_tracker.db'); db.all('SELECT \* FROM users', [], (err, rows) => { if (err) { console.error('Erreur lors de la vérification:', err); } else { console.log('Utilisateurs dans la base :', rows); } db.close(); });

Check-users.js : const sqlite3 = require('sqlite3').verbose(); const db = new sqlite3.Database('./data/employee\_tracker.db'); db.all('SELECT \* FROM users', [], (err, rows) => { if (err) { console.error('Erreur:', err); } else { console.log('Utilisateurs:', rows); } db.close(); });

Hash-passwords.js : const bcrypt = require('bcrypt'); // Liste des mots de passe à hacher const passwords = [ 'admin123', // Pour l'admin 'manager123', // Pour manager\_pk12 'manager123' // Pour manager\_zachman ]; // Fonction pour hacher un mot de passe async function hashPasswords() { for (const password of passwords) { const hash = await bcrypt.hash(password, 10); // 10 est le nombre de tours (salt rounds) console.log(`Mot de passe: ${password} -> Hachage: ${hash}`); } } // Exécuter la fonction hashPasswords().catch(err => console.error('Erreur:', err));

Init-db.js : const sqlite3 = require('sqlite3').verbose(); const fs = require('fs'); const path = require('path'); // 📍 Corriger ici : on part du chemin du fichier courant (\_\_dirname), qui est à la racine du projet const projectRoot = \_\_dirname; // 📁 Répertoire data dans le dossier du projet const dataDir = path.join(projectRoot, 'data'); if (!fs.existsSync(dataDir)) { fs.mkdirSync(dataDir, { recursive: true }); } // 📂 Base dans le bon dossier const dbPath = path.join(dataDir, 'employee\_tracker.db'); console.log('📂 Base de données utilisée :', dbPath); // Connexion const db = new sqlite3.Database(dbPath, (err) => { if (err) { console.error('❌ Erreur de connexion à la base de données :', err); process.exit(1); } console.log('✅ Connexion à la base de données établie.'); }); // 📄 Migrations const migrationDir = path.join(projectRoot, 'src', 'database', 'migrations'); const migrations = [ '001\_create\_users.sql', '002\_create\_shops.sql', '003\_create\_employees.sql', '004\_create\_time\_records.sql' ]; migrations.forEach((migration) => { const sqlPath = path.join(migrationDir, migration); const sql = fs.readFileSync(sqlPath, 'utf8'); db.exec(sql, (err) => { if (err) { console.error(`❌ Erreur lors de ${migration} :`, err); process.exit(1); } console.log(`✅ Migration ${migration} exécutée.`); }); }); // 📥 Données initiales const seedPath = path.join(projectRoot, 'src', 'database', 'seeds', 'initial\_data.sql'); const seedSQL = fs.readFileSync(seedPath, 'utf8'); db.exec(seedSQL, (err) => { if (err) { console.error('❌ Erreur lors de l\'insertion des données initiales :', err); process.exit(1); } console.log('✅ Données initiales insérées.'); }); db.close((err) => { if (err) { console.error('❌ Erreur fermeture DB :', err); process.exit(1); } console.log('✅ Base de données prête et fermée.'); });

Main.js : const { app, BrowserWindow, ipcMain } = require('electron'); const path = require('path'); const AuthController = require('./src/backend/controllers/AuthController'); const UserController = require('./src/backend/controllers/UserController'); const EmployeeController = require('./src/backend/controllers/EmployeeController'); const ShopController = require('./src/backend/controllers/ShopController'); const TimeRecordController = require('./src/backend/controllers/TimeRecordController'); function createWindow() { const win = new BrowserWindow({ width: 1200, height: 800, webPreferences: { preload: path.join(\_\_dirname, 'preload.js'), contextIsolation: true, enableRemoteModule: false, nodeIntegration: false } }); win.loadFile('src/frontend/pages/auth/login.html'); } app.whenReady().then(() => { createWindow(); app.on('activate', () => { if (BrowserWindow.getAllWindows().length === 0) { createWindow(); } }); }); app.on('window-all-closed', () => { if (process.platform !== 'darwin') { app.quit(); } }); // IPC handlers for backend communication ipcMain.handle('auth:login', async (event, credentials) => { return await AuthController.login(credentials); }); ipcMain.handle('users:getAll', async (event) => { return await UserController.getAll(); }); ipcMain.handle('users:create', async (event, userData) => { return await UserController.create(userData); }); ipcMain.handle('users:update', async (event, id, userData) => { return await UserController.update(id, userData); }); ipcMain.handle('users:delete', async (event, id) => { return await UserController.delete(id); }); ipcMain.handle('employees:getAll', async (event) => { return await EmployeeController.getAll(); }); ipcMain.handle('employees:create', async (event, employeeData) => { return await EmployeeController.create(employeeData); }); ipcMain.handle('employees:update', async (event, id, employeeData) => { return await EmployeeController.update(id, employeeData); }); ipcMain.handle('employees:delete', async (event, id) => { return await EmployeeController.delete(id); }); ipcMain.handle('employees:generateBadge', async (event, employeeId) => { return await EmployeeController.generateBadge(employeeId); }); ipcMain.handle('shops:getAll', async (event) => { return await ShopController.getAll(); }); ipcMain.handle('shops:create', async (event, shopData) => { return await ShopController.create(shopData); }); ipcMain.handle('shops:update', async (event, id, shopData) => { return await ShopController.update(id, shopData); }); ipcMain.handle('shops:delete', async (event, id) => { return await ShopController.delete(id); }); ipcMain.handle('timeRecords:record', async (event, recordData) => { return await TimeRecordController.recordTime(recordData); }); ipcMain.handle('timeRecords:getReports', async (event, filters) => { return await TimeRecordController.getReports(filters); });

Package.json: { "name": "berto\_tracker", "version": "1.0.0", "main": "main.js", "scripts": { "test": "echo \"Error: no test specified\" && exit 1", "start": "electron .", "build": "electron-builder" }, "keywords": [], "author": "", "license": "ISC", "description": "", "devDependencies": { "electron": "^36.5.0" }, "dependencies": { "bcrypt": "^6.0.0", "dotenv": "^16.5.0", "exceljs": "^4.4.0", "jsonwebtoken": "^9.0.2", "jsqr": "^1.4.0", "pdfkit": "^0.17.1", "qrcode": "^1.5.4", "sqlite3": "^5.1.7", "winston": "^3.17.0" } }

Preload.js : const { contextBridge, ipcRenderer } = require('electron'); contextBridge.exposeInMainWorld('api', { login: (credentials) => ipcRenderer.invoke('auth:login', credentials), getUsers: () => ipcRenderer.invoke('users:getAll'), createUser: (userData) => ipcRenderer.invoke('users:create', userData), updateUser: (id, userData) => ipcRenderer.invoke('users:update', id, userData), deleteUser: (id) => ipcRenderer.invoke('users:delete', id), getEmployees: () => ipcRenderer.invoke('employees:getAll'), createEmployee: (employeeData) => ipcRenderer.invoke('employees:create', employeeData), updateEmployee: (id, employeeData) => ipcRenderer.invoke('employees:update', id, employeeData), deleteEmployee: (id) => ipcRenderer.invoke('employees:delete', id), generateBadge: (employeeId) => ipcRenderer.invoke('employees:generateBadge', employeeId), getShops: () => ipcRenderer.invoke('shops:getAll'), createShop: (shopData) => ipcRenderer.invoke('shops:create', shopData), updateShop: (id, shopData) => ipcRenderer.invoke('shops:update', id, shopData), deleteShop: (id) => ipcRenderer.invoke('shops:delete', id), recordTime: (recordData) => ipcRenderer.invoke('timeRecords:record', recordData), getReports: (filters) => ipcRenderer.invoke('timeRecords:getReports', filters) });

Src/backend/controllers/AuthController.js : const AuthService = require('../services/AuthService'); const logger = require('../utils/logger'); class AuthController { static async login(credentials) { try { const user = await AuthService.authenticate(credentials); return { success: true, user }; } catch (error) { logger.error('Erreur d’authentification:', error); return { success: false, message: error.message }; } } } module.exports = AuthController;

Src/backend/controllers/EmployeeController.js : const Employee = require('../../database/models/Employee'); const BadgeService = require('../services/BadgeService'); const logger = require('../utils/logger'); class EmployeeController { static async getAll() { try { const employees = await Employee.getAll(); return { success: true, data: employees }; } catch (error) { logger.error('Erreur lors de la récupération des employés:', error); return { success: false, message: error.message }; } } static async create(employeeData) { try { const employee = await Employee.create(employeeData); return { success: true, data: employee }; } catch (error) { logger.error('Erreur lors de la création de l’employé:', error); return { success: false, message: error.message }; } } static async update(id, employeeData) { try { const employee = await Employee.update(id, employeeData); return { success: true, data: employee }; } catch (error) { logger.error('Erreur lors de la mise à jour de l’employé:', error); return { success: false, message: error.message }; } } static async delete(id) { try { await Employee.delete(id); return { success: true }; } catch (error) { logger.error('Erreur lors de la suppression de l’employé:', error); return { success: false, message: error.message }; } } static async generateBadge(employeeId) { try { const employee = await Employee.findById(employeeId); if (!employee) throw new Error('Employé non trouvé'); const badgePath = await BadgeService.generateBadge(employee); return { success: true, badgePath }; } catch (error) { logger.error('Erreur lors de la génération du badge:', error); return { success: false, message: error.message }; } } } module.exports = EmployeeController;

Src/backend/controllers/ShopController.js : const Shop = require('../../database/models/Shop'); const logger = require('../utils/logger'); class ShopController { static async getAll() { try { const shops = await Shop.getAll(); return { success: true, data: shops }; } catch (error) { logger.error('Erreur lors de la récupération des boutiques:', error); return { success: false, message: error.message }; } } static async create(shopData) { try { const shop = await Shop.create(shopData); return { success: true, data: shop }; } catch (error) { logger.error('Erreur lors de la création de la boutique:', error); return { success: false, message: error.message }; } } static async update(id, shopData) { try { const shop = await Shop.update(id, shopData); return { success: true, data: shop }; } catch (error) { logger.error('Erreur lors de la mise à jour de la boutique:', error); return { success: false, message: error.message }; } } static async delete(id) { try { await Shop.delete(id); return { success: true }; } catch (error) { logger.error('Erreur lors de la suppression de la boutique:', error); return { success: false, message: error.message }; } } } module.exports = ShopController;

Src/backend/controllers/TimeRecordController.js : const TimeRecord = require('../../database/models/TimeRecord'); const ReportService = require('../services/ReportService'); const logger = require('../utils/logger'); class TimeRecordController { static async recordTime(recordData) { try { const record = await TimeRecord.create(recordData); return { success: true, data: record }; } catch (error) { logger.error('Erreur lors de l’enregistrement du temps:', error); return { success: false, message: error.message }; } } static async getReports(filters) { try { const records = await TimeRecord.getReports(filters); const reportPath = await ReportService.generateExcelReport(records, filters); return { success: true, data: records, reportPath }; } catch (error) { logger.error('Erreur lors de la génération du rapport:', error); return { success: false, message: error.message }; } } } module.exports = TimeRecordController;

Src/backend/controllers/UserController.js : const User = require('../../database/models/User'); const logger = require('../utils/logger'); class UserController { static async getAll() { try { const users = await User.getAll(); return { success: true, data: users }; } catch (error) { logger.error('Erreur lors de la récupération des utilisateurs:', error); return { success: false, message: error.message }; } } static async create(userData) { try { const user = await User.create(userData); return { success: true, data: user }; } catch (error) { logger.error('Erreur lors de la création de l’utilisateur:', error); return { success: false, message: error.message }; } } static async update(id, userData) { try { const user = await User.update(id, userData); return { success: true, data: user }; } catch (error) { logger.error('Erreur lors de la mise à jour de l’utilisateur:', error); return { success: false, message: error.message }; } } static async delete(id) { try { await User.delete(id); return { success: true }; } catch (error) { logger.error('Erreur lors de la suppression de l’utilisateur:', error); return { success: false, message: error.message }; } } } module.exports = UserController;

Src/backend/middleware/auth.js : const jwt = require('jsonwebtoken'); const logger = require('../utils/logger'); module.exports = (req, res, next) => { const token = req.headers['authorization']?.split(' ')[1]; if (!token) { logger.warn('Tentative d’accès sans token'); return res.status(401).json({ message: 'Token requis' }); } try { const decoded = jwt.verify(token, 'secret\_key'); req.user = decoded; next(); } catch (error) { logger.error('Erreur d’authentification:', error); res.status(401).json({ message: 'Token invalide' }); } };

Src/backend/middleware/validation.js : const logger = require('../utils/logger'); module.exports = (schema) => (req, res, next) => { const { error } = schema.validate(req.body); if (error) { logger.warn('Erreur de validation:', error.details); return res.status(400).json({ message: error.details[0].message }); } next(); };

Src/backend/services/AuthService.js : const bcrypt = require('bcrypt'); const jwt = require('jsonwebtoken'); const User = require('../../database/models/User'); const logger = require('../utils/logger'); class AuthService { static async authenticate({ username, password }) { const user = await User.findByUsername(username); if (!user) { throw new Error('Utilisateur non trouvé'); } const isMatch = await bcrypt.compare(password, user.password); if (!isMatch) { throw new Error('Mot de passe incorrect'); } const token = jwt.sign({ id: user.id, role: user.role, shop\_id: user.shop\_id }, 'secret\_key', { expiresIn: '1h' }); return { user, token }; } } module.exports = AuthService;

Src/backend/services/BadgeService.js : const PDFDocument = require('pdfkit'); const fs = require('fs'); const path = require('path'); const QRCode = require('qrcode'); const logger = require('../utils/logger'); class BadgeService { static async generateBadge(employee) { return new Promise((resolve, reject) => { const badgePath = path.join(\_\_dirname, '../../../output/badges', `badge\_${employee.id}.pdf`); const doc = new PDFDocument({ size: [350, 500] }); const stream = fs.createWriteStream(badgePath); doc.pipe(stream); // Header doc.rect(0, 0, 350, 100).fill('#FF6600'); doc.font('Helvetica-Bold').fontSize(28).fillColor('white').text('SD BERTO', 20, 30, { align: 'center' }); doc.font('Helvetica').fontSize(14).text('Badge Employé', 20, 60, { align: 'center' }); // Employee Info doc.font('Helvetica-Bold').fontSize(24).fillColor('#333').text(employee.name, 20, 150, { align: 'center' }); doc.font('Helvetica').fontSize(16).text(`Poste: ${employee.department}`, 20, 200, { align: 'center' }); doc.fontSize(16).text(`Boutique: ${employee.shop\_id === 1 ? 'Boutique PK 12' : 'Boutique Zachman'}`, 20, 230, { align: 'center' }); // QR Code const qrPath = path.join(\_\_dirname, '../../../output/temp', `qr\_${employee.id}.png`); QRCode.toFile(qrPath, `EMP-${employee.id}`, { width: 120 }, (err) => { if (err) { logger.error('Erreur lors de la génération du QR code:', err); reject(err); return; } doc.image(qrPath, 115, 300, { width: 120 }); doc.font('Helvetica').fontSize(11).fillColor('#888').text(`ID: EMP-${employee.id} | Validé le ${new Date().toLocaleDateString('fr-FR')}`, 20, 470, { align: 'center' }); doc.end(); stream.on('finish', () => { fs.unlinkSync(qrPath); // Supprime le fichier QR temporaire resolve(badgePath); }); stream.on('error', (err) => { logger.error('Erreur lors de l’écriture du badge PDF:', err); reject(err); }); }); }); } } module.exports = BadgeService;

Src/backend/services/QRCodeService.js : const QRCode = require('qrcode'); const path = require('path'); const logger = require('../utils/logger'); class QRCodeService { static async generateQRCode(data) { try { const qrPath = path.join(\_\_dirname, '../../../output/temp', `qr\_${Date.now()}.png`); await QRCode.toFile(qrPath, data, { width: 120 }); return qrPath; } catch (error) { logger.error('Erreur lors de la génération du QR code:', error); throw error; } } } module.exports = QRCodeService;

