Documentation Technique Complète

Cyber Solution Platform - Conformité NIS2/RGPD

Table des Matières

1. Vue d'Ensemble

1.1 Présentation du Projet

Cyber Solution est une plateforme B2B SaaS de conformité cybersécurité offrant :

- **Audit de Conformité NIS2** : Évaluation automatisée avec scoring et recommandations
- **Formations Certifiantes** : Catalogue de formations payantes en cybersécurité
- **Assistant IA** : Chatbot spécialisé en NIS2/RGPD
- **Dashboard Temps Réel** : Suivi de la conformité de l'entreprise

1.2 Public Cible

- **DSI (Directeurs des Systèmes d'Information)**
- **RSSI (Responsables de la Sécurité des Systèmes d'Information)**
- **DPO (Data Protection Officers)**
- **Consultants en Cybersécurité**
- **PME/ETI des secteurs critiques** (Santé, Énergie, Finance, Transport)

1.3 Valeur Ajoutée

1. **Simplification** : Audit en 10 minutes vs consultants (3 mois)
2. **Économie** : 0€ pour l'audit vs 15k€ cabinet conseil
3. **Formation** : 199-599€ vs 2000€+ formations traditionnelles
4. **Autonomie** : Outils self-service disponibles 24/7
5. **Conformité** : Respect RGPD Article 8 & NIS2 2024

1.4 Technologies Utilisées

Frontend

- **React 19** : Framework UI moderne
- **TypeScript 5.x** : Typage statique
- **Vite 5.x** : Build tool ultra-rapide
- **Tailwind CSS 3.x** : Utility-first CSS
- **Lucide React** : Icônes modernes

Backend

- **Node.js 20 LTS** : Runtime JavaScript
- **Express 4.x** : Framework web minimaliste
- **SQLite 3** : Base de données embarquée
- **bcryptjs** : Hash de mots de passe
- **jsonwebtoken** : Authentification JWT
- **PDFKit** : Génération de certificats PDF

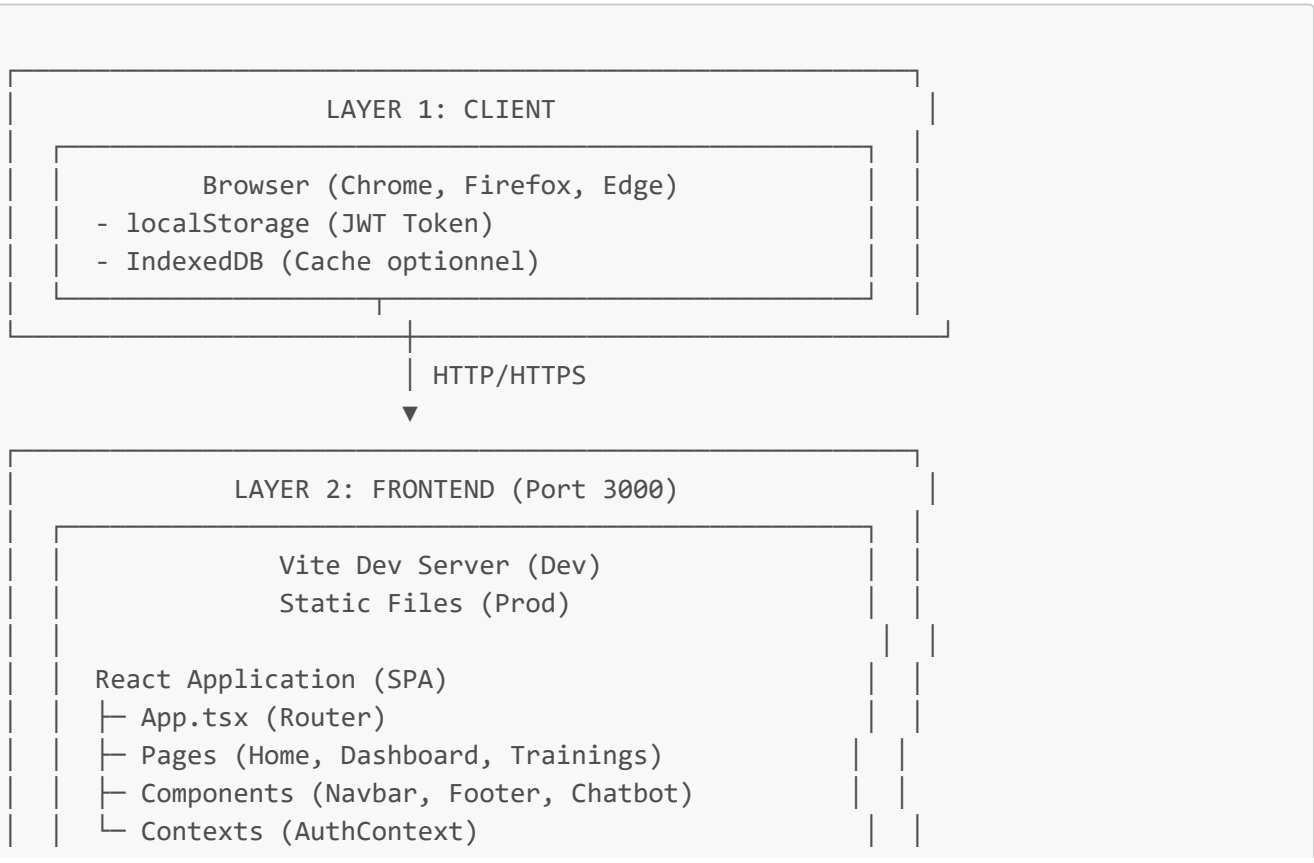
DevOps

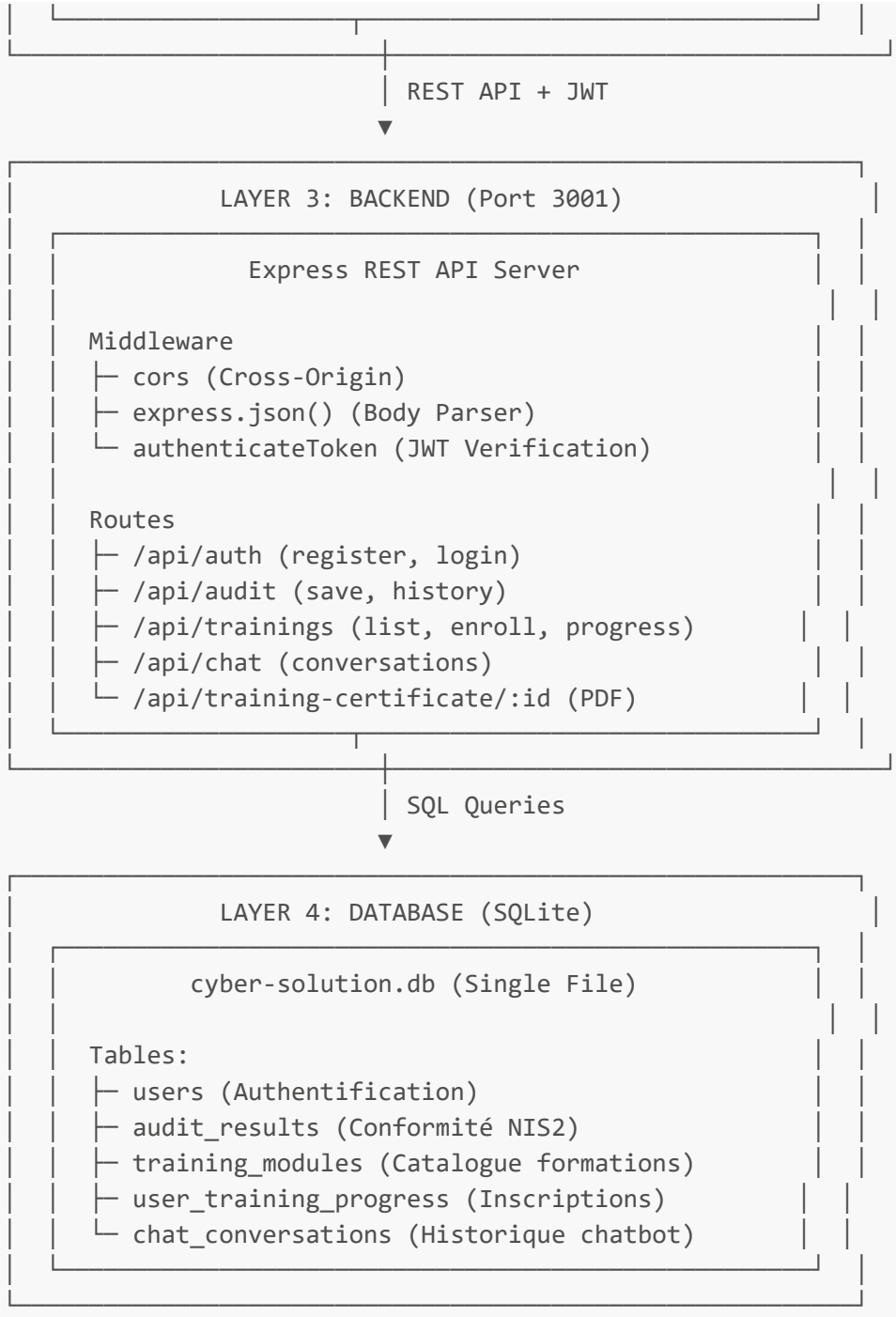
- **Git** : Contrôle de version
- **npm** : Gestionnaire de packages
- **tsx** : Exécuteur TypeScript
- **PowerShell** : Script de lancement