Src/backend/services/ReportService.js : const ExcelJS = require('exceljs'); const path = require('path'); const logger = require('../utils/logger'); class ReportService { static async generateExcelReport(records, filters) { const workbook = new ExcelJS.Workbook(); const worksheet = workbook.addWorksheet('Rapport'); // En-têtes worksheet.columns = [ { header: 'Nom', key: 'name', width: 20 }, { header: 'Boutique', key: 'shop\_name', width: 20 }, { header: 'Département', key: 'department', width: 20 }, { header: 'Date', key: 'date', width: 15 }, { header: 'Arrivée', key: 'arrival\_time', width: 15 }, { header: 'Départ', key: 'departure\_time', width: 15 }, { header: 'Pauses (min)', key: 'breaks', width: 15 }, { header: 'Statut', key: 'status', width: 15 } ]; // Données records.forEach(record => { worksheet.addRow(record); }); // Style worksheet.getRow(1).font = { bold: true }; worksheet.getRow(1).fill = { type: 'pattern', pattern: 'solid', fgColor: { argb: 'FFFF6600' } }; const reportPath = path.join(\_\_dirname, '../../../output/reports', `rapport\_${Date.now()}.xlsx`); await workbook.xlsx.writeFile(reportPath); return reportPath; } } module.exports = ReportService;

Src/backend/services/TimeTrackingService.js : const TimeRecord = require('../../database/models/TimeRecord'); const logger = require('../utils/logger'); class TimeTrackingService { static async recordTime(employeeId, type) { try { const date = new Date().toISOString().split('T')[0]; const time = new Date().toTimeString().split(' ')[0]; let record = await TimeRecord.create({ employee\_id: employeeId, date, [type === 'arrival' ? 'arrival\_time' : 'departure\_time']: time, breaks: 0, status: type === 'arrival' ? 'present' : 'present' }); return record; } catch (error) { logger.error('Erreur lors de l’enregistrement du temps:', error); throw error; } } } module.exports = TimeTrackingService;

Src/backend/utils/constants.js : module.exports = { DEPARTMENTS: ['Supervision Réseau', 'Commercial', 'Comptabilité', 'SAV', 'Caissière'], SCHEDULES: ['07h30-15h', '07h30-18h', '15h-21h', '07h30-16h'], ROLES: ['admin', 'manager'], ZONES: ['Douala'] };

Src/backend/utils/database.js : const db = require('../../database/database'); module.exports = { runMigration: (sql) => { return new Promise((resolve, reject) => { db.exec(sql, (err) => { if (err) reject(err); else resolve(); }); }); } };

Src/backend/utils/encryption.js : const bcrypt = require('bcrypt'); const logger = require('./logger'); module.exports = { hashPassword: async (password) => { try { return await bcrypt.hash(password, 10); } catch (error) { logger.error('Erreur lors du hachage du mot de passe:', error); throw error; } }, comparePassword: async (password, hash) => { try { return await bcrypt.compare(password, hash); } catch (error) { logger.error('Erreur lors de la comparaison du mot de passe:', error); throw error; } } };

Src/backend/utils/logger.js : const winston = require('winston'); const path = require('path'); const logger = winston.createLogger({ level: 'info', format: winston.format.combine( winston.format.timestamp(), winston.format.json() ), transports: [ new winston.transports.File({ filename: path.join(\_\_dirname, '../../../logs/error.log'), level: 'error' }), new winston.transports.File({ filename: path.join(\_\_dirname, '../../../logs/app.log') }) ] }); if (process.env.NODE\_ENV !== 'production') { logger.add(new winston.transports.Console({ format: winston.format.simple() })); } module.exports = logger;

Src/database/database.js : const sqlite3 = require('sqlite3').verbose(); const path = require('path'); const fs = require('fs'); const logger = require('../backend/utils/logger'); // 📌 Chemin racine du projet const projectRoot = path.resolve(\_\_dirname, '../../..'); // 📁 Dossier data à la racine du projet const dataDir = path.join(\_\_dirname, '..', '..', '..', 'berto\_tracker', 'data'); if (!fs.existsSync(dataDir)) { fs.mkdirSync(dataDir, { recursive: true }); } // 📌 Chemin absolu vers la base const dbPath = path.join(dataDir, 'employee\_tracker.db'); console.log('📂 Base SQLite utilisée :', dbPath); const db = new sqlite3.Database(dbPath, (err) => { if (err) { logger.error('Erreur de connexion à la base de données:', err); } else { logger.info('Connexion à la base de données SQLite établie'); } }); module.exports = db;

Src/database /migrations/001\_create\_users.sql : CREATE TABLE IF NOT EXISTS users ( id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT NOT NULL, username TEXT NOT NULL UNIQUE, password TEXT NOT NULL, role TEXT NOT NULL CHECK(role IN ('admin', 'manager')), shop\_id INTEGER, FOREIGN KEY (shop\_id) REFERENCES shops(id) );

Src/database /migrations/002\_create\_shops.sql : CREATE TABLE IF NOT EXISTS shops ( id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT NOT NULL, zone TEXT NOT NULL );

Src/database /migrations/003\_create\_employees.sql : CREATE TABLE IF NOT EXISTS employees ( id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT NOT NULL, shop\_id INTEGER NOT NULL, department TEXT NOT NULL CHECK(department IN ('Supervision Réseau', 'Commercial', 'Comptabilité', 'SAV', 'Caissière')), schedule TEXT NOT NULL, hire\_date TEXT NOT NULL, FOREIGN KEY (shop\_id) REFERENCES shops(id) );

Src/migrations/004\_create\_time\_records.sql : CREATE TABLE IF NOT EXISTS time\_records ( id INTEGER PRIMARY KEY AUTOINCREMENT, employee\_id INTEGER NOT NULL, date TEXT NOT NULL, arrival\_time TEXT, departure\_time TEXT, breaks INTEGER, status TEXT CHECK(status IN ('present', 'absent', 'late')), FOREIGN KEY (employee\_id) REFERENCES employees(id) );

Src/database/models/Employee.js : const db = require('../database'); const logger = require('../../backend/utils/logger'); class Employee { static createTable() { return new Promise((resolve, reject) => { db.run(` CREATE TABLE IF NOT EXISTS employees ( id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT NOT NULL, shop\_id INTEGER NOT NULL, department TEXT NOT NULL CHECK(department IN ('Supervision Réseau', 'Commercial', 'Comptabilité', 'SAV', 'Caissière')), schedule TEXT NOT NULL, hire\_date TEXT NOT NULL, FOREIGN KEY (shop\_id) REFERENCES shops(id) ) `, (err) => { if (err) { logger.error('Erreur lors de la création de la table employees:', err); reject(err); } else { logger.info('Table employees créée ou déjà existante'); resolve(); } }); }); } static create(employeeData) { return new Promise((resolve, reject) => { db.run( 'INSERT INTO employees (name, shop\_id, department, schedule, hire\_date) VALUES (?, ?, ?, ?, ?)', [employeeData.name, employeeData.shop\_id, employeeData.department, employeeData.schedule, employeeData.hire\_date], function (err) { if (err) { logger.error('Erreur lors de la création de l’employé:', err); reject(err); } else { resolve({ id: this.lastID, ...employeeData }); } } ); }); } static getAll() { return new Promise((resolve, reject) => { db.all('SELECT \* FROM employees', [], (err, rows) => { if (err) { logger.error('Erreur lors de la récupération des employés:', err); reject(err); } else { resolve(rows); } }); }); } static update(id, employeeData) { return new Promise((resolve, reject) => { db.run( 'UPDATE employees SET name = ?, shop\_id = ?, department = ?, schedule = ?, hire\_date = ? WHERE id = ?', [employeeData.name, employeeData.shop\_id, employeeData.department, employeeData.schedule, employeeData.hire\_date, id], (err) => { if (err) { logger.error('Erreur lors de la mise à jour de l’employé:', err); reject(err); } else { resolve({ id, ...employeeData }); } } ); }); } static delete(id) { return new Promise((resolve, reject) => { db.run('DELETE FROM employees WHERE id = ?', [id], (err) => { if (err) { logger.error('Erreur lors de la suppression de l’employé:', err); reject(err); } else { resolve(); } }); }); } static findById(id) { return new Promise((resolve, reject) => { db.get('SELECT \* FROM employees WHERE id = ?', [id], (err, row) => { if (err) { logger.error('Erreur lors de la recherche de l’employé:', err); reject(err); } else { resolve(row); } }); }); } } module.exports = Employee;

Src/database/models/Shop.js : const db = require('../database'); const logger = require('../../backend/utils/logger'); class Shop { static createTable() { return new Promise((resolve, reject) => { db.run(` CREATE TABLE IF NOT EXISTS shops ( id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT NOT NULL, zone TEXT NOT NULL ) `, (err) => { if (err) { logger.error('Erreur lors de la création de la table shops:', err); reject(err); } else { logger.info('Table shops créée ou déjà existante'); resolve(); } }); }); } static create(shopData) { return new Promise((resolve, reject) => { db.run( 'INSERT INTO shops (name, zone) VALUES (?, ?)', [shopData.name, shopData.zone], function (err) { if (err) { logger.error('Erreur lors de la création de la boutique:', err); reject(err); } else { resolve({ id: this.lastID, ...shopData }); } } ); }); } static getAll() { return new Promise((resolve, reject) => { db.all('SELECT \* FROM shops', [], (err, rows) => { if (err) { logger.error('Erreur lors de la récupération des boutiques:', err); reject(err); } else { resolve(rows); } }); }); } static update(id, shopData) { return new Promise((resolve, reject) => { db.run( 'UPDATE shops SET name = ?, zone = ? WHERE id = ?', [shopData.name, shopData.zone, id], (err) => { if (err) { logger.error('Erreur lors de la mise à jour de la boutique:', err); reject(err); } else { resolve({ id, ...shopData }); } } ); }); } static delete(id) { return new Promise((resolve, reject) => { db.run('DELETE FROM shops WHERE id = ?', [id], (err) => { if (err) { logger.error('Erreur lors de la suppression de la boutique:', err); reject(err); } else { resolve(); } }); }); } } module.exports = Shop;

Src/database/models/TimeRecord.js : const db = require('../database'); const logger = require('../../backend/utils/logger'); class TimeRecord { static createTable() { return new Promise((resolve, reject) => { db.run(` CREATE TABLE IF NOT EXISTS time\_records ( id INTEGER PRIMARY KEY AUTOINCREMENT, employee\_id INTEGER NOT NULL, date TEXT NOT NULL, arrival\_time TEXT, departure\_time TEXT, breaks INTEGER, status TEXT CHECK(status IN ('present', 'absent', 'late')), FOREIGN KEY (employee\_id) REFERENCES employees(id) ) `, (err) => { if (err) { logger.error('Erreur lors de la création de la table time\_records:', err); reject(err); } else { logger.info('Table time\_records créée ou déjà existante'); resolve(); } }); }); } static create(recordData) { return new Promise((resolve, reject) => { db.run( 'INSERT INTO time\_records (employee\_id, date, arrival\_time, departure\_time, breaks, status) VALUES (?, ?, ?, ?, ?, ?)', [recordData.employee\_id, recordData.date, recordData.arrival\_time, recordData.departure\_time, recordData.breaks, recordData.status], function (err) { if (err) { logger.error('Erreur lors de la création de l’enregistrement:', err); reject(err); } else { resolve({ id: this.lastID, ...recordData }); } } ); }); } static getReports(filters) { return new Promise((resolve, reject) => { let query = 'SELECT tr.\*, e.name, e.department, s.name as shop\_name FROM time\_records tr JOIN employees e ON tr.employee\_id = e.id JOIN shops s ON e.shop\_id = s.id WHERE 1=1'; let params = []; if (filters.dateFrom) { query += ' AND tr.date >= ?'; params.push(filters.dateFrom); } if (filters.dateTo) { query += ' AND tr.date <= ?'; params.push(filters.dateTo); } if (filters.shopId) { query += ' AND e.shop\_id = ?'; params.push(filters.shopId); } if (filters.department) { query += ' AND e.department = ?'; params.push(filters.department); } if (filters.employeeId) { query += ' AND tr.employee\_id = ?'; params.push(filters.employeeId); } if (filters.status) { query += ' AND tr.status = ?'; params.push(filters.status); } db.all(query, params, (err, rows) => { if (err) { logger.error('Erreur lors de la récupération des rapports:', err); reject(err); } else { resolve(rows); } }); }); } } module.exports = TimeRecord;

Src/database/models/User.js : const db = require('../database'); const logger = require('../../backend/utils/logger'); class User { static createTable() { return new Promise((resolve, reject) => { db.run(` CREATE TABLE IF NOT EXISTS users ( id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT NOT NULL, username TEXT NOT NULL UNIQUE, password TEXT NOT NULL, role TEXT NOT NULL CHECK(role IN ('admin', 'manager')), shop\_id INTEGER, FOREIGN KEY (shop\_id) REFERENCES shops(id) ) `, (err) => { if (err) { logger.error('Erreur lors de la création de la table users:', err); reject(err); } else { logger.info('Table users créée ou déjà existante'); resolve(); } }); }); } static create(userData) { return new Promise((resolve, reject) => { db.run( 'INSERT INTO users (name, username, password, role, shop\_id) VALUES (?, ?, ?, ?, ?)', [userData.name, userData.username, userData.password, userData.role, userData.shop\_id], function (err) { if (err) { logger.error('Erreur lors de la création de l’utilisateur:', err); reject(err); } else { resolve({ id: this.lastID, ...userData }); } } ); }); } static findByUsername(username) { return new Promise((resolve, reject) => { db.get('SELECT \* FROM users WHERE username = ?', [username], (err, row) => { if (err) { logger.error('Erreur lors de la recherche de l’utilisateur:', err); reject(err); } else { resolve(row); } }); }); } static getAll() { return new Promise((resolve, reject) => { db.all('SELECT \* FROM users', [], (err, rows) => { if (err) { logger.error('Erreur lors de la récupération des utilisateurs:', err); reject(err); } else { resolve(rows); } }); }); } static update(id, userData) { return new Promise((resolve, reject) => { db.run( 'UPDATE users SET name = ?, username = ?, password = ?, role = ?, shop\_id = ? WHERE id = ?', [userData.name, userData.username, userData.password, userData.role, userData.shop\_id, id], (err) => { if (err) { logger.error('Erreur lors de la mise à jour de l’utilisateur:', err); reject(err); } else { resolve({ id, ...userData }); } } ); }); } static delete(id) { return new Promise((resolve, reject) => { db.run('DELETE FROM users WHERE id = ?', [id], (err) => { if (err) { logger.error('Erreur lors de la suppression de l’utilisateur:', err); reject(err); } else { resolve(); } }); }); } } module.exports = User;

Src/database/seeds/initial\_data.sql : INSERT INTO shops (name, zone) VALUES ('Boutique PK 12', 'Douala'); INSERT INTO shops (name, zone) VALUES ('Boutique Zachman', 'Douala'); INSERT INTO users (name, username, password, role, shop\_id) VALUES ('Admin', 'admin', '$2b$10$6m/8hG6fW./zWi6UDvgGGuRHL/hT0o/rhtEbFp875kvT1sB4NdAzC', 'admin', NULL); INSERT INTO users (name, username, password, role, shop\_id) VALUES ('Manager PK 12', 'manager\_pk12', '$2b$10$IlTn3KKrCMxVRDKruK5N4uE9Un8FFmEyVP6/h9v8ZRYmuk9RjZ3FC', 'manager', 1); INSERT INTO users (name, username, password, role, shop\_id) VALUES ('Manager Zachman', 'manager\_zachman', '$2b$10$p5UFXc0KRQwUl7336zZiYeXA.iTON3rgJq5bArKSjryKIyoMgXy7u', 'manager', 2); INSERT INTO employees (name, shop\_id, department, schedule, hire\_date) VALUES ('Jean Dupont', 1, 'Commercial', '07h30-15h', '2025-01-01'); INSERT INTO employees (name, shop\_id, department, schedule, hire\_date) VALUES ('Marie Dubois', 2, 'Caissière', '15h-21h', '2025-02-01'); INSERT INTO time\_records (employee\_id, date, arrival\_time, departure\_time, breaks, status) VALUES (1, '2025-06-23', '07:30', '15:00', 30, 'present'); INSERT INTO time\_records (employee\_id, date, arrival\_time, departure\_time, breaks, status) VALUES (2, '2025-06-23', '15:15', '21:00', 15, 'late');

Src/frontend/assets/templates/badge-template.html : <!DOCTYPE html> <html lang="fr"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Badge Employé - SD BERTO</title> <link href="https://fonts.googleapis.com/css2?family=Montserrat:wght@400;600;700&family=Roboto:wght@300;400;500&display=swap" rel="stylesheet"> <style> \* { margin: 0; padding: 0; box-sizing: border-box; } body { font-family: 'Roboto', sans-serif; background: #f5f5f5; padding: 20px; display: flex; justify-content: center; align-items: center; min-height: 100vh; } .badge-container { background: white; width: 350px; height: 500px; border-radius: 20px; box-shadow: 0 10px 30px rgba(255, 102, 0, 0.2); overflow: hidden; position: relative; border: 3px solid #FF6600; } .badge-header { background: linear-gradient(135deg, #FF6600 0%, #FF8533 100%); padding: 25px 20px; text-align: center; color: white; position: relative; } .badge-header::before { content: ''; position: absolute; top: 0; left: 0; right: 0; bottom: 0; background: url('data:image/svg+xml,<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 100 100"><circle cx="10" cy="10" r="1" fill="rgba(255,255,255,0.1)"/><circle cx="90" cy="20" r="1.5" fill="rgba(255,255,255,0.1)"/><circle cx="30" cy="80" r="1" fill="rgba(255,255,255,0.1)"/><circle cx="70" cy="70" r="1.2" fill="rgba(255,255,255,0.1)"/></svg>'); pointer-events: none; } .company-logo { font-family: 'Montserrat', sans-serif; font-size: 28px; font-weight: 700; letter-spacing: 2px; margin-bottom: 8px; position: relative; z-index: 1; } .badge-subtitle { font-family: 'Roboto', sans-serif; font-size: 14px; font-weight: 300; opacity: 0.9; position: relative; z-index: 1; } .badge-body { padding: 30px 25px; text-align: center; flex: 1; display: flex; flex-direction: column; justify-content: space-between; } .employee-info { margin-bottom: 25px; } .employee-name { font-family: 'Montserrat', sans-serif; font-size: 24px; font-weight: 600; color: #333; margin-bottom: 15px; line-height: 1.2; } .employee-details { margin-bottom: 20px; } .detail-item { margin-bottom: 12px; display: flex; align-items: center; justify-content: center; gap: 10px; } .detail-label { font-family: 'Montserrat', sans-serif; font-size: 12px; font-weight: 600; color: #FF6600; text-transform: uppercase; letter-spacing: 1px; min-width: 60px; } .detail-value { font-family: 'Roboto', sans-serif; font-size: 16px; font-weight: 400; color: #555; } .qr-section { margin-top: 20px; } .qr-title { font-family: 'Montserrat', sans-serif; font-size: 12px; font-weight: 600; color: #FF6600; text-transform: uppercase; letter-spacing: 1px; margin-bottom: 15px; } .qr-code { width: 120px; height: 120px; border: 3px solid #FF6600; border-radius: 10px; margin: 0 auto; display: flex; align-items: center; justify-content: center; background: #f9f9f9; } .qr-placeholder { width: 100px; height: 100px; background: linear-gradient(45deg, #FF6600, #FF8533); border-radius: 5px; display: flex; align-items: center; justify-content: center; color: white; font-family: 'Montserrat', sans-serif; font-size: 10px; text-align: center; line-height: 1.2; } .badge-footer { background: #f8f8f8; padding: 15px; text-align: center; border-top: 2px solid #FF6600; } .badge-id { font-family: 'Roboto', sans-serif; font-size: 11px; color: #888; font-weight: 300; } /\* Animation au hover \*/ .badge-container:hover { transform: translateY(-5px); box-shadow: 0 15px 40px rgba(255, 102, 0, 0.3); transition: all 0.3s ease; } /\* Version imprimable \*/ @media print { body { background: white; padding: 0; } .badge-container { box-shadow: none; border: 2px solid #FF6600; } } /\* Responsive \*/ @media (max-width: 480px) { .badge-container { width: 300px; height: 450px; } .company-logo { font-size: 24px; } .employee-name { font-size: 20px; } } </style> </head> <body> <div class="badge-container"> <div class="badge-header"> <div class="company-logo">SD BERTO</div> <div class="badge-subtitle">Badge Employé</div> </div> <div class="badge-body"> <div class="employee-info"> <div class="employee-name">DUPONT Jean-Pierre</div> <div class="employee-details"> <div class="detail-item"> <span class="detail-label">Poste:</span> <span class="detail-value">Vendeur Senior</span> </div> <div class="detail-item"> <span class="detail-label">Boutique:</span> <span class="detail-value">Douala Centre</span> </div> </div> </div> <div class="qr-section"> <div class="qr-title">Code QR</div> <div class="qr-code"> <div class="qr-placeholder"> QR CODE<br> GÉNÉRÉ<br> AUTOMATIQUEMENT </div> </div> </div> </div> <div class="badge-footer"> <div class="badge-id">ID: EMP-2024-001 | Validé le 22/06/2025</div> </div> </div> <script> // Fonction pour générer un vrai QR code (à intégrer avec une librairie QR) function generateQRCode(employeeId, element) { // Ici vous pouvez intégrer QRCode.js ou une autre librairie // Exemple d'utilisation: // new QRCode(element, { // text: employeeId, // width: 100, // height: 100, // colorDark: "#FF6600", // colorLight: "#ffffff" // }); } // Fonction pour personnaliser le badge function customizeBadge(employeeData) { document.querySelector('.employee-name').textContent = employeeData.name; document.querySelector('.detail-value').textContent = employeeData.position; document.querySelectorAll('.detail-value')[1].textContent = employeeData.shop; document.querySelector('.badge-id').textContent = `ID: ${employeeData.id} | Validé le ${new Date().toLocaleDateString('fr-FR')}`; } // Exemple d'utilisation // customizeBadge({ // name: "MARTIN Sophie", // position: "Caissière", // shop: "Bonabéri", // id: "EMP-2024-002" // }); </script> </body> </html>

Src/frontend/js/api/auth.js : const api = window.api; export async function login(credentials) { try { const response = await api.login(credentials); if (response.success) { localStorage.setItem('token', response.user.token); localStorage.setItem('user', JSON.stringify(response.user.user)); } return response; } catch (error) { console.error('Erreur lors de la connexion:', error); throw error; } }

Src/frontend/js/api/employees.js : const api = window.api; export async function getEmployees() { try { return await api.getEmployees(); } catch (error) { console.error('Erreur lors de la récupération des employés:', error); throw error; } } export async function createEmployee(employeeData) { try { return await api.createEmployee(employeeData); } catch (error) { console.error('Erreur lors de la création de l’employé:', error); throw error; } } export async function updateEmployee(id, employeeData) { try { return await api.updateEmployee(id, employeeData); } catch (error) { console.error('Erreur lors de la mise à jour de l’employé:', error); throw error; } } export async function deleteEmployee(id) { try { return await api.deleteEmployee(id); } catch (error) { console.error('Erreur lors de la suppression de l’employé:', error); throw error; } } export async function generateBadge(employeeId) { try { return await api.generateBadge(employeeId); } catch (error) { console.error('Erreur lors de la génération du badge:', error); throw error; } }

Src/frontend/js/api/reports.js : const api = window.api; export async function getReports(filters) { try { return await api.getReports(filters); } catch (error) { console.error('Erreur lors de la récupération des rapports:', error); throw error; } }

Src/frontend/js/api/shops.js : const api = window.api; export async function getShops() { try { return await api.getShops(); } catch (error) { console.error('Erreur lors de la récupération des boutiques:', error); throw error; } } export async function createShop(shopData) { try { return await api.createShop(shopData); } catch (error) { console.error('Erreur lors de la création de la boutique:', error); throw error; } } export async function updateShop(id, shopData) { try { return await api.updateShop(id, shopData); } catch (error) { console.error('Erreur lors de la mise à jour de la boutique:', error); throw error; } } export async function deleteShop(id) { try { return await api.deleteShop(id); } catch (error) { console.error('Erreur lors de la suppression de la boutique:', error); throw error; } }

Src/frontend/js/api/users.js : const api = window.api; export async function getUsers() { try { return await api.getUsers(); } catch (error) { console.error('Erreur lors de la récupération des utilisateurs:', error); throw error; } } export async function createUser(userData) { try { return await api.createUser(userData); } catch (error) { console.error('Erreur lors de la création de l’utilisateur:', error); throw error; } } export async function updateUser(id, userData) { try { return await api.updateUser(id, userData); } catch (error) { console.error('Erreur lors de la mise à jour de l’utilisateur:', error); throw error; } } export async function deleteUser(id) { try { return await api.deleteUser(id); } catch (error) { console.error('Erreur lors de la suppression de l’utilisateur:', error); throw error; } }

Src/frontend/js/components/datatable.js : export function initDataTable(tableId, columns, data, actions = {}) { const tableBody = document.querySelector(`#${tableId}`); tableBody.innerHTML = ''; data.forEach(row => { const tr = document.createElement('tr'); columns.forEach(col => { const td = document.createElement('td'); td.textContent = row[col.key] || '-'; tr.appendChild(td); }); if (actions.edit || actions.delete || actions.custom) { const td = document.createElement('td'); if (actions.edit) { const editBtn = document.createElement('button'); editBtn.textContent = '✏'; editBtn.onclick = () => actions.edit(row); td.appendChild(editBtn); } if (actions.delete) { const deleteBtn = document.createElement('button'); deleteBtn.textContent = '🗑'; deleteBtn.onclick = () => actions.delete(row.id); td.appendChild(deleteBtn); } if (actions.custom) { actions.custom.forEach(custom => { const btn = document.createElement('button'); btn.textContent = custom.label; btn.onclick = () => custom.action(row); td.appendChild(btn); }); } tr.appendChild(td); } tableBody.appendChild(tr); }); }

Src/frontend/js/components/modal.js : export function showModal(title, content, buttons = []) { const modal = document.createElement('div'); modal.className = 'modal'; modal.innerHTML = ` <div class="modal-content"> <h2>${title}</h2> <div class="modal-body">${content}</div> <div class="modal-buttons"> ${buttons.map(btn => `<button class="${btn.className}" onclick="${btn.onClick}">${btn.label}</button>`).join('')} </div> </div> `; document.body.appendChild(modal); } export function closeModal() { const modal = document.querySelector('.modal'); if (modal) modal.remove(); }

Src/frontend/js/components/notification.js : export function showNotification(message, type = 'success') { const notification = document.createElement('div'); notification.className = `notification ${type}`; notification.textContent = message; document.body.appendChild(notification); setTimeout(() => { notification.remove(); }, 3000); }

Src/frontend/js/components/qr-scanner.js : import jsQR from 'jsqr'; let videoStream = null; export function initQRScanner(videoId, startBtnId, stopBtnId, onScan) { const video = document.getElementById(videoId); const startBtn = document.getElementById(startBtnId); const stopBtn = document.getElementById(stopBtnId); const canvas = document.createElement('canvas'); const ctx = canvas.getContext('2d'); startBtn.addEventListener('click', async () => { try { videoStream = await navigator.mediaDevices.getUserMedia({ video: { facingMode: 'environment' } }); video.srcObject = videoStream; video.play(); startBtn.disabled = true; stopBtn.disabled = false; scanQRCode(video, canvas, ctx, onScan); } catch (error) { console.error('Erreur lors de l’accès à la caméra:', error); } }); stopBtn.addEventListener('click', () => { if (videoStream) { videoStream.getTracks().forEach(track => track.stop()); videoStream = null; video.srcObject = null; startBtn.disabled = false; stopBtn.disabled = true; } }); } function scanQRCode(video, canvas, ctx, onScan) { if (!video.srcObject) return; canvas.width = video.videoWidth; canvas.height = video.videoHeight; ctx.drawImage(video, 0, 0, canvas.width, canvas.height); const imageData = ctx.getImageData(0, 0, canvas.width, canvas.height); const code = jsQR(imageData.data, imageData.width, imageData.height); if (code) { onScan(code.data); } else { requestAnimationFrame(() => scanQRCode(video, canvas, ctx, onScan)); } }

Src/frontend/js/pages/dashboard.js : import { getEmployees, getShops } from '../api/employees.js'; import { getReports } from '../api/reports.js'; import { initDataTable } from '../components/datatable.js'; import Chart from 'chart.js/auto'; document.addEventListener('DOMContentLoaded', async () => { try { // Charger les statistiques const employees = await (await getEmployees()).data; const shops = await (await getShops()).data; const reports = await (await getReports({ dateFrom: new Date().toISOString().split('T')[0] })).data; document.getElementById('totalEmployees').textContent = employees.length; document.getElementById('totalShops').textContent = shops.length; document.getElementById('presentToday').textContent = reports.filter(r => r.status === 'present').length; document.getElementById('lateToday').textContent = reports.filter(r => r.status === 'late').length; // Tableau des employés initDataTable('employeeTable', [ { key: 'name' }, { key: 'shop\_name' }, { key: 'department' }, { key: 'arrival\_time' }, { key: 'status' }, { key: 'hours' } ], reports); // Graphique par boutique const storeChart = new Chart(document.getElementById('storeChart'), { type: 'bar', data: { labels: shops.map(s => s.name), datasets: [{ label: 'Présence', data: shops.map(s => reports.filter(r => r.shop\_id === s.id && r.status === 'present').length), backgroundColor: '#FF6600' }] } }); // Graphique des heures const hoursChart = new Chart(document.getElementById('hoursChart'), { type: 'line', data: { labels: ['Lun', 'Mar', 'Mer', 'Jeu', 'Ven', 'Sam', 'Dim'], datasets: [{ label: 'Heures travaillées', data: [8, 7.5, 8, 8.5, 7, 6, 0], borderColor: '#FF6600', fill: false }] } }); } catch (error) { console.error('Erreur lors du chargement du dashboard:', error); } });

Src/frontend/js/pages/employees.js : import { getEmployees, createEmployee, updateEmployee, deleteEmployee, generateBadge } from '../api/employees.js'; import { initDataTable } from '../components/datatable.js'; import { showNotification } from '../components/notification.js'; document.addEventListener('DOMContentLoaded', async () => { await loadEmployees(); document.getElementById('addEmployeeForm').addEventListener('submit', async (e) => { e.preventDefault(); const employeeData = { name: document.getElementById('employeeName').value, shop\_id: document.getElementById('employeeStore').value, department: document.getElementById('employeeDepartment').value, schedule: document.getElementById('employeeSchedule').value, hire\_date: document.getElementById('hireDate').value }; try { await createEmployee(employeeData); showNotification('Employé ajouté avec succès !'); await loadEmployees(); e.target.reset(); } catch (error) { showNotification('Erreur lors de l’ajout de l’employé', 'error'); } }); }); async function loadEmployees() { try { const employees = await (await getEmployees()).data; document.getElementById('totalEmployees').textContent = employees.length; initDataTable('employeesTable', [ { key: 'name' }, { key: 'shop\_id' }, { key: 'department' }, { key: 'schedule' }, { key: 'hire\_date' } ], employees, { edit: (employee) => editEmployee(employee), delete: (id) => deleteEmployee(id), custom: [ { label: 'Badge', action: (employee) => generateBadge(employee.id) } ] }); } catch (error) { showNotification('Erreur lors du chargement des employés', 'error'); } } function editEmployee(employee) { // Implémenter la logique pour modifier un employé }

Src/frontend/js/pages/login.js : import { login } from '../api/auth.js'; import { showNotification } from '../components/notification.js'; document.getElementById('loginForm').addEventListener('submit', async (e) => { e.preventDefault(); const username = document.getElementById('username').value; const password = document.getElementById('password').value; try { const response = await login({ username, password }); if (response.success) { const user = JSON.parse(localStorage.getItem('user')); showNotification('Connexion réussie !'); if (user.role === 'admin') { window.location.href = '../../pages/admin/dashboard.html'; } else { window.location.href = '../../pages/manager/time-tracking.html'; } } else { showNotification(response.message, 'error'); } } catch (error) { showNotification('Erreur lors de la connexion', 'error'); } });

Src/frontend/js/pages/reports.js : import { getReports } from '../api/reports.js'; import { initDataTable } from '../components/datatable.js'; import { showNotification } from '../components/notification.js'; document.addEventListener('DOMContentLoaded', () => { document.getElementById('reportForm').addEventListener('submit', async (e) => { e.preventDefault(); const filters = { dateFrom: document.getElementById('dateFrom').value, dateTo: document.getElementById('dateTo').value, shopId: document.getElementById('shop').value, department: document.getElementById('department').value, status: document.getElementById('status').value }; try { const response = await getReports(filters); initDataTable('reportTable', [ { key: 'name' }, { key: 'shop\_name' }, { key: 'department' }, { key: 'date' }, { key: 'arrival\_time' }, { key: 'departure\_time' }, { key: 'breaks' }, { key: 'status' } ], response.data); document.getElementById('downloadExcel').onclick = () => { window.open(response.reportPath); }; } catch (error) { showNotification('Erreur lors de la génération du rapport', 'error'); } }); });