---

2. Architecture Système

2.1 Architecture Globale





2.2 Flux de Données Principaux

Flux 1: Authentification

```
User → Frontend → Backend → Database
     ←          ←          ←

1. User saisit email + password
2. Frontend: POST /api/auth/login { email, password }
3. Backend: SELECT user WHERE email = ?
4. Backend: bcrypt.compare(password, hash)
5. Backend: jwt.sign({ userId, email }, SECRET)
```

```
6. Frontend: localStorage.setItem('token', jwt)
7. Frontend: Redirige vers Dashboard
```

Flux 2: Réalisation d'un Audit

```
User → Frontend → Backend → Database
     ←          ←          ←

1. User sélectionne secteur (Santé)
2. User sélectionne taille (Grande)
3. User coche 12/16 mesures de sécurité
4. Frontend: Calcule score = 12/16 = 75%
5. Frontend: POST /api/audit/save + JWT
6. Backend: Vérifie JWT (req.user.id)
7. Backend: INSERT INTO audit_results (...)
8. Backend: Retourne 201 Created
9. Frontend: dispatchEvent('auditCompleted')
10. Dashboard: Écoute event et se met à jour
```

Flux 3: Inscription Formation avec Paiement

```
User → PaymentModal → Frontend → Backend → Database
     ←              ←          ←          ←

1. User clique "S'inscrire - 299€"
2. Frontend: Affiche PaymentModal
3. User remplit formulaire CB (fictif)
4. Frontend: Valide format (Luhn algo)
5. Frontend: POST /api/trainings/:id/enroll + JWT
6. Backend: INSERT INTO user_training_progress
7. Backend: Retourne 201 Created
8. Frontend: POST /api/trainings/:id/confirm-payment
9. Backend: UPDATE payment_status = 'paid'
10. Frontend: Affiche "Accéder au contenu"
11. User: Accède modules, leçons, quiz
```

2.3 Composants Frontend

```
App.tsx (Router Principal)
│
├─ Navbar.tsx (Navigation Globale)
│  ├─ Logo + Links
│  └─ Auth State (Connecté/Déconnecté)
│
├─ Pages
│  ├─ Home.tsx
```

```
│  │  ├─ Hero.tsx (Section Hero)
│  │  ├─ Services.tsx (Liste services)
│  │  ├─ RegulatedSectors.tsx (Secteurs NIS2)
│  │  └─ NotarySolutions.tsx (Solutions métiers)
│  │
│  ├─ Dashboard.tsx (Stats Utilisateur)
│  │  └─ Fetch /api/audit/history
│  │
│  ├─ Trainings.tsx (Catalogue Formations)
│  │  ├─ Fetch /api/trainings
│  │  ├─ PaymentModal.tsx (Paiement CB)
│  │  └─ TrainingContent.tsx (Viewer Contenu)
│  │     ├─ Sidebar (Modules/Leçons)
│  │     ├─ Content Area (Texte leçon)
│  │     └─ Quiz.tsx (Questions)
│  │
│  ├─ Nis2Calculator.tsx (Wizard Audit)
│  │  ├─ Step: Start
│  │  ├─ Step: Sector Selection
│  │  ├─ Step: Size Selection
│  │  ├─ Step: Readiness Checklist (16 items)
│  │  └─ Step: Results (Score + Recommandations)
│  │
│  ├─ ServiceDetail.tsx (Détail Service)
│  ├─ Contact.tsx (Formulaire Contact)
│  └─ CGU.tsx (Conditions Générales)
│
├─ Components
│  ├─ Chatbot.tsx (Assistant IA)
│  │  ├─ POST /api/chat
│  │  └─ Fetch /api/chat/history
│  │
│  ├─ ContactForm.tsx (Newsletter)
│  └─ Footer.tsx (Liens Légaux)
│
└─ Contexts
   └─ AuthContext.tsx
      ├─ State: user, token, isAuthenticated
      ├─ login(email, password)
      ├─ register(email, password, company)
      └─ logout()
```

2.4 Routes Backend

```
// Routes Publiques (Sans JWT)
POST   /api/auth/register     → Créer compte
POST   /api/auth/login         → Se connecter

// Routes Protégées (Avec JWT)
GET    /api/audit/history      → Historique audits user
POST   /api/audit/save         → Sauvegarder audit
```

```
GET    /api/trainings          → Liste formations
GET    /api/trainings/user/progress → Formations inscrites
POST   /api/trainings/:id/enroll   → Inscription formation
POST   /api/trainings/:id/confirm-payment → Confirmer paiement
GET    /api/training-certificate/:id → Télécharger certificat PDF

POST   /api/chat               → Envoyer message chatbot
GET    /api/chat/history       → Historique conversations
```

---

3. Base de Données

3.1 Schéma Complet

Table: users

```
CREATE TABLE users (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  email TEXT UNIQUE NOT NULL,
  password_hash TEXT NOT NULL,
  company_name TEXT NOT NULL,
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);

CREATE UNIQUE INDEX idx_users_email ON users(email);
```

Exemple de données:

```
INSERT INTO users (email, password_hash, company_name) VALUES
('jean.dupont@sante-paris.fr', '$2a$10$ABC...XYZ', 'Hôpital Saint-Antoine'),
('marie.martin@energy.com', '$2a$10$DEF...UVW', 'EDF Énergies Nouvelles');
```

Table: audit_results

```
CREATE TABLE audit_results (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  user_id INTEGER NOT NULL,
  sector_name TEXT NOT NULL,
  sector_type TEXT NOT NULL CHECK(sector_type IN ('critical', 'important',
'excluded')),
  size TEXT NOT NULL CHECK(size IN ('Micro', 'PME', 'Grande')),
  readiness_score INTEGER NOT NULL CHECK(readiness_score >= 0 AND readiness_score
<= 100),
  missing_items TEXT,
```

```
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

CREATE INDEX idx_audit_user_date ON audit_results(user_id, created_at DESC);
```

Exemple de données:

```
INSERT INTO audit_results (user_id, sector_name, sector_type, size,
readiness_score, missing_items) VALUES
(1, 'Santé', 'critical', 'Grande', 75, '["MFA non implémenté", "Pas de SOC",
"Chiffrement partiel", "Formations manquantes"]'),
(1, 'Santé', 'critical', 'Grande', 88, '["Exercices de crise à planifier",
"Documentation PSSI à jour"]'),
(2, 'Énergie', 'critical', 'PME', 62, '["EDR absent", "Backups non testés", "Plan
de reprise inexistant"]');
```

Calcul du Score:

```
const readinessScore = Math.round((checkedItems.length / 16) * 100);

// Interprétation
if (readinessScore >= 80) {
  level = 'Excellent';
  risk = 'Faible';
  status = '☑ CONFORME';
} else if (readinessScore >= 60) {
  level = 'Moyen';
  risk = 'Élevé';
  status = '⚠ NON CONFORME';
} else {
  level = 'Critique';
  risk = 'Maximal';
  status = '🚨 DANGER CRITIQUE';
}
```

Table: training_modules

```
CREATE TABLE training_modules (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  title TEXT UNIQUE NOT NULL,
  description TEXT NOT NULL,
  category TEXT NOT NULL,
  duration_hours INTEGER NOT NULL,
  level TEXT NOT NULL CHECK(level IN ('Débutant', 'Intermédiaire', 'Avancé')),
  price REAL NOT NULL CHECK(price >= 0),
```

```
  content TEXT NOT NULL,
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);

CREATE UNIQUE INDEX idx_training_title ON training_modules(title);
CREATE INDEX idx_training_category ON training_modules(category);
```

Exemple de données:

```
INSERT INTO training_modules (title, description, category, duration_hours, level,
price, content) VALUES
(
  'Introduction à NIS2',
  'Formation complète sur la directive NIS2 et ses implications',
  'Conformité',
  4,
  'Débutant',
  299.00,
  '{
    "modules": [
      {
        "title": "Comprendre NIS2",
        "duration": "15 min",
        "lessons": [
          {
            "title": "Historique et Contexte",
            "content": "La directive NIS2 (Network and Information Security) est
entrée en vigueur en janvier 2023..."
          },
          {
            "title": "Périmètre d\'Application",
            "content": "NIS2 s\'applique à 18 secteurs critiques et importants..."
          }
        ]
      },
      {
        "title": "Obligations Clés",
        "duration": "20 min",
        "lessons": [...]
      }
    ],
    "quiz": [
      {
        "question": "Quelle est la date d\'entrée en vigueur de NIS2 ?",
        "options": ["Janvier 2021", "Janvier 2023", "Octobre 2024", "Janvier
2025"],
        "correct": 2
      },
      {
        "question": "Combien de secteurs sont concernés par NIS2 ?",
        "options": ["12", "15", "18", "24"],
        "correct": 2
```

```
      }
    ]
  }'
);
```

Structure JSON du Content:

```
interface TrainingContent {
  modules: Module[];
  quiz: QuizQuestion[];
}

interface Module {
  title: string;
  duration: string; // "15 min"
  lessons: Lesson[];
}

interface Lesson {
  title: string;
  content: string; // Markdown ou HTML
}

interface QuizQuestion {
  question: string;
  options: string[]; // 4 options (A, B, C, D)
  correct: number;   // Index de la réponse correcte (0-3)
}
```

Table: user_training_progress

```
CREATE TABLE user_training_progress (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  user_id INTEGER NOT NULL,
  training_id INTEGER NOT NULL,
  progress INTEGER DEFAULT 0 CHECK(progress >= 0 AND progress <= 100),
  completed BOOLEAN DEFAULT 0,
  payment_status TEXT DEFAULT 'pending' CHECK(payment_status IN ('pending',
'paid', 'failed', 'refunded')),
  enrolled_at DATETIME DEFAULT CURRENT_TIMESTAMP,
  started_at DATETIME,
  completed_at DATETIME,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
  FOREIGN KEY (training_id) REFERENCES training_modules(id) ON DELETE CASCADE,
  UNIQUE(user_id, training_id)
);

CREATE UNIQUE INDEX idx_user_training ON user_training_progress(user_id,
```

```
training_id);
CREATE INDEX idx_progress_status ON user_training_progress(payment_status);
```

Exemple de données:

```
INSERT INTO user_training_progress (user_id, training_id, progress, completed,
payment_status, started_at, completed_at) VALUES
(1, 1, 100, 1, 'paid', '2026-01-15 09:00:00', '2026-01-20 14:30:00'), -- Formation
terminée
(1, 2, 45, 0, 'paid', '2026-01-22 10:00:00', NULL),                    -- En cours
(2, 3, 0, 0, 'pending', NULL, NULL);                                   -- Inscrit
mais pas payé
```

Workflow Paiement:

```
1. User clique "S'inscrire"
   → INSERT (payment_status='pending', started_at=NULL)

2. User valide paiement CB
   → POST /api/trainings/:id/confirm-payment
   → UPDATE payment_status='paid'

3. User commence la formation
   → UPDATE started_at=NOW()

4. User termine quiz avec 80%+
   → UPDATE progress=100, completed=1, completed_at=NOW()

5. User télécharge certificat
   → GET /api/training-certificate/:id
   → Génère PDF avec PDFKit
```

Table: chat_conversations

```
CREATE TABLE chat_conversations (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  user_id INTEGER NOT NULL,
  message TEXT NOT NULL,
  response TEXT NOT NULL,
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

CREATE INDEX idx_chat_user_date ON chat_conversations(user_id, created_at DESC);
```

Exemple de données:

```
INSERT INTO chat_conversations (user_id, message, response) VALUES
(
  1,
  'Quelles sont les obligations NIS2 pour le secteur santé ?',
  'Le secteur santé est classé comme secteur **critique** dans la directive
NIS2...'
),
(
  1,
  'Comment implémenter la MFA ?',
  'L\'authentification multi-facteurs (MFA) peut être mise en place avec...'
);
```

3.2 Requêtes SQL Fréquentes

Authentification

```
-- Vérifier si email existe (register)
SELECT id FROM users WHERE email = ?;

-- Login utilisateur
SELECT id, email, password_hash, company_name FROM users WHERE email = ?;

-- Créer compte
INSERT INTO users (email, password_hash, company_name) VALUES (?, ?, ?);
```

Audit

```
-- Sauvegarder audit
INSERT INTO audit_results (user_id, sector_name, sector_type, size,
readiness_score, missing_items)
VALUES (?, ?, ?, ?, ?, ?);

-- Historique audits d'un utilisateur
SELECT id, sector_name, sector_type, size, readiness_score, created_at
FROM audit_results
WHERE user_id = ?
ORDER BY created_at DESC
LIMIT 10;

-- Statistiques utilisateur (Dashboard)
SELECT
  COUNT(*) as total_audits,
  AVG(readiness_score) as avg_score,
```

```
  MAX(readiness_score) as best_score
FROM audit_results
WHERE user_id = ?;
```

Formations

```
-- Liste toutes formations
SELECT id, title, description, category, duration_hours, level, price
FROM training_modules
ORDER BY price ASC;

-- Formations inscrites par utilisateur
SELECT
  tm.id, tm.title, tm.price,
  utp.progress, utp.completed, utp.payment_status
FROM user_training_progress utp
JOIN training_modules tm ON utp.training_id = tm.id
WHERE utp.user_id = ?
ORDER BY utp.enrolled_at DESC;

-- Inscription formation
INSERT INTO user_training_progress (user_id, training_id, payment_status)
VALUES (?, ?, 'pending');

-- Confirmer paiement
UPDATE user_training_progress
SET payment_status = 'paid'
WHERE user_id = ? AND training_id = ?;

-- Marquer formation commencée
UPDATE user_training_progress
SET started_at = CURRENT_TIMESTAMP
WHERE user_id = ? AND training_id = ? AND started_at IS NULL;
```

Chatbot

```
-- Sauvegarder conversation
INSERT INTO chat_conversations (user_id, message, response)
VALUES (?, ?, ?);

-- Historique conversations
SELECT message, response, created_at
FROM chat_conversations
WHERE user_id = ?
ORDER BY created_at DESC
LIMIT 20;
```