Src/frontend/js/pages/shops.js : import { getShops, createShop, updateShop, deleteShop } from '../api/shops.js'; import { initDataTable } from '../components/datatable.js'; import { showNotification } from '../components/notification.js'; document.addEventListener('DOMContentLoaded', async () => { await loadShops(); document.getElementById('addShopForm').addEventListener('submit', async (e) => { e.preventDefault(); const shopData = { name: document.getElementById('storeName').value, zone: document.getElementById('storeZone').value }; try { await createShop(shopData); showNotification('Boutique ajoutée avec succès !'); await loadShops(); e.target.reset(); } catch (error) { showNotification('Erreur lors de l’ajout de la boutique', 'error'); } }); }); async function loadShops() { try { const shops = await (await getShops()).data; document.getElementById('totalShops').textContent = shops.length; initDataTable('shopsTable', [ { key: 'name' }, { key: 'zone' }, { key: 'employeeCount' } ], shops, { edit: (shop) => editShop(shop), delete: (id) => deleteShop(id) }); } catch (error) { showNotification('Erreur lors du chargement des boutiques', 'error'); } } function editShop(shop) { // Implémenter la logique pour modifier une boutique }

Src/frontend/js/pages/time-tracking.js : import { getEmployees } from '../api/employees.js'; import { recordTime } from '../api/reports.js'; import { initQRScanner } from '../components/qr-scanner.js'; import { showNotification } from '../components/notification.js'; document.addEventListener('DOMContentLoaded', async () => { await loadEmployees(); initQRScanner('qrScanner', 'startScanner', 'stopScanner', async (data) => { const employeeId = data.split('-')[1]; try { await recordTime({ employee\_id: employeeId, type: 'arrival' }); showNotification('Pointage enregistré avec succès !'); } catch (error) { showNotification('Erreur lors du pointage', 'error'); } }); document.getElementById('manualTimeForm').addEventListener('submit', async (e) => { e.preventDefault(); const employeeId = document.getElementById('employeeSelect').value; const type = document.getElementById('timeType').value; try { await recordTime({ employee\_id: employeeId, type }); showNotification('Pointage enregistré avec succès !'); e.target.reset(); } catch (error) { showNotification('Erreur lors du pointage', 'error'); } }); }); async function loadEmployees() { try { const employees = await (await getEmployees()).data; const select = document.getElementById('employeeSelect'); employees.forEach(emp => { const option = document.createElement('option'); option.value = emp.id; option.textContent = emp.name; select.appendChild(option); }); } catch (error) { showNotification('Erreur lors du chargement des employés', 'error'); } }

Src/frontend/js/pages/users.js : import { getUsers, createUser, updateUser, deleteUser } from '../api/users.js'; import { initDataTable } from '../components/datatable.js'; import { showNotification } from '../components/notification.js'; import { hashPassword } from '../utils/encryption.js'; document.addEventListener('DOMContentLoaded', async () => { await loadUsers(); document.getElementById('addUserForm').addEventListener('submit', async (e) => { e.preventDefault(); const userData = { name: document.getElementById('nom').value, username: document.getElementById('identifiant').value, password: await hashPassword(document.getElementById('motdepasse').value), role: document.getElementById('role').value, shop\_id: document.getElementById('boutique').value }; try { await createUser(userData); showNotification('Utilisateur ajouté avec succès !'); await loadUsers(); e.target.reset(); } catch (error) { showNotification('Erreur lors de l’ajout de l’utilisateur', 'error'); } }); }); async function loadUsers() { try { const users = await (await getUsers()).data; document.getElementById('totalUsers').textContent = users.length; document.getElementById('totalAdmins').textContent = users.filter(u => u.role === 'admin').length; document.getElementById('totalManagers').textContent = users.filter(u => u.role === 'manager').length; initDataTable('usersTable', [ { key: 'name' }, { key: 'shop\_id' }, { key: 'username' }, { key: 'password' }, { key: 'role' } ], users, { edit: (user) => editUser(user), delete: (id) => deleteUser(id) }); } catch (error) { showNotification('Erreur lors du chargement des utilisateurs', 'error'); } } function editUser(user) { // Implémenter la logique pour modifier un utilisateur }

Src/frontend/js/utils/constants.js : export const DEPARTMENTS = ['Supervision Réseau', 'Commercial', 'Comptabilité', 'SAV', 'Caissière']; export const SCHEDULES = ['07h30-15h', '07h30-18h', '15h-21h', '07h30-16h']; export const ROLES = ['admin', 'manager']; export const ZONES = ['Douala'];

Src/frontend/js/utils/helpers.js : export function formatDate(date) { return new Date(date).toLocaleDateString('fr-FR'); } export function formatTime(time) { return time ? time.slice(0, 5) : '-'; }

Src/frontend/js/utils/validation.js : export function validateForm(data, schema) { for (const [key, rule] of Object.entries(schema)) { if (rule.required && !data[key]) { return { valid: false, message: `${key} est requis` }; } if (rule.minLength && data[key].length < rule.minLength) { return { valid: false, message: `${key} doit avoir au moins ${rule.minLength} caractères` }; } } return { valid: true }; }

Src/frontend/js/main.js : document.addEventListener('DOMContentLoaded', () => { // Vérifie si l'utilisateur est connecté const token = localStorage.getItem('token'); if (!token && window.location.pathname !== '/src/frontend/pages/auth/login.html') { window.location.href = '/src/frontend/pages/auth/login.html'; } }); function logout() { localStorage.removeItem('token'); localStorage.removeItem('user'); window.location.href = '/src/frontend/pages/auth/login.html'; }

Src/frontend/js/pages/admin/dashboard.html : <!DOCTYPE html> <html lang="fr"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Dashboard - BERTO Tracker</title> <link rel="stylesheet" href="../../css/stylesdashboard.css"> </head> <body> <header> <div class="logo">📊 BERTO Tracker - Admin</div> </header> <div class="container"> <h1>Dashboard de Suivi des Employés</h1> <!-- Filtres --> <div class="filter-section"> <div class="filter-group"> <label>Boutique</label> <select id="storeFilter"> <option value="">Toutes les boutiques</option> <option value="1">Boutique PK 12</option> <option value="2">Boutique Zachman</option> <option value="3">Boutique Dakar</option> </select> </div> <div class="filter-group"> <label>Période</label> <select id="periodFilter"> <option value="today">Aujourd'hui</option> <option value="week">Cette semaine</option> <option value="month">Ce mois</option> </select> </div> <div class="filter-group"> <label>Département</label> <select id="departmentFilter"> <option value="">Tous les départements</option> <option value="Supervision Réseau">Supervision Réseau</option> <option value="Commercial">Commercial</option> <option value="Comptabilité">Comptabilité</option> <option value="SAV">SAV</option> <option value="Caissière">Caissière</option> </select> </div> </div> <!-- Statistiques générales --> <div class="stats-grid"> <div class="stat-card"> <div class="stat-number" id="totalEmployees">0</div> <div class="stat-label">Employés Total</div> </div> <div class="stat-card"> <div class="stat-number" id="presentToday">0</div> <div class="stat-label">Présents Aujourd'hui</div> </div> <div class="stat-card"> <div class="stat-number" id="lateToday">0</div> <div class="stat-label">Retards Aujourd'hui</div> </div> <div class="stat-card"> <div class="stat-number" id="avgHours">0h</div> <div class="stat-label">Moyenne Heures/Jour</div> </div> <div class="stat-card"> <div class="stat-number" id="attendanceRate">0%</div> <div class="stat-label">Taux de Présence</div> </div> <div class="stat-card"> <div class="stat-number" id="overtime">0h</div> <div class="stat-label">Heures Supplémentaires</div> </div> </div> <!-- Alertes --> <div class="alert" id="alerts"></div> <div class="two-column"> <!-- Graphiques --> <div> <div class="card"> <h2>Présence par Boutique</h2> <div class="chart-container"> <div class="chart-placeholder" id="storeChart"> 📊 Graphique en barres<br> Présence par boutique </div> </div> </div> <div class="card"> <h2>Évolution des Heures</h2> <div class="chart-container"> <div class="chart-placeholder" id="hoursChart"> 📈 Graphique linéaire<br> Heures travaillées sur 30 jours </div> </div> </div> </div> <!-- Tableaux de bord --> <div> <div class="card"> <h2>Top Retardataires (Cette Semaine)</h2> <table class="table"> <thead> <tr> <th>Employé</th> <th>Boutique</th> <th>Retards</th> <th>Temps Moyen</th> </tr> </thead> <tbody id="lateEmployees"> <tr> <td>Jarvis</td> <td>Boutique PK12</td> <td>4</td> <td>12 min</td> </tr> </tbody> </table> </div> <div class="card"> <h2>Performance par Boutique</h2> <div style="margin-bottom: 1rem;" id="shopPerformance"> <strong>Boutique PK12</strong> <div class="progress-bar"> <div class="progress-fill" style="width: 92%"></div> </div> <small>92% - 23/25 employés présents</small> </div> <div style="margin-bottom: 1rem;" id="shopPerformance"> <strong>Boutique Zachman</strong> <div class="progress-bar"> <div class="progress-fill" style="width: 88%"></div> </div> <small>88% - 22/25 employés présents</small> </div> <div style="margin-bottom: 1rem;" id="shopPerformance"> <strong>Boutique Dakar</strong> <div class="progress-bar"> <div class="progress-fill" style="width: 85%"></div> </div> <small>85% - 17/20 employés présents</small> </div> </div> </div> </div> <!-- Tableau détaillé --> <div class="card"> <h2>État Actuel des Employés</h2> <table class="table"> <thead> <tr> <th>Nom</th> <th>Boutique</th> <th>Département</th> <th>Arrivée</th> <th>Statut</th> <th>Heures Aujourd'hui</th> </tr> </thead> <tbody id="employeeTable"> <tr> <td>Lislaure</td> <td>Boutique PK12</td> <td>Supervision Réseau</td> <td>08:00</td> <td><span class="status-badge status-present">Ponctuel</span></td> <td>6h 15min</td> </tr> <tr> <td>Francine</td> <td>Boutique Zachman</td> <td>Supervision Réseau</td> <td>08:15</td> <td><span class="status-badge status-late">En Retard</span></td> <td>5h 45min</td> </tr> <tr> <td>Armelle</td> <td>Boutique Dakar</td> <td>Caissière</td> <td>07:45</td> <td><span class="status-badge status-present">Présent</span></td> <td>7h 30min</td> </tr> <tr> <td>Leonel</td> <td>Boutique Zachman</td> <td>Commercial</td> <td>-</td> <td><span class="status-badge status-absent">Absent</span></td> <td>0h 00min</td> </tr> </tr> </tbody> </table> </div> <!-- Statistiques détaillées --> <div class="two-column"> <div class="card"> <h2>Répartition par Horaires</h2> <div style="margin-bottom: 1rem;" id="scheduleDistribution"> <strong>7h30-15h</strong> <div class="progress-bar"> <div class="progress-fill" style="width: 78%"></div> </div> <small>45 employés - 78% présents</small> </div> <div style="margin-bottom: 1rem;" id="scheduleDistribution"> <strong>15h-21h</strong> <div class="progress-bar"> <div class="progress-fill" style="width: 92%"></div> </div> <small>52 employés - 92% présents</small> </div> <div id="scheduleDistribution"> <strong>7h30-18h</strong> <div class="progress-bar"> <div class="progress-fill" style="width: 85%"></div> </div> <small>30 employés - 85% présents</small> </div> </div> <div class="card"> <h2>Tendances de la Semaine</h2> <table class="table"> <thead> <tr> <th>Jour</th> <th>Présents</th> <th>Retards</th> <th>Absences</th> </tr> </thead> <tbody id="weekTrends"> <tr> <td>Lundi</td> <td>118</td> <td>12</td> <td>9</td> </tr> <tr> <td>Mardi</td> <td>122</td> <td>8</td> <td>5</td> </tr> <tr> <td>Mercredi</td> <td>119</td> <td>10</td> <td>8</td> </tr> <tr> <td>Jeudi</td> <td>116</td> <td>15</td> <td>11</td> </tr> <tr> <td>Vendredi</td> <td>95</td> <td>8</td> <td>32</td> </tr> <tr> <td>Samedi</td> <td>95</td> <td>8</td> <td>32</td> </tr> </tbody> </table> </div> </div> <!-- Actions rapides --> <div class="nav-buttons"> <button onclick="window.location.href='shops.html'">🏨 Boutiques</button> <button onclick="window.location.href='users.html'">👷‍♀️ Utilisateurs</button> <button onclick="window.location.href='employees.html'">🙎‍♂️ Employés</button> <button onclick="window.location.href='global-reports.html'">📝 Rapports</button> <button onclick="logout()">Déconnexion</button> </div> </div> <script type="module" src="https://cdn.jsdelivr.net/npm/chart.js"></script> <script type="module" src="../../js/pages/dashboard.js"></script> </body> </html>

Src/frontend/js/pages/admin/employees.html : <!DOCTYPE html> <html lang="fr"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Gestion des Employés - BERTO Tracker</title> <link rel="stylesheet" href="../../css/stylesemployees.css"> </head> <body> <header> <div class="logo">👥 BERTO Tracker - Gestion des Employés</div> </header> <div class="container"> <h1>Gestion des Employés</h1> <!-- Statistiques rapides --> <div class="stats-summary"> <div class="stat-card"> <div class="stat-number" id="totalEmployees">0</div> <div class="stat-label">Total Employés</div> </div> <div class="stat-card"> <div class="stat-number" id="totalShops">0</div> <div class="stat-label">Boutiques</div> </div> <div class="stat-card"> <div class="stat-number" id="totalDepartments">0</div> <div class="stat-label">Départements</div> </div> <div class="stat-card"> <div class="stat-number" id="newEmployees">0</div> <div class="stat-label">Nouveaux ce mois</div> </div> </div> <!-- Formulaire d'ajout d'employé --> <div class="card"> <h2>➕ Ajouter un Nouvel Employé</h2> <form id="addEmployeeForm"> <div class="form-row"> <div class="form-group"> <label for="employeeName">Nom complet \*</label> <input type="text" id="employeeName" name="employeeName" placeholder="Ex: Jarvis" required> </div> <div class="form-group"> <label for="employeeStore">Boutique d'affectation \*</label> <select id="employeeStore" name="employeeStore" required> <option value="">Sélectionner une boutique</option> <option value="1">Boutique PK 12</option> <option value="2">Boutique Zachman</option> <option value="3">Boutique Dakar</option> </select> </div> </div> <div class="form-row"> <div class="form-group"> <label for="employeeDepartment">Département \*</label> <select id="employeeDepartment" name="employeeDepartment" required> <option value="">Sélectionner un département</option> <option value="Supervision Réseau">Supervision Réseau</option> <option value="Commercial">Commercial</option> <option value="Comptabilité">Comptabilité</option> <option value="SAV">SAV</option> <option value="Caissière">Caissière</option> </select> </div> <div class="form-group"> <label for="employeeSchedule">Horaires de travail \*</label> <select id="employeeSchedule" name="employeeSchedule" required> <option value="">Sélectionner les horaires</option> <option value="07h30-15h">Équipe Matin (07h30 - 15h00)</option> <option value="07h30-18h">Journée Complète (07h30 - 18h00)</option> <option value="15h-21h">Équipe Soir (15h00 - 21h00)</option> <option value="07h30-16h">Journée Standard (07h30 - 16h00)</option> </select> </div> </div> <div class="form-group"> <label for="hireDate">Date d'embauche \*</label> <input type="date" id="hireDate" name="hireDate" required> </div> <div class="form-actions"> <button type="submit">✅ Ajouter l'Employé</button> <button type="reset" class="btn-secondary">🔄 Réinitialiser</button> </div> </form> </div> <!-- Filtres et recherche --> <div class="card"> <h2>🔍 Rechercher et Filtrer</h2> <div class="search-filter"> <input type="text" placeholder="Rechercher par nom..." id="searchName"> <select id="filterStore"> <option value="">Toutes les boutiques</option> <option value="1">Boutique PK 12</option> <option value="2">Boutique Zachman</option> <option value="3">Boutique Dakar</option> </select> <select id="filterDepartment"> <option value="">Tous les départements</option> <option value="Supervision Réseau">Supervision Réseau</option> <option value="Commercial">Commercial</option> <option value="Comptabilité">Comptabilité</option> <option value="SAV">SAV</option> <option value="Caissière">Caissière</option> </select> <select id="filterSchedule"> <option value="">Tous les horaires</option> <option value="07h30-15h">07h30 - 15h00</option> <option value="07h30-18h">07h30 - 18h00</option> <option value="15h-21h">15h00 - 21h00</option> <option value="07h30-16h">07h30 - 16h00</option> </select> </div> </div> <!-- Tableau des employés --> <div class="card"> <h2>📋 Liste des Employés</h2> <table class="table"> <thead> <tr> <th>Nom</th> <th>Boutique</th> <th>Département</th> <th>Horaires</th> <th>Date d'embauche</th> <th>Actions</th> </tr> </thead> <tbody > <tr> <td><strong>Jarvis</strong></td> <td>Boutique PK12</td> <td><span class="badge badge-department">Supervision Réseau</span></td> <td><span class="badge badge-schedule">07h30 - 18h00</span></td> <td>02/12/2024</td> <td> <div class="action-buttons"> <button class="btn-success">🆔 Badge</button> <button class="btn-secondary">✏️ Modifier</button> <button class="btn-danger">🗑️ Supprimer</button> </div> </td> </tr> </tbody> </table> </div> <!-- Actions rapides --> <div class="nav-buttons"> <button onclick="window.location.href='dashboard.html'">📈 Dashboard</button> <button onclick="window.location.href='shops.html'">🏨 Boutiques</button> <button onclick="window.location.href='users.html'">👷‍♀️ Utilisateurs</button> <button onclick="window.location.href='employees.html'">🙎‍♂️ Employés</button> <button onclick="window.location.href='global-reports.html'">📝 Rapports</button> <button onclick="logout()">Déconnexion</button> </div> </div> <script type="module" src="../../js/pages/employees.js"></script> </body> </html>

Src/frontend/js/pages/admin/global-reports.html : <!DOCTYPE html> <html lang="fr"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Rapports Globaux - BERTO Tracker</title> <link rel="stylesheet" href="../../css/stylesglore.css"> </head> <body> <header> <div class="logo">📊 BERTO Tracker - Rapports</div> </header> <div class="container"> <h1>Rapports Globaux</h1> <!-- Onglets de période --> <div class="period-tabs"> <button class="period-tab active">📅 Quotidien</button> <button class="period-tab">📊 Hebdomadaire</button> <button class="period-tab">📈 Mensuel</button> </div> <!-- Filtres --> <div class="card"> <h2>Filtres de Génération</h2> <div class="filters-grid"> <div class="form-group"> <label for="dateFrom">Date de début</label> <input type="date" id="dateFrom" value="2024-01-01"> </div> <div class="form-group"> <label for="dateTo">Date de fin</label> <input type="date" id="dateTo" value="2024-01-31"> </div> <div class="form-group"> <label for="shop">Boutique</label> <select id="shop" name="shop"> <option value="all">Toutes les boutiques</option> <option value="1">Boutique PK 12</option> <option value="2">Boutique Zachman</option> <option value="3">Boutique Dakar</option> </select> </div> <div class="form-group"> <label for="department">Département</label> <select id="department" name="department"> <option value="">Tous les départements</option> <option value="Supervision Réseau">Supervision Réseau</option> <option value="Commercial">Commercial</option> <option value="Comptabilité">Comptabilité</option> <option value="SAV">SAV</option> <option value="Caissière">Caissière</option> </select> </div> <div class="form-group"> <label for="employeeSelect">Employé</label> <select id="employeeSelect"> <option value="all">Tous les employés</option> <option value="Jarvis">Jarvis</option> <option value="Morel">Morel</option> </select> </div> <div class="form-group"> <label for="status">Statut</label> <select id="status" name="status"> <option value="">Tous les statuts</option> <option value="present">Présents uniquement</option> <option value="absent">Absents uniquement</option> <option value="late">Retards uniquement</option> </select> </div> </div> <div class="form-group"> <label>Colonnes à inclure</label> <div class="checkbox-group"> <div class="checkbox-item"> <input type="checkbox" id="col-name" checked> <label for="col-name">Nom employé</label> </div> <div class="checkbox-item"> <input type="checkbox" id="col-store" checked> <label for="col-store">Boutique</label> </div> <div class="checkbox-item"> <input type="checkbox" id="col-department" checked> <label for="col-department">Département</label> </div> <div class="checkbox-item"> <input type="checkbox" id="col-arrival" checked> <label for="col-arrival">Heure d'arrivée</label> </div> <div class="checkbox-item"> <input type="checkbox" id="col-departure" checked> <label for="col-departure">Heure de départ</label> </div> <div class="checkbox-item"> <input type="checkbox" id="col-breaks" checked> <label for="col-breaks">Pauses</label> </div> <div class="checkbox-item"> <input type="checkbox" id="col-total" checked> <label for="col-total">Heures totales</label> </div> <div class="checkbox-item"> <input type="checkbox" id="col-overtime"> <label for="col-overtime">Heures supplémentaires</label> </div> <div class="checkbox-item"> <input type="checkbox" id="col-status" checked> <label for="col-status">Statut</label> </div> <div class="checkbox-item"> <input type="checkbox" id="col-notes"> <label for="col-notes">Notes</label> </div> </div> </div> </div> <!-- Types de rapports et téléchargement --> <div class="two-column"> <div> <div class="card"> <h2>Types de Rapports</h2> <div class="report-type-grid"> <div class="report-card selected"> <div class="report-icon">📋</div> <div class="report-title">Rapport de Présence</div> <div class="report-description">Détails des arrivées, départs et heures travaillées par employé</div> </div> <div class="report-card"> <div class="report-icon">⏰</div> <div class="report-title">Rapport des Retards</div> <div class="report-description">Analyse des retards avec durées et fréquences</div> </div> <div class="report-card"> <div class="report-icon">📊</div> <div class="report-title">Statistiques par Boutique</div> <div class="report-description">Comparaison des performances entre boutiques</div> </div> <div class="report-card"> <div class="report-icon">💰</div> <div class="report-title">Heures Supplémentaires</div> <div class="report-description">Détail des heures supplémentaires par employé</div> </div> <div class="report-card"> <div class="report-icon">🏥</div> <div class="report-title">Absences et Congés</div> <div class="report-description">Suivi des absences justifiées et non justifiées</div> </div> <div class="report-card"> <div class="report-icon">📈</div> <div class="report-title">Rapport Complet</div> <div class="report-description">Toutes les données combinées en un seul fichier</div> </div> </div> </div> </div> <div> <div class="card"> <div class="download-section"> <div class="download-icon">📥</div> <h2>Télécharger le Rapport</h2> <p style="margin-bottom: 1.5rem;">Générer et télécharger le rapport Excel selon vos critères</p> <button class="btn-primary">📊 Générer Excel</button> <br><br> <button class="btn-secondary">👁️ Aperçu</button> </div> </div> <div class="card"> <h2>Actions Rapides</h2> <div class="nav-buttons"> <button>📅 Rapport Aujourd'hui</button> <button>📊 Semaine Actuelle</button> <button>📈 Mois Dernier</button> <button>⚙️ Modèles</button> </div> </div> </div> </div> <!-- Historique des téléchargements --> <div class="card"> <h2>Téléchargements Récents</h2> <div class="recent-downloads"> <div class="download-item"> <div class="download-info"> <div class="download-name">Rapport\_Presence\_Janvier\_2024.xlsx</div> <div class="download-meta">Généré le 15/01/2024 à 14:30 - Toutes boutiques - 2.3 MB</div> </div> <div class="download-actions"> <span class="status-badge status-ready">Prêt</span> <button class="btn-small">📥 Télécharger</button> <button class="btn-small btn-secondary">🗑️</button> </div> </div> <div class="download-item"> <div class="download-info"> <div class="download-name">Retards\_Semaine\_02\_2024.xlsx</div> <div class="download-meta">Généré le 14/01/2024 à 09:15 - Paris Centre - 456 KB</div> </div> <div class="download-actions"> <span class="status-badge status-ready">Prêt</span> <button class="btn-small">📥 Télécharger</button> <button class="btn-small btn-secondary">🗑️</button> </div> </div> <div class="download-item"> <div class="download-info"> <div class="download-name">Heures\_Supp\_Decembre\_2023.xlsx</div> <div class="download-meta">En cours de génération... - Toutes boutiques</div> </div> <div class="download-actions"> <span class="status-badge status-processing">En cours</span> <button class="btn-small" disabled>⏳ Traitement</button> <button class="btn-small btn-secondary">❌</button> </div> </div> <div class="download-item"> <div class="download-info"> <div class="download-name">Stats\_Boutiques\_Q4\_2023.xlsx</div> <div class="download-meta">Généré le 12/01/2024 à 16:45 - Lyon Bellecour - 1.8 MB</div> </div> <div class="download-actions"> <span class="status-badge status-ready">Prêt</span> <button class="btn-small">📥 Télécharger</button> <button class="btn-small btn-secondary">🗑️</button> </div> </div> <div class="download-item"> <div class="download-info"> <div class="download-name">Rapport\_Complet\_2023.xlsx</div> <div class="download-meta">Erreur lors de la génération - Données insuffisantes</div> </div> <div class="download-actions"> <span class="status-badge status-error">Erreur</span> <button class="btn-small">🔄 Régénérer</button> <button class="btn-small btn-secondary">🗑️</button> </div> </div> </div> </div> <!-- Actions rapides --> <div class="nav-buttons"> <button onclick="window.location.href='dashboard.html'">📈 Dashboard</button> <button onclick="window.location.href='shops.html'">🏨 Boutiques</button> <button onclick="window.location.href='users.html'">👷‍♀️ Utilisateurs</button> <button onclick="window.location.href='employees.html'">🙎‍♂️ Employés</button> <button onclick="window.location.href='global-reports.html'">📝 Rapports</button> <button onclick="logout()">Déconnexion</button> </div> </div> <footer class="footer"> <p>© 2025 BERTO Tracker - Rapports globaux</p> </footer> <script type="module" src="../../js/pages/reports.js"></script> </body> </html>

Src/frontend/js/pages/admin/shops.html : <!DOCTYPE html> <html lang="fr"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Gestion des Boutiques - BERTO Tracker</title> <link rel="stylesheet" href="../../css/stylesshops.css"> </head> <body> <header> <div class="logo">🏪 BERTO Tracker - Gestion des Boutiques</div> </header> <div class="container"> <h1>Gestion des Boutiques</h1> <!-- Statistiques rapides --> <div class="stats-summary"> <div class="stat-card"> <div class="stat-number" id="totalShops">0</div> <div class="stat-label">Boutiques Actives</div> </div> <div class="stat-card"> <div class="stat-number" id="totalEmployees">0</div> <div class="stat-label">Employés Total</div> </div> <div class="stat-card"> <div class="stat-number" id="totalZones">0</div> <div class="stat-label">Zones Couvertes</div> </div> </div> <!-- Message de succès --> <div class="success-message" id="successMessage" style="display: none;">✅ Boutique ajoutée avec succès !</div> <div class="form-section"> <!-- Formulaire d'ajout --> <div class="card"> <h2>Ajouter une Boutique</h2> <form id="addShopForm"> <div class="form-group"> <label for="storeName">Nom de la boutique \*</label> <input type="text" id="storeName" name="storeName" required placeholder="Ex: Boutique PK 12"> </div> <div class="form-group"> <label for="storeZone">Zone \*</label> <select id="storeZone" name="storeZone" required> <option value="">Sélectionner une zone</option> <option value="Douala">Douala</option> </select> </div> <button type="submit">➕ Ajouter la Boutique</button> </form> </div> <!-- Liste des boutiques --> <div class="card"> <h2>Boutiques Existantes</h2> <p>Gérez vos boutiques existantes, modifiez leurs informations ou supprimez-les.</p> <table class="table"> <thead> <tr> <th>Nom de la Boutique</th> <th>Zone</th> <th>Employés</th> <th>Actions</th> </tr> </thead> <tbody> <tr> <td>Boutique PK12</td> <td>Douala</td> <td>25</td> <td> <div class="action-buttons"> <button class="edit-btn">✏️ Modifier</button> <button class="delete-btn">🗑️ Supprimer</button> </div> </td> </tr> </tbody> </table> </div> </div> <!-- Actions rapides --> <div class="nav-buttons"> <button onclick="window.location.href='dashboard.html'">📈 Dashboard</button> <button onclick="window.location.href='shops.html'">🏨 Boutiques</button> <button onclick="window.location.href='users.html'">👷‍♀️ Utilisateurs</button> <button onclick="window.location.href='employees.html'">🙎‍♂️ Employés</button> <button onclick="window.location.href='global-reports.html'">📝 Rapports</button> <button onclick="logout()">Déconnexion</button> </div> <!-- Informations supplémentaires --> <div class="card"> <h2>Informations</h2> <p><strong>📝 Note :</strong> La suppression d'une boutique entraînera la suppression de tous les employés associés et de leur historique. Cette action est irréversible.</p> <p><strong>🔄 Modification :</strong> Vous pouvez modifier le nom et la zone d'une boutique à tout moment. Les employés associés seront automatiquement mis à jour.</p> <p><strong>📊 Statistiques :</strong> Le nombre d'employés affiché correspond au nombre d'employés actifs dans chaque boutique.</p> </div> </div> <footer class="footer"> <p>© 2025 BERTO Tracker - Gestion des boutiques</p> </footer> <script type="module" src="../../js/pages/shops.js"></script> </body> </html>

Src/frontend/js/pages/admin/users.html : <!DOCTYPE html> <html lang="fr"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Gestion des Utilisateurs - BERTO Tracker</title> <link rel="stylesheet" href="../../css/stylesusers.css"> </head> <body> <header> <div class="logo">👥 BERTO Tracker - Gestion Utilisateurs</div> </header> <div class="container"> <h1>Gestion des Utilisateurs</h1> <!-- Statistiques --> <div class="stats-section"> <div class="stat-card"> <div class="stat-number" id="totalUsers">0</div> <div class="stat-label">Total Utilisateurs</div> </div> <div class="stat-card"> <div class="stat-number" id="totalAdmins">0</div> <div class="stat-label">Administrateurs</div> </div> <div class="stat-card"> <div class="stat-number" id="totalManagers">0</div> <div class="stat-label">Managers</div> </div> <div class="stat-card"> <div class="stat-number" id="totalShops">0</div> <div class="stat-label">Boutiques</div> </div> </div> <div class="two-column"> <!-- Formulaire d'ajout --> <div class="card"> <h2>Ajouter un Utilisateur</h2> <form id="addUserForm"> <div class="form-group"> <label for="nom">Nom complet</label> <input type="text" id="nom" name="nom" placeholder="Ex: Jean Dupont" required> </div> <div class="form-group"> <label for="identifiant">Identifiant de connexion</label> <input type="text" id="identifiant" name="identifiant" placeholder="Ex: j.dupont" required> </div> <div class="form-group"> <label for="boutique">Boutique d'affectation</label> <select id="boutique" name="boutique" required> <option value="">Sélectionner une boutique</option> <option value="1">Boutique PK 12</option> <option value="2">Boutique Zachman</option> </select> </div> <div class="form-group"> <label for="motdepasse">Mot de passe</label> <input type="password" id="motdepasse" name="motdepasse" placeholder="Mot de passe sécurisé" required> </div> <div class="form-group"> <label for="role">Rôle</label> <select id="role" name="role" required> <option value="">Sélectionner un rôle</option> <option value="admin">Administrateur</option> <option value="manager">Manager</option> </select> </div> <button type="submit">➕ Ajouter l'utilisateur</button> </form> </div> <!-- Liste des utilisateurs --> <div class="card"> <h2>Liste des Utilisateurs</h2> <!-- Recherche et filtres --> <div class="search-section"> <div class="search-input"> <input type="text" id="searchInput" placeholder="🔍 Rechercher par nom ou identifiant..." style="width: 100%; padding: 0.6rem; border: 1px solid #ccc; border-radius: var(--radius);"> </div> <div class="filter-select"> <select id="filterShop" style="width: 100%; padding: 0.6rem; border: 1px solid #ccc; border-radius: var(--radius);"> <option value="">Toutes les boutiques</option> <option value="1">Boutique PK 12</option> <option value="2">Boutique Zachman</option> <option value="3">Boutique Dakar</option> </select> </div> <div class="filter-select"> <select id="filterRole" style="width: 100%; padding: 0.6rem; border: 1px solid #ccc; border-radius: var(--radius);"> <option value="">Tous les rôles</option> <option value="admin">Administrateur</option> <option value="manager">Manager</option> </select> </div> </div> <!-- Tableau des utilisateurs --> <div style="overflow-x: auto;"> <table class="table"> <thead> <tr> <th>Nom</th> <th>Boutique</th> <th>Identifiant</th> <th>Mot de passe</th> <th>Rôle</th> <th class="actions-column">Actions</th> </tr> </thead> <tbody> <tr> <td>Admin</td> <td></td> <td>admin</td> <td>••••••••</td> <td><span class="role-badge role-admin">Admin</span></td> <td> <button class="btn-edit">✏️ Modifier</button> <button class="btn-delete">🗑️ Supprimer</button> </td> </tr> <tr> <td>Manager</td> <td></td> <td>manager\_pk12</td> <td>••••••••</td> <td><span class="role-badge role-manager">Manager</span></td> <td> <button class="btn-edit">✏️ Modifier</button> <button class="btn-delete">🗑️ Supprimer</button> </td> </tr> </tbody> </table> </div> </div> </div> <!-- Actions rapides --> <div class="nav-buttons"> <button onclick="window.location.href='dashboard.html'">📈 Dashboard</button> <button onclick="window.location.href='shops.html'">🏨 Boutiques</button> <button onclick="window.location.href='users.html'">👷‍♀️ Utilisateurs</button> <button onclick="window.location.href='employees.html'">🙎‍♂️ Employés</button> <button onclick="window.location.href='global-reports.html'">📝 Rapports</button> <button onclick="logout()">Déconnexion</button> </div> </div> <script type="module" src="../../js/pages/users.js"></script> </body> </html>