3.3 Migrations & Seed Data

Script d'Initialisation (database.ts)

```
import Database from 'better-sqlite3';

const db = new Database('cyber-solution.db');

// 1. Créer tables
db.exec(`
  CREATE TABLE IF NOT EXISTS users (...);
  CREATE TABLE IF NOT EXISTS audit_results (...);
  CREATE TABLE IF NOT EXISTS training_modules (...);
  CREATE TABLE IF NOT EXISTS user_training_progress (...);
  CREATE TABLE IF NOT EXISTS chat_conversations (...);
`);

// 2. Créer index
db.exec(`
  CREATE UNIQUE INDEX IF NOT EXISTS idx_users_email ON users(email);
  CREATE INDEX IF NOT EXISTS idx_audit_user_date ON audit_results(user_id,
created_at DESC);
  -- ...
`);

// 3. Seed formations si table vide
const count = db.prepare('SELECT COUNT(*) as count FROM training_modules').get();
if (count.count === 0) {
  const insertTraining = db.prepare(`
    INSERT INTO training_modules (title, description, category, duration_hours,
level, price, content)
    VALUES (?, ?, ?, ?, ?, ?, ?)
  `);

  insertTraining.run('Introduction à NIS2', '...', 'Conformité', 4, 'Débutant',
299.00, JSON.stringify(nis2Content));
  insertTraining.run('RGPD: Les Essentiels', '...', 'Protection des données', 4,
'Intermédiaire', 349.00, JSON.stringify(rgpdContent));
  // ...
}
```

---

4. API Backend

4.1 Serveur Express (index.ts)

```
import express from 'express';
import cors from 'cors';
import jwt from 'jsonwebtoken';
import bcrypt from 'bcryptjs';
```

```
import db from './database';

const app = express();
const PORT = 3001;
const JWT_SECRET = process.env.JWT_SECRET || 'dev-secret-key-change-in-
production';

// Middleware
app.use(cors({ origin: 'http://localhost:3000', credentials: true }));
app.use(express.json());

// Middleware d'authentification
interface AuthRequest extends express.Request {
  user?: { userId: number; email: string };
}

function authenticateToken(req: AuthRequest, res: express.Response, next:
express.NextFunction) {
  const authHeader = req.headers['authorization'];
  const token = authHeader && authHeader.split(' ')[1]; // "Bearer TOKEN"

  if (!token) return res.sendStatus(401);

  jwt.verify(token, JWT_SECRET, (err: any, user: any) => {
    if (err) return res.sendStatus(403);
    req.user = user;
    next();
  });
}

// Routes publiques
app.post('/api/auth/register', async (req, res) => { /* ... */ });
app.post('/api/auth/login', async (req, res) => { /* ... */ });

// Routes protégées
app.get('/api/audit/history', authenticateToken, (req: AuthRequest, res) => { /*
... */ });
app.post('/api/audit/save', authenticateToken, (req: AuthRequest, res) => { /* ...
*/ });
app.get('/api/trainings', authenticateToken, (req: AuthRequest, res) => { /* ...
*/ });
// ...

app.listen(PORT, () => console.log(`Server running on http://localhost:${PORT}`));
```

4.2 Routes Détaillées

POST /api/auth/register

Description: Créer un nouveau compte utilisateur

Body:

```
{
  "email": "jean.dupont@company.com",
  "password": "SecurePassword123",
  "company_name": "Cyber Solutions SARL"
}
```

Réponse Success (201):

```
{
  "message": "Utilisateur créé avec succès",
  "user": {
    "id": 42,
    "email": "jean.dupont@company.com",
    "company_name": "Cyber Solutions SARL"
  },
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

Réponse Error (400):

```
{
  "error": "Cet email est déjà utilisé"
}
```

Code:

```
app.post('/api/auth/register', async (req, res) => {
  try {
    const { email, password, company_name } = req.body;

    // Validation
    if (!email || !password || !company_name) {
      return res.status(400).json({ error: 'Tous les champs sont requis' });
    }

    // Vérifier email unique
    const existing = db.prepare('SELECT id FROM users WHERE email =
?').get(email);
    if (existing) {
      return res.status(400).json({ error: 'Cet email est déjà utilisé' });
    }

    // Hash password
    const password_hash = await bcrypt.hash(password, 10);
```

```
    // Insérer user
    const result = db.prepare(`
      INSERT INTO users (email, password_hash, company_name)
      VALUES (?, ?, ?)
    `).run(email, password_hash, company_name);

    const userId = result.lastInsertRowid as number;

    // Générer JWT
    const token = jwt.sign({ userId, email }, JWT_SECRET, { expiresIn: '7d' });

    res.status(201).json({
      message: 'Utilisateur créé avec succès',
      user: { id: userId, email, company_name },
      token
    });
  } catch (error) {
    console.error('Register error:', error);
    res.status(500).json({ error: 'Erreur serveur' });
  }
});
```

POST /api/auth/login

Description: Connexion utilisateur

Body:

```
{
  "email": "jean.dupont@company.com",
  "password": "SecurePassword123"
}
```

Réponse Success (200):

```
{
  "message": "Connexion réussie",
  "user": {
    "id": 42,
    "email": "jean.dupont@company.com",
    "company_name": "Cyber Solutions SARL"
  },
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

Réponse Error (401):

```
{
  "error": "Email ou mot de passe incorrect"
}
```

Code:

```
app.post('/api/auth/login', async (req, res) => {
  try {
    const { email, password } = req.body;

    // Récupérer user
    const user = db.prepare('SELECT * FROM users WHERE email = ?').get(email) as
any;

    if (!user) {
      return res.status(401).json({ error: 'Email ou mot de passe incorrect' });
    }

    // Vérifier password
    const isValid = await bcrypt.compare(password, user.password_hash);

    if (!isValid) {
      return res.status(401).json({ error: 'Email ou mot de passe incorrect' });
    }

    // Générer JWT
    const token = jwt.sign({ userId: user.id, email: user.email }, JWT_SECRET, {
expiresIn: '7d' });

    res.json({
      message: 'Connexion réussie',
      user: {
        id: user.id,
        email: user.email,
        company_name: user.company_name
      },
      token
    });
  } catch (error) {
    console.error('Login error:', error);
    res.status(500).json({ error: 'Erreur serveur' });
  }
});
```

POST /api/audit/save

Description: Sauvegarder résultat d'audit (protégé JWT)

Headers:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Body:

```
{
  "sector_name": "Santé",
  "sector_type": "critical",
  "size": "Grande",
  "readiness_score": 75,
  "missing_items": ["MFA non implémenté", "Pas de SOC"]
}
```

Réponse Success (201):

```
{
  "message": "Audit sauvegardé",
  "audit": {
    "id": 123,
    "user_id": 42,
    "sector_name": "Santé",
    "readiness_score": 75,
    "created_at": "2026-01-28T10:30:00Z"
  }
}
```

Code:

```
app.post('/api/audit/save', authenticateToken, (req: AuthRequest, res) => {
  try {
    const { sector_name, sector_type, size, readiness_score, missing_items } =
req.body;
    const user_id = req.user!.userId;

    const result = db.prepare(`
      INSERT INTO audit_results (user_id, sector_name, sector_type, size,
readiness_score, missing_items)
      VALUES (?, ?, ?, ?, ?, ?)
    `).run(user_id, sector_name, sector_type, size, readiness_score,
JSON.stringify(missing_items));

    res.status(201).json({
      message: 'Audit sauvegardé',
      audit: {
        id: result.lastInsertRowid,
```

```
        user_id,
        sector_name,
        readiness_score,
        created_at: new Date().toISOString()
      }
    });
  } catch (error) {
    console.error('Save audit error:', error);
    res.status(500).json({ error: 'Erreur serveur' });
  }
});
```

GET /api/trainings

Description: Liste toutes les formations (protégé JWT)

Réponse Success (200):

```
{
  "trainings": [
    {
      "id": 1,
      "title": "Introduction à NIS2",
      "description": "Formation complète sur la directive NIS2",
      "category": "Conformité",
      "duration_hours": 4,
      "level": "Débutant",
      "price": 299.00,
      "content": "{ modules: [...], quiz: [...] }"
    },
    // ...
  ]
}
```

Code:

```
app.get('/api/trainings', authenticateToken, (req: AuthRequest, res) => {
  try {
    const trainings = db.prepare('SELECT * FROM training_modules').all();
    res.json({ trainings });
  } catch (error) {
    console.error('Get trainings error:', error);
    res.status(500).json({ error: 'Erreur serveur' });
  }
});
```

POST /api/trainings/:id/enroll

Description: Inscription à une formation (protégé JWT)

Réponse Success (201):

```
{
  "message": "Inscription réussie",
  "enrollment": {
    "id": 456,
    "user_id": 42,
    "training_id": 1,
    "payment_status": "pending"
  }
}
```

Code:

```
app.post('/api/trainings/:id/enroll', authenticateToken, (req: AuthRequest, res)
=> {
  try {
    const training_id = parseInt(req.params.id);
    const user_id = req.user!.userId;

    // Vérifier si déjà inscrit
    const existing = db.prepare(`
      SELECT id FROM user_training_progress
      WHERE user_id = ? AND training_id = ?
    `).get(user_id, training_id);

    if (existing) {
      return res.status(400).json({ error: 'Déjà inscrit à cette formation' });
    }

    // Insérer inscription
    const result = db.prepare(`
      INSERT INTO user_training_progress (user_id, training_id, payment_status)
      VALUES (?, ?, 'pending')
    `).run(user_id, training_id);

    res.status(201).json({
      message: 'Inscription réussie',
      enrollment: {
        id: result.lastInsertRowid,
        user_id,
        training_id,
        payment_status: 'pending'
      }
    });
  } catch (error) {
```

```
    console.error('Enroll error:', error);
    res.status(500).json({ error: 'Erreur serveur' });
  }
});
```

POST /api/trainings/:id/confirm-payment

Description: Confirmer paiement formation (protégé JWT)

Réponse Success (200):

```
{
  "message": "Paiement confirmé"
}
```

Code:

```
app.post('/api/trainings/:id/confirm-payment', authenticateToken, (req:
AuthRequest, res) => {
  try {
    const training_id = parseInt(req.params.id);
    const user_id = req.user!.userId;

    db.prepare(`
      UPDATE user_training_progress
      SET payment_status = 'paid'
      WHERE user_id = ? AND training_id = ?
    `).run(user_id, training_id);

    res.json({ message: 'Paiement confirmé' });
  } catch (error) {
    console.error('Confirm payment error:', error);
    res.status(500).json({ error: 'Erreur serveur' });
  }
});
```

GET /api/training-certificate/:trainingId

Description: Télécharger certificat PDF (protégé JWT)

Réponse: Fichier PDF en stream

Code:

```
import PDFDocument from 'pdfkit';

app.get('/api/training-certificate/:trainingId', authenticateToken, (req:
AuthRequest, res) => {
  try {
    const trainingId = parseInt(req.params.trainingId);
    const userId = req.user!.userId;

    // Vérifier paiement et complétion
    const progress = db.prepare(`
      SELECT * FROM user_training_progress
      WHERE user_id = ? AND training_id = ? AND payment_status = 'paid' AND
completed = 1
    `).get(userId, trainingId) as any;

    if (!progress) {
      return res.status(404).json({ error: 'Formation non terminée ou non payée'
});
    }

    // Récupérer infos training + user
    const training = db.prepare('SELECT * FROM training_modules WHERE id =
?').get(trainingId) as any;
    const user = db.prepare('SELECT * FROM users WHERE id = ?').get(userId) as
any;

    // Générer PDF
    const doc = new PDFDocument({ size: 'A4', margin: 50 });

    res.setHeader('Content-Type', 'application/pdf');
    res.setHeader('Content-Disposition', `attachment;
filename=certificat-${training.title}.pdf`);

    doc.pipe(res);

    // Contenu PDF
    doc.fontSize(25).text('CERTIFICAT DE FORMATION', { align: 'center' });
    doc.moveDown();
    doc.fontSize(16).text(`Formation: ${training.title}`, { align: 'center' });
    doc.moveDown();
    doc.fontSize(12).text(`Entreprise: ${user.company_name}`, { align: 'center'
});
    doc.text(`Date de complétion: ${new
Date(progress.completed_at).toLocaleDateString('fr-FR')}`, { align: 'center' });
    doc.moveDown();
    doc.fontSize(10).text(`Durée: ${training.duration_hours} heures`, { align:
'center' });

    doc.end();
  } catch (error) {
    console.error('Certificate error:', error);
    res.status(500).json({ error: 'Erreur serveur' });
```

```
  }
});
```

4.3 Gestion des Erreurs

```
// Middleware global error handler
app.use((err: Error, req: express.Request, res: express.Response, next:
express.NextFunction) => {
  console.error('Unhandled error:', err);
  res.status(500).json({ error: 'Erreur serveur interne' });
});

// 404 Not Found
app.use((req, res) => {
  res.status(404).json({ error: 'Route non trouvée' });
});
```

---

5. Frontend

5.1 Structure des Composants

```
src/
├─ App.tsx                    → Router principal
├─ index.tsx                  → Entry point
├─ index.css                  → Styles globaux Tailwind
│
├─ components/
│  ├─ Navbar.tsx              → Navigation globale
│  ├─ Footer.tsx              → Pied de page + liens légaux
│  ├─ Hero.tsx                → Section hero homepage
│  ├─ Home.tsx                → Page d'accueil complète
│  ├─ Services.tsx            → Liste services offerts
│  ├─ ServiceDetail.tsx       → Détail d'un service
│  ├─ RegulatedSectors.tsx    → Secteurs NIS2
│  ├─ NotarySolutions.tsx     → Solutions métiers
│  ├─ ContactForm.tsx         → Formulaire newsletter
│  ├─ Dashboard.tsx           → Tableau de bord utilisateur
│  ├─ Trainings.tsx           → Catalogue formations
│  ├─ TrainingContent.tsx     → Viewer contenu formation
│  ├─ PaymentModal.tsx        → Modal paiement CB
│  ├─ Nis2Calculator.tsx      → Wizard audit NIS2
│  ├─ Chatbot.tsx             → Assistant IA
│  └─ CGU.tsx                 → Conditions générales
│
└─ contexts/
   └─ AuthContext.tsx         → Contexte authentification
```

5.2 AuthContext.tsx

Rôle: Gestion globale de l'authentification (JWT, user state)

```
import React, { createContext, useContext, useState, useEffect } from 'react';

interface User {
  id: number;
  email: string;
  company_name: string;
}

interface AuthContextType {
  user: User | null;
  token: string | null;
  isAuthenticated: boolean;
  login: (email: string, password: string) => Promise<void>;
  register: (email: string, password: string, company_name: string) =>
Promise<void>;
  logout: () => void;
}

const AuthContext = createContext<AuthContextType | undefined>(undefined);

export const AuthProvider: React.FC<{ children: React.ReactNode }> = ({ children
}) => {
  const [user, setUser] = useState<User | null>(null);
  const [token, setToken] = useState<string | null>(null);

  useEffect(() => {
    // Récupérer token au démarrage
    const savedToken = localStorage.getItem('token');
    if (savedToken) {
      setToken(savedToken);
      // Optionnel: Décoder JWT pour récupérer user
      // const decoded = jwt_decode(savedToken);
      // setUser(decoded);
    }
  }, []);

  const login = async (email: string, password: string) => {
    const response = await fetch('http://localhost:3001/api/auth/login', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ email, password })
    });

    if (!response.ok) throw new Error('Login failed');

    const data = await response.json();
    setUser(data.user);
```

```
    setToken(data.token);
    localStorage.setItem('token', data.token);
  };

  const register = async (email: string, password: string, company_name: string)
=> {
    const response = await fetch('http://localhost:3001/api/auth/register', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ email, password, company_name })
    });

    if (!response.ok) {
      const error = await response.json();
      throw new Error(error.error || 'Registration failed');
    }

    const data = await response.json();
    setUser(data.user);
    setToken(data.token);
    localStorage.setItem('token', data.token);
  };

  const logout = () => {
    setUser(null);
    setToken(null);
    localStorage.removeItem('token');
  };

  return (
    <AuthContext.Provider value={{
      user,
      token,
      isAuthenticated: !!token,
      login,
      register,
      logout
    }}>
      {children}
    </AuthContext.Provider>
  );
};

export const useAuth = () => {
  const context = useContext(AuthContext);
  if (!context) throw new Error('useAuth must be used within AuthProvider');
  return context;
};
```