Src/frontend/js/pages/auth/login.html : <!DOCTYPE html> <html lang="fr"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Connexion - BERTO Tracker</title> <link rel="stylesheet" href="../../css/styleslogin.css"> </head> <body> <header> <div class="logo">📊 BERTO Tracker</div> </header> <div class="login-container"> <div class="login-card"> <div class="login-header"> <h1 class="login-title">Connexion</h1> <p class="login-subtitle">Accédez à votre espace de gestion</p> </div> <div class="company-info"> <h3>Système de Pointage</h3> <p>Gestion centralisée pour toutes vos boutiques</p> </div> <form id="loginForm"> <div class="form-group"> <label for="username">Identifiant</label> <div class="input-icon user-icon"> <input type="text" id="username" name="username" placeholder="Votre identifiant" required > </div> </div> <div class="form-group"> <label for="password">Mot de passe</label> <div class="input-icon password-icon"> <input type="password" id="password" name="password" placeholder="Votre mot de passe" required > </div> </div> <button type="submit" class="login-button"> Se connecter </button> </form> <div class="login-footer"> <a href="#" class="forgot-password">Mot de passe oublié ?</a> </div> </div> </div> <footer class="footer"> <p>© 2025 BERTO Tracker - Système de gestion des temps de travail</p> </footer> <script type="module" src="../../js/pages/login.js"></script> </body> </html>

Src/frontend/js/pages/manager/shop-reports.html : <!DOCTYPE html> <html lang="fr"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Rapports Boutique - BERTO Tracker</title> <link rel="stylesheet" href="../../css/stylesshore.css"> </head> <body> <header> <div class="logo">📊 BERTO Tracker - Rapports Boutique</div> </header> <div class="container"> <h1>Rapports de la Boutique</h1> <!-- Information boutique --> <div class="store-info"> <h3>🏪 Boutique : PK 12</h3> <p>Manager : Patricia | 25 employés actifs |</p> </div> <!-- Sélection du type de rapport --> <div class="card"> <h2>Type de Rapport</h2> <div class="report-type-grid"> <div class="report-type-card selected"> <div class="report-icon">📅</div> <h3>Quotidien</h3> <p>Rapport journalier des présences et horaires</p> </div> <div class="report-type-card"> <div class="report-icon">📊</div> <h3>Hebdomadaire</h3> <p>Synthèse de la semaine avec heures travaillées</p> </div> <div class="report-type-card"> <div class="report-icon">📈</div> <h3>Mensuel</h3> <p>Rapport complet du mois avec statistiques</p> </div> </div> </div> <!-- Filtres --> <div class="card"> <h2>Filtres de Période</h2> <div class="form-row"> <div class="form-group"> <label for="startDate">Date de début</label> <input type="date" id="startDate" value="2025-01-01"> </div> <div class="form-group"> <label for="endDate">Date de fin</label> <input type="date" id="endDate" value="2025-01-31"> </div> <div class="form-group"> <label for="quickSelect">Sélection rapide</label> <select id="quickSelect"> <option value="">Sélectionner une période</option> <option value="today">Aujourd'hui</option> <option value="yesterday">Hier</option> <option value="thisweek">Cette semaine</option> <option value="lastweek">Semaine dernière</option> <option value="thismonth">Ce mois</option> <option value="lastmonth">Mois dernier</option> </select> </div> </div> </div> <!-- Filtres avancés --> <div class="filter-section"> <h3>Filtres Avancés</h3> <div class="form-row"> <div class="form-group"> <label for="department">Département</label> <select id="department"> <option value="">Tous les départements</option> <option value="Supervision Réseau">Supervision Réseau</option> <option value="Commercial">Commercial</option> <option value="Comptabilité">Comptabilité</option> <option value="SAV">SAV</option> <option value="Caissière">Caissière</option> </select> </div> <div class="form-group"> <label for="shift">Équipe</label> <select id="shift"> <option value="">Toutes les équipes</option> <option value="matin">Équipe matin (7h30-15h)</option> <option value="jour">Équipe jour (7h30-17h)</option> <option value="soir">Équipe soir (15h-21h)</option> </select> </div> </div> </div> <!-- Sélection des employés --> <div class="card"> <h2>Sélection des Employés</h2> <div class="form-row"> <div class="form-group"> <label for="employeeSelect">Employés spécifiques</label> <select id="employeeSelect" multiple size="8"> <option value="all" selected>Tous les employés (25)</option> <option value="Jarvis">Jarvis - Supervision Réseau</option> <option value="Patricia">Patricia - SAV</option> <option value="Carine">Carine - Caissière</option> <option value="Leonel">Leonel - Commercial</option> </select> </div> <div class="form-group"> <label>Actions rapides</label> <button type="button" style="width: 100%; margin-bottom: 0.5rem;">Sélectionner tout</button> <button type="button" style="width: 100%; margin-bottom: 0.5rem;">Désélectionner tout</button> <button type="button" style="width: 100%; margin-bottom: 0.5rem;">Sélectionner équipe matin</button> <button type="button" style="width: 100%;">Sélectionner équipe soir</button> </div> </div> </div> <!-- Résumé --> <div class="summary-section"> <h3>Résumé du Rapport</h3> <div class="summary-item"> <span>Type de rapport :</span> <span>Quotidien</span> </div> <div class="summary-item"> <span>Période :</span> <span>01/01/2025 - 31/01/2025</span> </div> <div class="summary-item"> <span>Boutique :</span> <span>PK 12</span> </div> <div class="summary-item"> <span>Employés sélectionnés :</span> <span>25</span> </div> <div class="summary-item"> <span>Format :</span> <span>Excel (.xlsx)</span> </div> <div class="summary-item"> <span>Taille estimée :</span> <span>~ 2.5 MB</span> </div> </div> <!-- Téléchargement --> <div class="download-section"> <h2>Télécharger le Rapport</h2> <p style="margin-bottom: 2rem; color: #666;"> Le rapport sera généré au format Excel et téléchargé automatiquement.<br> Temps de génération estimé : 30 secondes </p> <button class="download-button"> 📥 Télécharger le Rapport Excel </button> <p style="margin-top: 1rem; font-size: 0.9rem; color: #666;"> Le fichier sera nommé : rapport\_boutique\_pk\_12\_01-01-2025\_31-01-2025.xlsx </p> </div> <!-- Historique des téléchargements récents --> <div class="card"> <h2>Téléchargements Récents</h2> <div style="margin-top: 1rem;"> <div style="display: flex; justify-content: space-between; align-items: center; padding: 0.75rem; border-bottom: 1px solid #e0e0e0;"> <div> <strong>Rapport Hebdomadaire</strong><br> <small>16/12/2024 - 22/12/2024 | 25 employés</small> </div> <div style="text-align: right;"> <small>Téléchargé le 23/12/2024</small><br> <a href="#" style="color: var(--orange); text-decoration: none;">Retélécharger</a> </div> </div> <div style="display: flex; justify-content: space-between; align-items: center; padding: 0.75rem; border-bottom: 1px solid #e0e0e0;"> <div> <strong>Rapport Mensuel</strong><br> <small>01/11/2024 - 30/11/2024 | 25 employés</small> </div> <div style="text-align: right;"> <small>Téléchargé le 01/12/2024</small><br> <a href="#" style="color: var(--orange); text-decoration: none;">Retélécharger</a> </div> </div> <div style="display: flex; justify-content: space-between; align-items: center; padding: 0.75rem;"> <div> <strong>Rapport Quotidien</strong><br> <small>15/11/2024 | 23 employés</small> </div> <div style="text-align: right;"> <small>Téléchargé le 16/11/2024</small><br> <a href="#" style="color: var(--orange); text-decoration: none;">Retélécharger</a> </div> </div> </div> </div> <!-- Actions rapides --> <div class="nav-buttons"> <button onclick="window.location.href='time-tracking.html'">⏰ Pointage</button> <button onclick="window.location.href='shop-reports.html'">📝 Rapports Boutique</button> <button onclick="logout()">Déconnexion</button> </div> </div> <footer class="footer"> <p>© 2025 BERTO Tracker - Rapports boutique</p> </footer> <script type="module" src="../../js/pages/reports.js"></script> </body> </html>

Src/frontend/js/pages/manager/time-tracking.html : <!DOCTYPE html> <html lang="fr"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Pointage - BERTO Tracker</title> <link rel="stylesheet" href="../../css/stylestitra.css"> </head> <body> <header> <div class="logo">⏰ BERTO Tracker - Manager</div> </header> <div class="container"> <h1>Pointage des Employés</h1> <!-- Bouton Scanner --> <div class="nav-buttons"> <button onclick="toggleScanner()">📱 Démarrer le Scanner</button> <button>💾 Sauvegarder les Heures</button> <button>📊 Voir le Récapitulatif</button> </div> <!-- Zone Scanner (masquée par défaut) --> <div id="scanner-container"> <h2>📷 Scanner de Badge</h2> <div id="scanner-video"> <div class="scanner-placeholder"> <div>📹</div> <div style="margin-top: 1rem;">Caméra activée</div> <div style="font-size: 0.9rem; opacity: 0.8; margin-top: 0.5rem;"> Placez le badge dans le cadre orange </div> </div> <div class="scanner-overlay"></div> </div> <div class="scanner-info"> <p><strong>Instructions :</strong></p> <p>• Tenez le badge bien droit dans le cadre orange</p> <p>• Assurez-vous que le code-barres est visible</p> <p>• Attendez le bip de confirmation</p> </div> <div class="scanner-controls"> <button class="btn-small">🔄 Réinitialiser</button> <button class="btn-small">❌ Fermer Scanner</button> </div> </div> <!-- Tableau des employés --> <div class="card"> <h2>👥 Employés de la Boutique (25 employés)</h2> <table class="table"> <thead> <tr> <th>ID</th> <th>Nom</th> <th>Département</th> <th>Heure d'Arrivée</th> <th>Heure de Départ</th> <th>Statut</th> <th>Actions</th> </tr> </thead> <tbody> <tr> <td>001</td> <td>Alice Johnson</td> <td>Vente</td> <td><input type="time" class="time-input" value="08:00"></td> <td><input type="time" class="time-input" placeholder="--:--"></td> <td><span class="status-badge status-present">Présent</span></td> <td> <div class="action-buttons"> <button class="btn-small">✓</button> <button class="btn-small">✏️</button> </div> </td> </tr> <tr> <td>002</td> <td>Bob Wilson</td> <td>Caisse</td> <td><input type="time" class="time-input" value="08:15"></td> <td><input type="time" class="time-input" placeholder="--:--"></td> <td><span class="status-badge status-late">Retard</span></td> <td> <div class="action-buttons"> <button class="btn-small">✓</button> <button class="btn-small">✏️</button> </div> </td> </tr> <tr> <td>003</td> <td>Claire Martin</td> <td>Management</td> <td><input type="time" class="time-input" value="07:45"></td> <td><input type="time" class="time-input" placeholder="--:--"></td> <td><span class="status-badge status-present">Présent</span></td> <td> <div class="action-buttons"> <button class="btn-small">✓</button> <button class="btn-small">✏️</button> </div> </td> </tr> <tr> <td>004</td> <td>David Brown</td> <td>Stock</td> <td><input type="time" class="time-input" placeholder="--:--"></td> <td><input type="time" class="time-input" placeholder="--:--"></td> <td><span class="status-badge status-absent">Absent</span></td> <td> <div class="action-buttons"> <button class="btn-small">✓</button> <button class="btn-small">✏️</button> </div> </td> </tr> <tr> <td>005</td> <td>Emma Davis</td> <td>Vente</td> <td><input type="time" class="time-input" value="08:05"></td> <td><input type="time" class="time-input" placeholder="--:--"></td> <td><span class="status-badge status-present">Présent</span></td> <td> <div class="action-buttons"> <button class="btn-small">✓</button> <button class="btn-small">✏️</button> </div> </td> </tr> <tr> <td>006</td> <td>Frank Miller</td> <td>Vente</td> <td><input type="time" class="time-input" value="09:00"></td> <td><input type="time" class="time-input" placeholder="--:--"></td> <td><span class="status-badge status-present">Présent</span></td> <td> <div class="action-buttons"> <button class="btn-small">✓</button> <button class="btn-small">✏️</button> </div> </td> </tr> <tr> <td>007</td> <td>Grace Lee</td> <td>Caisse</td> <td><input type="time" class="time-input" value="08:30"></td> <td><input type="time" class="time-input" placeholder="--:--"></td> <td><span class="status-badge status-present">Présent</span></td> <td> <div class="action-buttons"> <button class="btn-small">✓</button> <button class="btn-small">✏️</button> </div> </td> </tr> <tr> <td>008</td> <td>Henry Taylor</td> <td>Stock</td> <td><input type="time" class="time-input" placeholder="--:--"></td> <td><input type="time" class="time-input" placeholder="--:--"></td> <td><span class="status-badge status-absent">Absent</span></td> <td> <div class="action-buttons"> <button class="btn-small">✓</button> <button class="btn-small">✏️</button> </div> </td> </tr> </tbody> </table> </div> <!-- Résumé du jour --> <div class="card"> <h2>📊 Résumé du Jour</h2> <div style="display: grid; grid-template-columns: repeat(auto-fit, minmax(200px, 1fr)); gap: 1rem;"> <div style="text-align: center; padding: 1rem; background: #e8f5e8; border-radius: var(--radius);"> <div style="font-size: 2rem; color: #2e7d2e; font-weight: bold;">6</div> <div>Employés Présents</div> </div> <div style="text-align: center; padding: 1rem; background: #ffe6e6; border-radius: var(--radius);"> <div style="font-size: 2rem; color: #e74c3c; font-weight: bold;">2</div> <div>Employés Absents</div> </div> <div style="text-align: center; padding: 1rem; background: #fff3cd; border-radius: var(--radius);"> <div style="font-size: 2rem; color: #856404; font-weight: bold;">1</div> <div>Retards</div> </div> <div style="text-align: center; padding: 1rem; background: #d1ecf1; border-radius: var(--radius);"> <div style="font-size: 2rem; color: #0c5460; font-weight: bold;">42h</div> <div>Heures Travaillées</div> </div> </div> </div> <!-- Actions rapides --> <div class="nav-buttons"> <button onclick="window.location.href='time-tracking.html'">⏰ Pointage</button> <button onclick="window.location.href='shop-reports.html'">📝 Rapports Boutique</button> <button onclick="logout()">Déconnexion</button> </div> </div> <footer class="footer"> <p>© 2025 BERTO Tracker - Pointage des employés</p> </footer> <script type="module" src="../../js/components/qr-scanner.js"></script> <script type="module" src="../../js/pages/time-tracking.js"></script> <script> function toggleScanner() { const scanner = document.getElementById('scanner-container'); if (scanner.style.display === 'none' || scanner.style.display === '') { scanner.style.display = 'block'; scanner.scrollIntoView({ behavior: 'smooth' }); } else { scanner.style.display = 'none'; } } </script> </body> </html>