5.3 Dashboard.tsx

Rôle: Afficher statistiques de conformité utilisateur

```
import React, { useEffect, useState } from 'react';
import { useAuth } from '../contexts/AuthContext';

interface DashboardStats {
  totalAudits: number;
  avgScore: number;
  bestScore: number;
  trainingsCount: number;
}

export const Dashboard: React.FC = () => {
  const { token } = useAuth();
  const [stats, setStats] = useState<DashboardStats | null>(null);
  const [loading, setLoading] = useState(true);

  const fetchStats = async () => {
    try {
      const response = await fetch('http://localhost:3001/api/audit/history', {
        headers: {
          'Authorization': `Bearer ${token}`
        }
      });

      const data = await response.json();

      // Calculer stats
      const audits = data.audits || [];
      const totalAudits = audits.length;
      const avgScore = audits.reduce((sum: number, audit: any) => sum +
audit.readiness_score, 0) / totalAudits || 0;
      const bestScore = Math.max(...audits.map((a: any) => a.readiness_score), 0);

      setStats({ totalAudits, avgScore, bestScore, trainingsCount: 0 });
      setLoading(false);
    } catch (error) {
      console.error('Fetch stats error:', error);
      setLoading(false);
    }
  };

  useEffect(() => {
    fetchStats();

    // Écouter événement audit terminé
    const handleAuditCompleted = () => fetchStats();
    window.addEventListener('auditCompleted', handleAuditCompleted);

    // Polling toutes les 5 secondes
    const interval = setInterval(fetchStats, 5000);

    return () => {
```

```
      window.removeEventListener('auditCompleted', handleAuditCompleted);
      clearInterval(interval);
    };
  }, [token]);

  if (loading) return <div>Chargement...</div>;

  return (
    <div className="container mx-auto px-4 py-8">
      <h1 className="text-3xl font-bold mb-8">Tableau de Bord</h1>

      <div className="grid grid-cols-1 md:grid-cols-3 gap-6">
        {/* Audits réalisés */}
        <div className="bg-white p-6 rounded-lg shadow">
          <h3 className="text-gray-600 text-sm font-medium">Audits Réalisés</h3>
          <p className="text-4xl font-bold text-blue-600">{stats?.totalAudits ||
0}</p>
        </div>

        {/* Score moyen */}
        <div className="bg-white p-6 rounded-lg shadow">
          <h3 className="text-gray-600 text-sm font-medium">Score Moyen</h3>
          <p className="text-4xl font-bold text-green-600">
{Math.round(stats?.avgScore || 0)}%</p>
        </div>

        {/* Meilleur score */}
        <div className="bg-white p-6 rounded-lg shadow">
          <h3 className="text-gray-600 text-sm font-medium">Meilleur Score</h3>
          <p className="text-4xl font-bold text-purple-600">{stats?.bestScore ||
0}%</p>
        </div>
      </div>
    </div>
  );
};
```

5.4 PaymentModal.tsx

Rôle: Formulaire paiement carte bancaire (fictif)

```
import React, { useState } from 'react';
import { CreditCard, X } from 'lucide-react';

export interface PaymentData {
  cardNumber: string;
  expiryDate: string;
  cvv: string;
  cardHolder: string;
}
```

```tsx
interface PaymentModalProps {
  isOpen: boolean;
  onClose: () => void;
  onConfirm: (data: PaymentData) => void;
  amount: number;
  trainingTitle: string;
}

export const PaymentModal: React.FC<PaymentModalProps> = ({
  isOpen,
  onClose,
  onConfirm,
  amount,
  trainingTitle
}) => {
  const [formData, setFormData] = useState<PaymentData>({
    cardNumber: '',
    expiryDate: '',
    cvv: '',
    cardHolder: ''
  });

  const [errors, setErrors] = useState<Partial<PaymentData>>({});
  const [processing, setProcessing] = useState(false);

  const formatCardNumber = (value: string) => {
    const cleaned = value.replace(/\D/g, '');
    const formatted = cleaned.match(/.{1,4}/g)?.join(' ') || cleaned;
    return formatted.slice(0, 19); // 16 digits + 3 spaces
  };

  const formatExpiryDate = (value: string) => {
    const cleaned = value.replace(/\D/g, '');
    if (cleaned.length >= 2) {
      return `${cleaned.slice(0, 2)}/${cleaned.slice(2, 4)}`;
    }
    return cleaned;
  };

  const validate = (): boolean => {
    const newErrors: Partial<PaymentData> = {};

    // Numéro carte (16 chiffres)
    const cardDigits = formData.cardNumber.replace(/\s/g, '');
    if (cardDigits.length !== 16) {
      newErrors.cardNumber = 'Numéro de carte invalide (16 chiffres requis)';
    }

    // Date expiration (MM/AA)
    const [month, year] = formData.expiryDate.split('/');
    if (!month || !year || parseInt(month) < 1 || parseInt(month) > 12) {
      newErrors.expiryDate = 'Date invalide (MM/AA)';
    }
```

```
    // CVV (3 chiffres)
    if (formData.cvv.length !== 3) {
      newErrors.cvv = 'CVV invalide (3 chiffres)';
    }

    // Titulaire
    if (formData.cardHolder.trim().length < 3) {
      newErrors.cardHolder = 'Nom du titulaire requis';
    }

    setErrors(newErrors);
    return Object.keys(newErrors).length === 0;
  };

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();

    if (!validate()) return;

    setProcessing(true);

    // Simuler traitement paiement (2 secondes)
    setTimeout(() => {
      setProcessing(false);
      onConfirm(formData);
    }, 2000);
  };

  if (!isOpen) return null;

  return (
    <div className="fixed inset-0 bg-black/50 flex items-center justify-center z-
50">
      <div className="bg-white rounded-lg shadow-xl max-w-md w-full mx-4 p-6">
        {/* Header */}
        <div className="flex justify-between items-center mb-6">
          <h2 className="text-2xl font-bold flex items-center gap-2">
            <CreditCard className="w-6 h-6" />
            Paiement Sécurisé
          </h2>
          <button onClick={onClose} className="text-gray-500 hover:text-gray-700">
            <X className="w-6 h-6" />
          </button>
        </div>

        {/* Formation info */}
        <div className="bg-blue-50 p-4 rounded-lg mb-6">
          <p className="font-semibold">{trainingTitle}</p>
          <p className="text-2xl font-bold text-blue-600">{amount}€</p>
        </div>

        {/* Formulaire */}
        <form onSubmit={handleSubmit}>
```

```
          {/* Numéro carte */}
          <div className="mb-4">
            <label className="block text-sm font-medium mb-2">Numéro de
carte</label>
            <input
              type="text"
              value={formData.cardNumber}
              onChange={(e) => setFormData({ ...formData, cardNumber:
formatCardNumber(e.target.value) })}
              placeholder="1234 5678 9012 3456"
              className="w-full px-4 py-2 border rounded-lg"
            />
            {errors.cardNumber && <p className="text-red-500 text-sm mt-1">
{errors.cardNumber}</p>}
          </div>

          {/* Date & CVV */}
          <div className="grid grid-cols-2 gap-4 mb-4">
            <div>
              <label className="block text-sm font-medium mb-2">Date
d'expiration</label>
              <input
                type="text"
                value={formData.expiryDate}
                onChange={(e) => setFormData({ ...formData, expiryDate:
formatExpiryDate(e.target.value) })}
                placeholder="MM/AA"
                className="w-full px-4 py-2 border rounded-lg"
              />
              {errors.expiryDate && <p className="text-red-500 text-sm mt-1">
{errors.expiryDate}</p>}
            </div>

            <div>
              <label className="block text-sm font-medium mb-2">CVV</label>
              <input
                type="text"
                maxLength={3}
                value={formData.cvv}
                onChange={(e) => setFormData({ ...formData, cvv:
e.target.value.replace(/\D/g, '') })}
                placeholder="123"
                className="w-full px-4 py-2 border rounded-lg"
              />
              {errors.cvv && <p className="text-red-500 text-sm mt-1">{errors.cvv}
</p>}
            </div>
          </div>

          {/* Titulaire */}
          <div className="mb-6">
            <label className="block text-sm font-medium mb-2">Titulaire de la
carte</label>
            <input
```

```
              type="text"
              value={formData.cardHolder}
              onChange={(e) => setFormData({ ...formData, cardHolder:
e.target.value.toUpperCase() })}
              placeholder="JEAN DUPONT"
              className="w-full px-4 py-2 border rounded-lg uppercase"
            />
            {errors.cardHolder && <p className="text-red-500 text-sm mt-1">
{errors.cardHolder}</p>}
          </div>

          {/* Boutons */}
          <div className="flex gap-4">
            <button
              type="button"
              onClick={onClose}
              className="flex-1 px-6 py-3 border border-gray-300 rounded-lg
hover:bg-gray-50"
              disabled={processing}
            >
              Annuler
            </button>
            <button
              type="submit"
              className="flex-1 px-6 py-3 bg-blue-600 text-white rounded-lg
hover:bg-blue-700 disabled:opacity-50"
              disabled={processing}
            >
              {processing ? 'Traitement...' : `Payer ${amount}€`}
            </button>
          </div>
        </form>

        {/* Disclaimer */}
        <p className="text-xs text-gray-500 text-center mt-4">
          🔒 Paiement fictif pour démonstration. Aucune transaction réelle n'est
effectuée.
        </p>
      </div>
    </div>
  );
};
```

---

6. Sécurité

6.1 Mesures Implémentées

1. Authentification JWT

- Tokens expiration: 7 jours
- Stockage: localStorage (client)

- Secret: Variable d'environnement
- Middleware: Vérification sur toutes routes protégées

2. Hachage Mots de Passe

- Algorithme: bcrypt
- Salt rounds: 10 (équilibre sécurité/performance)
- Jamais stockés en clair
- Politique minimale: 6 caractères (à renforcer en prod)

3. Validation Entrées

- TypeScript: Typage fort frontend/backend
- Regex: Email, cartes bancaires, dates
- Sanitization: trim(), toLowerCase()
- Validation double: Frontend + Backend

4. CORS Configuration

```
app.use(cors({
  origin: 'http://localhost:3000', // Whitelist frontend
  credentials: true,
  methods: ['GET', 'POST', 'PUT', 'DELETE'],
  allowedHeaders: ['Content-Type', 'Authorization']
}));
```

5. Protection Données Sensibles

- Pas de logs des mots de passe
- Pas de numéros CB stockés (paiement fictif)
- Pas d'emails en clair dans logs
- SQLite permissions: 644 (lecture/écriture propriétaire)

6.2 Checklist Production

Critique (Avant Mise en Ligne):

- ☐ **HTTPS obligatoire** (Let's Encrypt gratuit)
- ☐ **JWT_SECRET fort** (32+ caractères aléatoires)
- ☐ **Variables d'environnement** (.env.production, jamais commit Git)
- ☐ **Rate Limiting** (express-rate-limit: 100 req/15min)
- ☐ **Helmet.js** (Headers sécurisés: CSP, HSTS, X-Frame-Options)
- ☐ **Input sanitization** (express-validator)
- ☐ **SQL Injection protection** (Prepared statements ☑ déjà fait)
- ☐ **XSS protection** (React escape automatique ☑)
- ☐ **CSRF tokens** (pour formulaires sensibles)

Important (Premier Mois):

- ☐ Monitoring erreurs (Sentry)
- ☐ Logs centralisés (Winston + CloudWatch)
- ☐ Backup BDD quotidien automatique
- ☐ Tests de pénétration (OWASP Top 10)
- ☐ Audit RGPD complet (DPO externe)
- ☐ Documentation sécurité (incident response plan)

Recommandé (Premier Trimestre):

- ☐ 2FA/MFA pour authentification
- ☐ WAF (Cloudflare, AWS WAF)
- ☐ DDoS protection
- ☐ Penetration testing professionnel
- ☐ Certification ISO 27001 (si cible entreprise)
- ☐ Bug Bounty program (HackerOne, Bugcrowd)

---

7. Déploiement

7.1 Option 1: Vercel + Railway (Recommandé MVP)

Frontend (Vercel)

Avantages:

- Déploiement automatique depuis GitHub
- CDN global (Edge Network)
- HTTPS gratuit avec certificat auto-renouvelé
- Preview deployments pour chaque PR
- Analytics intégré

Steps:

```
# 1. Installer Vercel CLI
npm install -g vercel

# 2. Build frontend
cd cyber-solution
npm run build

# 3. Deploy
vercel --prod

# Ou connecter GitHub repo sur vercel.com
```

Configuration (vercel.json):

```
{
  "buildCommand": "npm run build",
  "outputDirectory": "dist",
  "devCommand": "npm run dev",
  "framework": "vite",
  "rewrites": [
    { "source": "/(.*)", "destination": "/index.html" }
  ]
}
```

Coût: Gratuit (plan Hobby jusqu'à 100 GB bandwidth)

Backend (Railway)

Avantages:

- Support Node.js + SQLite natif
- Variables d'environnement sécurisées
- Logs en temps réel
- Scaling automatique
- Monitoring intégré

Steps:

```
# 1. Créer compte sur railway.app
# 2. Créer nouveau projet "cyber-solution-backend"
# 3. Connecter GitHub repo

# 4. Configurer variables d'environnement:
JWT_SECRET=production-secret-key-CHANGE-ME-32-chars
NODE_ENV=production
PORT=3001

# 5. Railway détecte package.json et lance:
npm install
npm run build
npm start
```

Configuration (package.json):

```
{
  "scripts": {
    "start": "NODE_ENV=production npx tsx server/index.ts",
    "build": "echo 'No build needed for tsx'"
  },
  "engines": {
    "node": "20.x"
```

```
  }
}
```

Coût: $5/mois (500h runtime incluses)

7.2 Option 2: VPS (Digital Ocean / OVH)

Configuration Serveur:

```
# Ubuntu 22.04 LTS - 2 GB RAM - $12/mois

# 1. Installer Node.js 20
curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
sudo apt-get install -y nodejs

# 2. Installer PM2 (process manager)
sudo npm install -g pm2

# 3. Cloner repository
git clone https://github.com/username/cyber-solution.git
cd cyber-solution

# 4. Installer dépendances
npm install

# 5. Build frontend
npm run build

# 6. Configurer Nginx (reverse proxy)
sudo apt install nginx
sudo nano /etc/nginx/sites-available/cyber-solution

# Configuration Nginx:
server {
    listen 80;
    server_name cyber-solution.fr;

    # Frontend (static files)
    location / {
        root /var/www/cyber-solution/dist;
        try_files $uri /index.html;
    }

    # Backend (proxy)
    location /api {
        proxy_pass http://localhost:3001;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
```

```
    }
}

# 7. Activer site
sudo ln -s /etc/nginx/sites-available/cyber-solution /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl reload nginx

# 8. Installer SSL (Let's Encrypt)
sudo apt install certbot python3-certbot-nginx
sudo certbot --nginx -d cyber-solution.fr -d www.cyber-solution.fr

# 9. Démarrer backend avec PM2
cd /home/user/cyber-solution
pm2 start "npx tsx server/index.ts" --name cyber-solution-backend
pm2 save
pm2 startup

# 10. Monitoring
pm2 monit
pm2 logs cyber-solution-backend
```