.gitignore : node\_modules/ dist/ data/\*.db logs/\* output/\* .env

Src/frontend/css/stylesdashboard.css : @import url('https://fonts.googleapis.com/css2?family=Montserrat:wght@600&family=Roboto&display=swap'); :root { --orange: #FF6600; --dark: #333333; --light: #F4F4F4; --white: #FFFFFF; --radius: 8px; --transition: 0.2s ease; } \* { box-sizing: border-box; margin: 0; padding: 0; } body { font-family: 'Roboto', sans-serif; background: var(--light); color: var(--dark); } header { background: var(--white); box-shadow: 0 2px 6px rgba(0,0,0,0.1); padding: 1rem; } header .logo { font-family: 'Montserrat', sans-serif; color: var(--orange); font-size: 1.5rem; } .container { max-width: 1200px; margin: 2rem auto; padding: 0 1rem; } h1, h2 { font-family: 'Montserrat', sans-serif; color: var(--orange); margin-bottom: 1rem; } .nav-buttons { display: grid; grid-template-columns: repeat(auto-fill, minmax(140px, 1fr)); gap: 1rem; margin-bottom: 2rem; } button { font-family: inherit; cursor: pointer; border: none; background: var(--orange); color: var(--white); padding: 0.75rem 1rem; border-radius: var(--radius); transition: background var(--transition), transform var(--transition); } button:hover { background: #e55a00; transform: translateY(-2px); } .card { background: var(--white); border-radius: var(--radius); padding: 1.5rem; box-shadow: 0 2px 8px rgba(0,0,0,0.05); margin-bottom: 2rem; } .stats-grid { display: grid; grid-template-columns: repeat(auto-fit, minmax(250px, 1fr)); gap: 1.5rem; margin-bottom: 2rem; } .stat-card { background: var(--white); border-radius: var(--radius); padding: 1.5rem; box-shadow: 0 2px 8px rgba(0,0,0,0.05); text-align: center; transition: transform var(--transition); } .stat-card:hover { transform: translateY(-3px); } .stat-number { font-size: 2.5rem; font-weight: bold; color: var(--orange); font-family: 'Montserrat', sans-serif; } .stat-label { font-size: 0.9rem; color: var(--dark); margin-top: 0.5rem; } .chart-container { height: 300px; position: relative; background: var(--white); border-radius: var(--radius); padding: 1rem; margin-bottom: 2rem; } .chart-placeholder { height: 100%; background: linear-gradient(135deg, #ff6600, #ff8533); border-radius: var(--radius); display: flex; align-items: center; justify-content: center; color: var(--white); font-size: 1.2rem; } .table { width: 100%; border-collapse: collapse; margin-bottom: 1rem; } .table th, .table td { padding: 0.75rem 1rem; border: 1px solid #e0e0e0; text-align: left; } .table th { background: var(--orange); color: var(--white); } .table tr:nth-child(even) { background: #f9f9f9; } .status-badge { padding: 0.25rem 0.75rem; border-radius: 20px; font-size: 0.8rem; font-weight: bold; } .status-present { background: #e8f5e8; color: #2e7d2e; } .status-absent { background: #ffe6e6; color: #e74c3c; } .status-late { background: #fff3cd; color: #856404; } .filter-section { display: flex; gap: 1rem; margin-bottom: 2rem; flex-wrap: wrap; } .filter-group { display: flex; flex-direction: column; gap: 0.5rem; } .filter-group select, .filter-group input { padding: 0.6rem; border: 1px solid #ccc; border-radius: var(--radius); font-family: inherit; } .two-column { display: grid; grid-template-columns: 1fr 1fr; gap: 2rem; } @media (max-width: 768px) { .two-column { grid-template-columns: 1fr; } .stats-grid { grid-template-columns: repeat(auto-fit, minmax(200px, 1fr)); } } .progress-bar { width: 100%; height: 20px; background: #e0e0e0; border-radius: 10px; overflow: hidden; margin: 0.5rem 0; } .progress-fill { height: 100%; background: var(--orange); transition: width 0.3s ease; } .alert { background: #fff3cd; color: #856404; padding: 1rem; border-radius: var(--radius); margin-bottom: 1rem; border-left: 4px solid #ffc107; }

Src/frontend/css/stylesemployees.css : @import url('https://fonts.googleapis.com/css2?family=Montserrat:wght@600&family=Roboto&display=swap'); :root { --orange: #FF6600; --dark: #333333; --light: #F4F4F4; --white: #FFFFFF; --radius: 8px; --transition: 0.2s ease; } \* { box-sizing: border-box; margin: 0; padding: 0; } body { font-family: 'Roboto', sans-serif; background: var(--light); color: var(--dark); } header { background: var(--white); box-shadow: 0 2px 6px rgba(0,0,0,0.1); padding: 1rem; } header .logo { font-family: 'Montserrat', sans-serif; color: var(--orange); font-size: 1.5rem; } .container { max-width: 1200px; margin: 2rem auto; padding: 0 1rem; } h1, h2 { font-family: 'Montserrat', sans-serif; color: var(--orange); margin-bottom: 1rem; } .nav-buttons { display: grid; grid-template-columns: repeat(auto-fill, minmax(140px, 1fr)); gap: 1rem; margin-bottom: 2rem; } button { font-family: inherit; cursor: pointer; border: none; background: var(--orange); color: var(--white); padding: 0.75rem 1rem; border-radius: var(--radius); transition: background var(--transition), transform var(--transition); } button:hover { background: #e55a00; transform: translateY(-2px); } .card { background: var(--white); border-radius: var(--radius); padding: 1.5rem; box-shadow: 0 2px 8px rgba(0,0,0,0.05); margin-bottom: 2rem; } .table { width: 100%; border-collapse: collapse; margin-bottom: 1rem; } .table th, .table td { padding: 0.75rem 1rem; border: 1px solid #e0e0e0; } .table th { background: var(--orange); color: var(--white); text-align: left; } .form-group { margin-bottom: 1rem; } .form-group label { display: block; margin-bottom: 0.5rem; font-weight: 500; } .form-group input, .form-group select { width: 100%; padding: 0.6rem; border: 1px solid #ccc; border-radius: var(--radius); font-family: inherit; } .form-row { display: grid; grid-template-columns: 1fr 1fr; gap: 1rem; } .form-actions { display: flex; gap: 1rem; margin-top: 1.5rem; } .btn-secondary { background: #6c757d; } .btn-secondary:hover { background: #5a6268; } .btn-danger { background: #dc3545; padding: 0.5rem 1rem; font-size: 0.9rem; } .btn-danger:hover { background: #c82333; } .btn-success { background: #28a745; padding: 0.5rem 1rem; font-size: 0.9rem; } .btn-success:hover { background: #218838; } .table tr:nth-child(even) { background: #f9f9f9; } .table tr:hover { background: #f0f0f0; } .action-buttons { display: flex; gap: 0.5rem; } .badge { display: inline-block; padding: 0.25rem 0.75rem; font-size: 0.8rem; font-weight: 500; border-radius: 20px; text-align: center; } .badge-department { background: #e3f2fd; color: #1976d2; } .badge-schedule { background: #f3e5f5; color: #7b1fa2; } .stats-summary { display: grid; grid-template-columns: repeat(auto-fit, minmax(200px, 1fr)); gap: 1rem; margin-bottom: 2rem; } .stat-card { background: var(--white); padding: 1rem; border-radius: var(--radius); text-align: center; box-shadow: 0 2px 8px rgba(0,0,0,0.05); } .stat-number { font-size: 1.8rem; font-weight: bold; color: var(--orange); font-family: 'Montserrat', sans-serif; } .stat-label { font-size: 0.9rem; color: var(--dark); margin-top: 0.25rem; } .search-filter { display: flex; gap: 1rem; margin-bottom: 1rem; flex-wrap: wrap; } .search-filter input, .search-filter select { flex: 1; min-width: 200px; } @media (max-width: 768px) { .form-row { grid-template-columns: 1fr; } .form-actions { flex-direction: column; } .action-buttons { flex-direction: column; } .search-filter { flex-direction: column; } }

Src/frontend/css/stylesglore.css : @import url('https://fonts.googleapis.com/css2?family=Montserrat:wght@600&family=Roboto&display=swap'); :root { --orange: #FF6600; --dark: #333333; --light: #F4F4F4; --white: #FFFFFF; --radius: 8px; --transition: 0.2s ease; } \* { box-sizing: border-box; margin: 0; padding: 0; } body { font-family: 'Roboto', sans-serif; background: var(--light); color: var(--dark); } header { background: var(--white); box-shadow: 0 2px 6px rgba(0,0,0,0.1); padding: 1rem; } header .logo { font-family: 'Montserrat', sans-serif; color: var(--orange); font-size: 1.5rem; } .container { max-width: 1200px; margin: 2rem auto; padding: 0 1rem; } h1, h2 { font-family: 'Montserrat', sans-serif; color: var(--orange); margin-bottom: 1rem; } .nav-buttons { display: grid; grid-template-columns: repeat(auto-fill, minmax(140px, 1fr)); gap: 1rem; margin-bottom: 2rem; } button { font-family: inherit; cursor: pointer; border: none; background: var(--orange); color: var(--white); padding: 0.75rem 1rem; border-radius: var(--radius); transition: background var(--transition), transform var(--transition); } button:hover { background: #e55a00; transform: translateY(-2px); } .card { background: var(--white); border-radius: var(--radius); padding: 1.5rem; box-shadow: 0 2px 8px rgba(0,0,0,0.05); margin-bottom: 2rem; } .table { width: 100%; border-collapse: collapse; margin-bottom: 1rem; } .table th, .table td { padding: 0.75rem 1rem; border: 1px solid #e0e0e0; } .table th { background: var(--orange); color: var(--white); text-align: left; } .form-group { margin-bottom: 1rem; } .form-group label { display: block; margin-bottom: 0.5rem; font-weight: 500; } .form-group input, .form-group select { width: 100%; padding: 0.6rem; border: 1px solid #ccc; border-radius: var(--radius); } .filters-grid { display: grid; grid-template-columns: repeat(auto-fit, minmax(200px, 1fr)); gap: 1rem; margin-bottom: 2rem; } .period-tabs { display: flex; gap: 0.5rem; margin-bottom: 2rem; background: var(--white); padding: 0.5rem; border-radius: var(--radius); box-shadow: 0 2px 8px rgba(0,0,0,0.05); } .period-tab { flex: 1; background: transparent; color: var(--dark); border: 1px solid #ddd; padding: 0.75rem 1rem; border-radius: var(--radius); transition: all var(--transition); } .period-tab.active { background: var(--orange); color: var(--white); border-color: var(--orange); } .period-tab:hover { transform: none; background: var(--orange); color: var(--white); border-color: var(--orange); } .report-type-grid { display: grid; grid-template-columns: repeat(auto-fit, minmax(280px, 1fr)); gap: 1rem; margin-bottom: 2rem; } .report-card { background: var(--white); border: 2px solid #e0e0e0; border-radius: var(--radius); padding: 1.5rem; cursor: pointer; transition: all var(--transition); } .report-card:hover { border-color: var(--orange); transform: translateY(-2px); } .report-card.selected { border-color: var(--orange); box-shadow: 0 4px 12px rgba(255, 102, 0, 0.2); } .report-icon { font-size: 2rem; margin-bottom: 1rem; color: var(--orange); } .report-title { font-family: 'Montserrat', sans-serif; font-size: 1.1rem; color: var(--dark); margin-bottom: 0.5rem; } .report-description { font-size: 0.9rem; color: #666; } .two-column { display: grid; grid-template-columns: 2fr 1fr; gap: 2rem; } .download-section { text-align: center; padding: 2rem; background: #f8f9ff; border-radius: var(--radius); border: 2px dashed var(--orange); } .download-icon { font-size: 3rem; color: var(--orange); margin-bottom: 1rem; } .btn-primary { background: var(--orange); color: var(--white); padding: 1rem 2rem; font-size: 1.1rem; font-weight: 600; } .btn-secondary { background: transparent; color: var(--orange); border: 2px solid var(--orange); } .btn-secondary:hover { background: var(--orange); color: var(--white); } .checkbox-group { display: grid; grid-template-columns: repeat(auto-fit, minmax(200px, 1fr)); gap: 0.5rem; margin: 1rem 0; } .checkbox-item { display: flex; align-items: center; gap: 0.5rem; } .checkbox-item input[type="checkbox"] { width: auto; margin: 0; } .recent-downloads { max-height: 400px; overflow-y: auto; } .download-item { display: flex; justify-content: space-between; align-items: center; padding: 1rem; border-bottom: 1px solid #e0e0e0; } .download-item:last-child { border-bottom: none; } .download-info { flex: 1; } .download-name { font-weight: 500; margin-bottom: 0.25rem; } .download-meta { font-size: 0.85rem; color: #666; } .download-actions { display: flex; gap: 0.5rem; } .btn-small { padding: 0.25rem 0.75rem; font-size: 0.85rem; } .status-badge { padding: 0.25rem 0.75rem; border-radius: 20px; font-size: 0.8rem; font-weight: bold; } .status-ready { background: #e8f5e8; color: #2e7d2e; } .status-processing { background: #fff3cd; color: #856404; } .status-error { background: #ffe6e6; color: #e74c3c; } @media (max-width: 768px) { .two-column { grid-template-columns: 1fr; } .filters-grid { grid-template-columns: 1fr; } .period-tabs { flex-direction: column; } }

Src/frontend/css/styleslogin.css : @import url('https://fonts.googleapis.com/css2?family=Montserrat:wght@600&family=Roboto&display=swap'); :root { --orange: #FF6600; --dark: #333333; --light: #F4F4F4; --white: #FFFFFF; --radius: 8px; --transition: 0.2s ease; } \* { box-sizing: border-box; margin: 0; padding: 0; } body { font-family: 'Roboto', sans-serif; background: var(--light); color: var(--dark); min-height: 100vh; display: flex; flex-direction: column; } header { background: var(--white); box-shadow: 0 2px 6px rgba(0,0,0,0.1); padding: 1rem; } header .logo { font-family: 'Montserrat', sans-serif; color: var(--orange); font-size: 1.5rem; text-align: center; } .login-container { flex: 1; display: flex; align-items: center; justify-content: center; padding: 2rem 1rem; } .login-card { background: var(--white); border-radius: var(--radius); padding: 3rem; box-shadow: 0 4px 20px rgba(0,0,0,0.1); width: 100%; max-width: 400px; } .login-header { text-align: center; margin-bottom: 2rem; } .login-title { font-family: 'Montserrat', sans-serif; color: var(--orange); font-size: 1.8rem; margin-bottom: 0.5rem; } .login-subtitle { color: var(--dark); font-size: 0.9rem; opacity: 0.7; } .form-group { margin-bottom: 1.5rem; } .form-group label { display: block; margin-bottom: 0.5rem; font-weight: 500; color: var(--dark); } .form-group input { width: 100%; padding: 0.8rem; border: 2px solid #e0e0e0; border-radius: var(--radius); font-family: inherit; font-size: 1rem; transition: border-color var(--transition); } .form-group input:focus { outline: none; border-color: var(--orange); } .form-group input::placeholder { color: #999; } .login-button { width: 100%; font-family: inherit; cursor: pointer; border: none; background: var(--orange); color: var(--white); padding: 0.9rem 1rem; border-radius: var(--radius); font-size: 1rem; font-weight: 600; transition: background var(--transition), transform var(--transition); margin-top: 1rem; } .login-button:hover { background: #e55a00; transform: translateY(-2px); } .login-footer { text-align: center; margin-top: 2rem; padding-top: 1.5rem; border-top: 1px solid #e0e0e0; } .forgot-password { color: var(--orange); text-decoration: none; font-size: 0.9rem; transition: opacity var(--transition); } .forgot-password:hover { opacity: 0.8; } .footer { text-align: center; padding: 1rem; font-size: 0.9rem; color: var(--dark); background: var(--white); border-top: 1px solid #e0e0e0; } .company-info { text-align: center; margin-bottom: 2rem; padding: 1rem; background: rgba(255, 102, 0, 0.05); border-radius: var(--radius); border-left: 4px solid var(--orange); } .company-info h3 { font-family: 'Montserrat', sans-serif; color: var(--orange); margin-bottom: 0.5rem; font-size: 1.1rem; } .company-info p { font-size: 0.9rem; color: var(--dark); opacity: 0.8; } .input-icon { position: relative; } .input-icon::before { content: ''; position: absolute; left: 12px; top: 50%; transform: translateY(-50%); width: 16px; height: 16px; background-size: contain; opacity: 0.5; } .user-icon::before { background-image: url(''); } .password-icon::before { background-image: url('data:image/svg+xml;base64,'); } .input-icon input { padding-left: 2.5rem; } @media (max-width: 480px) { .login-card { padding: 2rem 1.5rem; margin: 1rem; } .login-title { font-size: 1.5rem; } } /\* Animation subtile au chargement \*/ .login-card { animation: fadeInUp 0.6s ease-out; } @keyframes fadeInUp { from { opacity: 0; transform: translateY(30px); } to { opacity: 1; transform: translateY(0); } }

Src/frontend/css/stylesshops.css : @import url('https://fonts.googleapis.com/css2?family=Montserrat:wght@600&family=Roboto&display=swap'); :root { --orange: #FF6600; --dark: #333333; --light: #F4F4F4; --white: #FFFFFF; --radius: 8px; --transition: 0.2s ease; } \* { box-sizing: border-box; margin: 0; padding: 0; } body { font-family: 'Roboto', sans-serif; background: var(--light); color: var(--dark); } header { background: var(--white); box-shadow: 0 2px 6px rgba(0,0,0,0.1); padding: 1rem; } header .logo { font-family: 'Montserrat', sans-serif; color: var(--orange); font-size: 1.5rem; } .container { max-width: 1200px; margin: 2rem auto; padding: 0 1rem; } h1, h2 { font-family: 'Montserrat', sans-serif; color: var(--orange); margin-bottom: 1rem; } .nav-buttons { display: grid; grid-template-columns: repeat(auto-fill, minmax(140px, 1fr)); gap: 1rem; margin-bottom: 2rem; } button { font-family: inherit; cursor: pointer; border: none; background: var(--orange); color: var(--white); padding: 0.75rem 1rem; border-radius: var(--radius); transition: background var(--transition), transform var(--transition); } button:hover { background: #e55a00; transform: translateY(-2px); } .card { background: var(--white); border-radius: var(--radius); padding: 1.5rem; box-shadow: 0 2px 8px rgba(0,0,0,0.05); margin-bottom: 2rem; } .table { width: 100%; border-collapse: collapse; margin-bottom: 1rem; } .table th, .table td { padding: 0.75rem 1rem; border: 1px solid #e0e0e0; } .table th { background: var(--orange); color: var(--white); text-align: left; } .table tr:nth-child(even) { background: #f9f9f9; } .form-group { margin-bottom: 1rem; } .form-group label { display: block; margin-bottom: 0.5rem; font-weight: 500; } .form-group input, .form-group select { width: 100%; padding: 0.6rem; border: 1px solid #ccc; border-radius: var(--radius); font-family: inherit; font-size: 1rem; } .form-group input:focus, .form-group select:focus { outline: none; border-color: var(--orange); box-shadow: 0 0 0 2px rgba(255, 102, 0, 0.1); } .footer { text-align: center; padding: 1rem; font-size: 0.9rem; color: var(--dark); } .card p { font-size: 1rem; margin: 0.5rem 0; } .error-message { background: #ffe6e6; color: #e74c3c; padding: 0.75rem; border-radius: var(--radius); margin-bottom: 1rem; text-align: center; } .success-message { background: #e8f5e8; color: #2e7d2e; padding: 0.75rem; border-radius: var(--radius); margin-bottom: 1rem; text-align: center; } .form-section { display: grid; grid-template-columns: 1fr 2fr; gap: 2rem; margin-bottom: 2rem; } .delete-btn { background: #e74c3c; color: var(--white); border: none; padding: 0.5rem 1rem; border-radius: var(--radius); cursor: pointer; font-size: 0.9rem; transition: background var(--transition); } .delete-btn:hover { background: #c0392b; } .edit-btn { background: #3498db; color: var(--white); border: none; padding: 0.5rem 1rem; border-radius: var(--radius); cursor: pointer; font-size: 0.9rem; margin-right: 0.5rem; transition: background var(--transition); } .edit-btn:hover { background: #2980b9; } .action-buttons { display: flex; gap: 0.5rem; } .stats-summary { display: grid; grid-template-columns: repeat(auto-fit, minmax(200px, 1fr)); gap: 1rem; margin-bottom: 2rem; } .stat-card { background: var(--white); border-radius: var(--radius); padding: 1.5rem; box-shadow: 0 2px 8px rgba(0,0,0,0.05); text-align: center; } .stat-number { font-size: 2rem; font-weight: bold; color: var(--orange); font-family: 'Montserrat', sans-serif; } .stat-label { font-size: 0.9rem; color: var(--dark); margin-top: 0.5rem; } .breadcrumb { margin-bottom: 1rem; color: #666; } .breadcrumb a { color: var(--orange); text-decoration: none; } .breadcrumb a:hover { text-decoration: underline; } @media (max-width: 768px) { .form-section { grid-template-columns: 1fr; gap: 1rem; } .stats-summary { grid-template-columns: 1fr; } .action-buttons { flex-direction: column; } .edit-btn { margin-right: 0; margin-bottom: 0.5rem; } }

Src/frontend/css/stylesshore.css : @import url('https://fonts.googleapis.com/css2?family=Montserrat:wght@600&family=Roboto&display=swap'); :root { --orange: #FF6600; --dark: #333333; --light: #F4F4F4; --white: #FFFFFF; --radius: 8px; --transition: 0.2s ease; } \* { box-sizing: border-box; margin: 0; padding: 0; } body { font-family: 'Roboto', sans-serif; background: var(--light); color: var(--dark); } header { background: var(--white); box-shadow: 0 2px 6px rgba(0,0,0,0.1); padding: 1rem; } header .logo { font-family: 'Montserrat', sans-serif; color: var(--orange); font-size: 1.5rem; } .container { max-width: 1200px; margin: 2rem auto; padding: 0 1rem; } h1, h2 { font-family: 'Montserrat', sans-serif; color: var(--orange); margin-bottom: 1rem; } button { font-family: inherit; cursor: pointer; border: none; background: var(--orange); color: var(--white); padding: 0.75rem 1rem; border-radius: var(--radius); transition: background var(--transition), transform var(--transition); } button:hover { background: #e55a00; transform: translateY(-2px); } .card { background: var(--white); border-radius: var(--radius); padding: 1.5rem; box-shadow: 0 2px 8px rgba(0,0,0,0.05); margin-bottom: 2rem; } .form-group { margin-bottom: 1rem; } .form-group label { display: block; margin-bottom: 0.5rem; font-weight: 500; } .form-group input, .form-group select { width: 100%; padding: 0.6rem; border: 1px solid #ccc; border-radius: var(--radius); font-family: inherit; } .form-row { display: grid; grid-template-columns: repeat(auto-fit, minmax(250px, 1fr)); gap: 1rem; } .report-type-grid { display: grid; grid-template-columns: repeat(auto-fit, minmax(200px, 1fr)); gap: 1rem; margin-bottom: 2rem; } .report-type-card { background: var(--white); border: 2px solid #e0e0e0; border-radius: var(--radius); padding: 1.5rem; text-align: center; cursor: pointer; transition: all var(--transition); } .report-type-card:hover { border-color: var(--orange); transform: translateY(-2px); } .report-type-card.selected { border-color: var(--orange); background: #fff5f0; } .report-type-card h3 { color: var(--orange); margin-bottom: 0.5rem; font-family: 'Montserrat', sans-serif; } .report-type-card p { font-size: 0.9rem; color: #666; } .report-icon { font-size: 2rem; margin-bottom: 1rem; } .filter-section { background: #f9f9f9; padding: 1.5rem; border-radius: var(--radius); margin-bottom: 2rem; } .filter-section h3 { color: var(--dark); margin-bottom: 1rem; font-family: 'Montserrat', sans-serif; } .checkbox-group { display: grid; grid-template-columns: repeat(auto-fit, minmax(200px, 1fr)); gap: 0.5rem; margin-top: 1rem; } .checkbox-item { display: flex; align-items: center; gap: 0.5rem; } .checkbox-item input[type="checkbox"] { width: auto; margin: 0; } .summary-section { background: #f0f8ff; border: 1px solid #b3d9ff; border-radius: var(--radius); padding: 1.5rem; margin-bottom: 2rem; } .summary-section h3 { color: var(--dark); margin-bottom: 1rem; font-family: 'Montserrat', sans-serif; } .summary-item { display: flex; justify-content: space-between; margin-bottom: 0.5rem; padding: 0.25rem 0; border-bottom: 1px solid #e0e0e0; } .summary-item:last-child { border-bottom: none; font-weight: bold; } .download-section { text-align: center; padding: 2rem; background: var(--white); border-radius: var(--radius); box-shadow: 0 2px 8px rgba(0,0,0,0.05); } .download-button { background: var(--orange); color: var(--white); padding: 1rem 2rem; border: none; border-radius: var(--radius); font-size: 1.1rem; font-weight: bold; cursor: pointer; transition: all var(--transition); min-width: 200px; } .download-button:hover { background: #e55a00; transform: translateY(-2px); } .download-button:disabled { background: #ccc; cursor: not-allowed; transform: none; } .store-info { background: #e8f5e8; border: 1px solid #4caf50; border-radius: var(--radius); padding: 1rem; margin-bottom: 2rem; } .store-info h3 { color: #2e7d2e; margin-bottom: 0.5rem; font-family: 'Montserrat', sans-serif; } .breadcrumb { background: var(--white); padding: 1rem; border-radius: var(--radius); margin-bottom: 2rem; box-shadow: 0 2px 8px rgba(0,0,0,0.05); } .breadcrumb a { color: var(--orange); text-decoration: none; } .breadcrumb a:hover { text-decoration: underline; } @media (max-width: 768px) { .form-row { grid-template-columns: 1fr; } .report-type-grid { grid-template-columns: 1fr; } }

Src/frontend/css/stylestitra.css : @import url('https://fonts.googleapis.com/css2?family=Montserrat:wght@600&family=Roboto&display=swap'); :root { --orange: #FF6600; --dark: #333333; --light: #F4F4F4; --white: #FFFFFF; --radius: 8px; --transition: 0.2s ease; } \* { box-sizing: border-box; margin: 0; padding: 0; } body { font-family: 'Roboto', sans-serif; background: var(--light); color: var(--dark); } header { background: var(--white); box-shadow: 0 2px 6px rgba(0,0,0,0.1); padding: 1rem; } header .logo { font-family: 'Montserrat', sans-serif; color: var(--orange); font-size: 1.5rem; } .container { max-width: 1200px; margin: 2rem auto; padding: 0 1rem; } h1, h2 { font-family: 'Montserrat', sans-serif; color: var(--orange); margin-bottom: 1rem; } .nav-buttons { display: grid; grid-template-columns: repeat(auto-fill, minmax(140px, 1fr)); gap: 1rem; margin-bottom: 2rem; } button { font-family: inherit; cursor: pointer; border: none; background: var(--orange); color: var(--white); padding: 0.75rem 1rem; border-radius: var(--radius); transition: background var(--transition), transform var(--transition); } button:hover { background: #e55a00; transform: translateY(-2px); } .card { background: var(--white); border-radius: var(--radius); padding: 1.5rem; box-shadow: 0 2px 8px rgba(0,0,0,0.05); margin-bottom: 2rem; } .table { width: 100%; border-collapse: collapse; margin-bottom: 1rem; } .table th, .table td { padding: 0.75rem 1rem; border: 1px solid #e0e0e0; } .table th { background: var(--orange); color: var(--white); text-align: left; } .table tr:nth-child(even) { background: #f9f9f9; } .form-group { margin-bottom: 1rem; } .form-group label { display: block; margin-bottom: 0.5rem; } .form-group input, .form-group select { width: 100%; padding: 0.6rem; border: 1px solid #ccc; border-radius: var(--radius); } .footer { text-align: center; padding: 1rem; font-size: 0.9rem; color: var(--dark); } /\* Styles pour le scanner \*/ #scanner-container { position: relative; margin: 1rem 0; display: none; background: var(--white); border-radius: var(--radius); padding: 1.5rem; box-shadow: 0 2px 8px rgba(0,0,0,0.05); } #scanner-video { border-radius: var(--radius); background: var(--dark); width: 100%; max-width: 400px; height: 300px; margin: 0 auto; display: block; position: relative; } .scanner-placeholder { width: 100%; height: 100%; background: linear-gradient(135deg, #333, #555); border-radius: var(--radius); display: flex; flex-direction: column; align-items: center; justify-content: center; color: var(--white); font-size: 1.1rem; } .scanner-overlay { position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%); width: 200px; height: 200px; border: 2px solid var(--orange); border-radius: 10px; background: rgba(255, 102, 0, 0.1); } .scanner-info { text-align: center; margin-top: 1rem; color: var(--dark); } .card p { font-size: 1rem; margin: 0.5rem 0; } .error-message { background: #ffe6e6; color: #e74c3c; padding: 0.75rem; border-radius: var(--radius); margin-bottom: 1rem; text-align: center; } .store-info { background: var(--orange); color: var(--white); padding: 1rem; border-radius: var(--radius); margin-bottom: 2rem; text-align: center; } .time-input { max-width: 100px; text-align: center; } .status-badge { padding: 0.25rem 0.75rem; border-radius: 20px; font-size: 0.8rem; font-weight: bold; } .status-present { background: #e8f5e8; color: #2e7d2e; } .status-absent { background: #ffe6e6; color: #e74c3c; } .status-late { background: #fff3cd; color: #856404; } .action-buttons { display: flex; gap: 0.5rem; justify-content: center; } .btn-small { padding: 0.4rem 0.8rem; font-size: 0.85rem; min-width: auto; } .scanner-controls { display: flex; gap: 1rem; justify-content: center; margin-top: 1rem; } .current-time { font-size: 1.2rem; font-weight: bold; color: var(--orange); text-align: center; margin-bottom: 1rem; }

Src/frontend/css/stylesusers.css : @import url('https://fonts.googleapis.com/css2?family=Montserrat:wght@600&family=Roboto&display=swap'); :root { --orange: #FF6600; --dark: #333333; --light: #F4F4F4; --white: #FFFFFF; --radius: 8px; --transition: 0.2s ease; } \* { box-sizing: border-box; margin: 0; padding: 0; } body { font-family: 'Roboto', sans-serif; background: var(--light); color: var(--dark); } header { background: var(--white); box-shadow: 0 2px 6px rgba(0,0,0,0.1); padding: 1rem; } header .logo { font-family: 'Montserrat', sans-serif; color: var(--orange); font-size: 1.5rem; } .container { max-width: 1200px; margin: 2rem auto; padding: 0 1rem; } h1, h2 { font-family: 'Montserrat', sans-serif; color: var(--orange); margin-bottom: 1rem; } .nav-buttons { display: grid; grid-template-columns: repeat(auto-fill, minmax(140px, 1fr)); gap: 1rem; margin-bottom: 2rem; } button { font-family: inherit; cursor: pointer; border: none; background: var(--orange); color: var(--white); padding: 0.75rem 1rem; border-radius: var(--radius); transition: background var(--transition), transform var(--transition); } button:hover { background: #e55a00; transform: translateY(-2px); } .card { background: var(--white); border-radius: var(--radius); padding: 1.5rem; box-shadow: 0 2px 8px rgba(0,0,0,0.05); margin-bottom: 2rem; } .table { width: 100%; border-collapse: collapse; margin-bottom: 1rem; } .table th, .table td { padding: 0.75rem 1rem; border: 1px solid #e0e0e0; } .table th { background: var(--orange); color: var(--white); text-align: left; } .table tr:nth-child(even) { background: #f9f9f9; } .form-group { margin-bottom: 1rem; } .form-group label { display: block; margin-bottom: 0.5rem; font-weight: 500; color: var(--dark); } .form-group input, .form-group select { width: 100%; padding: 0.6rem; border: 1px solid #ccc; border-radius: var(--radius); font-family: inherit; font-size: 1rem; } .form-group input:focus, .form-group select:focus { outline: none; border-color: var(--orange); box-shadow: 0 0 0 2px rgba(255, 102, 0, 0.1); } .form-grid { display: grid; grid-template-columns: repeat(auto-fit, minmax(250px, 1fr)); gap: 1rem; margin-bottom: 1.5rem; } .btn-delete { background: #e74c3c; color: white; border: none; padding: 0.5rem 1rem; border-radius: var(--radius); cursor: pointer; font-size: 0.9rem; transition: background var(--transition); } .btn-delete:hover { background: #c0392b; } .btn-edit { background: #3498db; color: white; border: none; padding: 0.5rem 1rem; border-radius: var(--radius); cursor: pointer; font-size: 0.9rem; margin-right: 0.5rem; transition: background var(--transition); } .btn-edit:hover { background: #2980b9; } .role-badge { padding: 0.25rem 0.75rem; border-radius: 20px; font-size: 0.8rem; font-weight: bold; } .role-admin { background: #e8f5e8; color: #2e7d2e; } .role-manager { background: #e3f2fd; color: #1976d2; } .stats-section { display: grid; grid-template-columns: repeat(auto-fit, minmax(200px, 1fr)); gap: 1rem; margin-bottom: 2rem; } .stat-card { background: var(--white); border-radius: var(--radius); padding: 1.5rem; box-shadow: 0 2px 8px rgba(0,0,0,0.05); text-align: center; } .stat-number { font-size: 2rem; font-weight: bold; color: var(--orange); font-family: 'Montserrat', sans-serif; } .stat-label { color: var(--dark); margin-top: 0.5rem; font-size: 0.9rem; } .search-section { display: flex; gap: 1rem; margin-bottom: 1.5rem; flex-wrap: wrap; } .search-input { flex: 1; min-width: 200px; } .filter-select { min-width: 150px; } .actions-column { min-width: 150px; } .two-column { display: grid; grid-template-columns: 1fr 2fr; gap: 2rem; } @media (max-width: 768px) { .two-column { grid-template-columns: 1fr; } .form-grid { grid-template-columns: 1fr; } .search-section { flex-direction: column; } } .success-message { background: #e8f5e8; color: #2e7d2e; padding: 0.75rem; border-radius: var(--radius); margin-bottom: 1rem; text-align: center; } .error-message { background: #ffe6e6; color: #e74c3c; padding: 0.75rem; border-radius: var(--radius); margin-bottom: 1rem; text-align: center; }