Coût: $12/mois + $15/an domaine = ~$160/an

7.3 CI/CD Pipeline (GitHub Actions)

```
# .github/workflows/deploy.yml

name: Deploy to Production

on:
  push:
    branches: [main]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: Setup Node.js
        uses: actions/setup-node@v3
        with:
          node-version: '20'

      - name: Install dependencies
        run: npm ci

      - name: Run TypeScript check
        run: npx tsc --noEmit
```

```
      - name: Run tests (when available)
        run: npm test

  deploy-frontend:
    needs: test
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: Deploy to Vercel
        uses: amondnet/vercel-action@v25
        with:
          vercel-token: ${{ secrets.VERCEL_TOKEN }}
          vercel-org-id: ${{ secrets.VERCEL_ORG_ID }}
          vercel-project-id: ${{ secrets.VERCEL_PROJECT_ID }}
          vercel-args: '--prod'

  deploy-backend:
    needs: test
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: Deploy to Railway
        run: |
          npm install -g railway
          railway up --service cyber-solution-backend
        env:
          RAILWAY_TOKEN: ${{ secrets.RAILWAY_TOKEN }}
```

---

8. Guide d'Installation

8.1 Prérequis

- **Node.js 20+** (nodejs.org)
- **npm 9+** (inclus avec Node.js)
- **Git** (git-scm.com)
- **PowerShell 7+** (Windows) ou Bash (Linux/Mac)

8.2 Installation Locale

```
# 1. Cloner le repository
git clone https://github.com/username/cyber-solution.git
cd cyber-solution

# 2. Installer dépendances
npm install

# 3. Initialiser la base de données
```

```
cd server
npx tsx database.ts  # Crée cyber-solution.db avec seed data

# 4. Lancer l'application (Option A: Script PowerShell)
cd ..
./start.ps1

# Ou Option B: Manuellement (2 terminaux)
# Terminal 1 (Backend):
cd server
npx tsx index.ts

# Terminal 2 (Frontend):
npm run dev

# 5. Ouvrir navigateur
# Frontend: http://localhost:3000
# Backend API: http://localhost:3001
```

8.3 Variables d'Environnement

```
# server/.env (Créer ce fichier)

# Backend
PORT=3001
JWT_SECRET=dev-secret-key-CHANGE-IN-PRODUCTION
NODE_ENV=development

# Database
DATABASE_PATH=./cyber-solution.db

# Frontend (optionnel)
VITE_API_URL=http://localhost:3001
```

8.4 Vérification Installation

```
# Check versions
node --version  # v20.x ou supérieur
npm --version   # v9.x ou supérieur

# Check backend
curl http://localhost:3001/api/trainings
# Devrait retourner 401 (Unauthorized) si backend fonctionne

# Check frontend
curl http://localhost:3000
# Devrait retourner HTML

# Check database
```

```
cd server
node check-db.js
# Affiche: 0 users, 5 trainings, 0 audits
```

---

9. Guide de Développement

9.1 Structure du Code

```
cyber-solution/
├─ package.json              → Dépendances npm
├─ tsconfig.json             → Configuration TypeScript
├─ vite.config.ts            → Configuration Vite (frontend)
├─ tailwind.config.js        → Configuration Tailwind CSS
├─ start.ps1                 → Script lancement (PowerShell)
│
├─ index.html                → Entry point HTML
├─ index.tsx                 → Entry point React
├─ index.css                 → Styles globaux Tailwind
├─ App.tsx                   → Router principal
│
├─ components/               → Composants React
│  ├─ *.tsx                  → Fichiers TypeScript React
│
├─ contexts/                 → Contextes React
│  └─ AuthContext.tsx
│
├─ server/                   → Backend Node.js
│  ├─ index.ts               → Serveur Express
│  ├─ database.ts            → Schema + Seed SQLite
│  ├─ check-db.js            → Script vérification BDD
│  └─ cyber-solution.db      → Base de données SQLite
│
└─ docs/                     → Documentation (ce fichier)
   ├─ UML-Diagrams.md
   ├─ MCD-MLD.md
   ├─ MVP-Technologies.md
   └─ Documentation-Technique.md
```

9.2 Conventions de Code

Naming Conventions

```
// Composants React: PascalCase
export const Dashboard: React.FC = () => { };
export const PaymentModal: React.FC<Props> = ({ }) => { };

// Interfaces: PascalCase avec "I" optionnel
interface User { }
```

```
interface PaymentData { }

// Fonctions: camelCase
function calculateScore() { }
const handleSubmit = () => { };

// Constantes: UPPER_SNAKE_CASE
const JWT_SECRET = 'secret';
const MAX_RETRIES = 3;

// Fichiers: PascalCase pour composants, camelCase pour utilitaires
Dashboard.tsx
authUtils.ts
```

TypeScript Best Practices

```
// ☑ Bon: Types explicites
interface Training {
  id: number;
  title: string;
  price: number;
}

function getTraining(id: number): Training | null {
  // ...
}

// ❌ Mauvais: Types any
function getTraining(id: any): any {
  // ...
}

// ☑ Bon: Props typées
interface Props {
  title: string;
  onClose: () => void;
}

// ❌ Mauvais: Props non typées
function Modal(props: any) { }
```

9.3 Workflow Git

```
# 1. Créer branche feature
git checkout -b feature/payment-stripe

# 2. Développer avec commits atomiques
git add components/PaymentModal.tsx
git commit -m "feat: Add Stripe payment modal"
```

```
git add server/routes/payment.ts
git commit -m "feat: Add Stripe webhook handler"

# 3. Push et créer Pull Request
git push origin feature/payment-stripe

# 4. Code Review → Merge dans main
# 5. Delete feature branch
git branch -d feature/payment-stripe
```

Format Commits (Conventional Commits):

```
feat: Add new feature
fix: Fix bug
docs: Update documentation
style: Format code (no logic change)
refactor: Refactor code
test: Add tests
chore: Update dependencies
```

9.4 Debugging

Frontend (React DevTools)

```
# Installer extension Chrome/Firefox:
# https://react.dev/learn/react-developer-tools

# Inspecter composants, props, state
# F12 → Onglet "Components"
```

Backend (Node.js Debugger)

```
# Lancer avec debugger
node --inspect server/index.ts

# Ouvrir Chrome DevTools
chrome://inspect

# Ou VS Code (launch.json):
{
  "type": "node",
  "request": "launch",
  "name": "Debug Backend",
  "program": "${workspaceFolder}/server/index.ts",
  "runtimeExecutable": "npx",
```

```
  "runtimeArgs": ["tsx"]
}
```

Database (SQLite Browser)

```
# Télécharger DB Browser: https://sqlitebrowser.org/

# Ouvrir cyber-solution.db
# Browse Data → users, audit_results, etc.

# Ou ligne de commande:
sqlite3 cyber-solution.db
sqlite> SELECT * FROM users;
sqlite> .schema users
sqlite> .exit
```

---

10. Maintenance & Troubleshooting

10.1 Problèmes Fréquents

Problème: "Port 3000/3001 already in use"

```
# Solution: Tuer processus existant
Get-NetTCPConnection -LocalPort 3000 | Select-Object -ExpandProperty OwningProcess
| ForEach-Object { Stop-Process -Id $_ -Force }

# Ou relancer start.ps1 (gère automatiquement)
.\start.ps1
```

Problème: "JWT token expired"

```
// Frontend: Rafraîchir token ou reconnecter
localStorage.removeItem('token');
window.location.href = '/login';

// Backend: Augmenter expiration (si nécessaire)
jwt.sign({ userId }, SECRET, { expiresIn: '30d' }); // Au lieu de 7d
```

Problème: "Database locked" (SQLite)

```
# Cause: Plusieurs processus écrivent simultanément
```

```
# Solution 1: Fermer tous les processus Node.js
Get-Process node | Stop-Process -Force

# Solution 2: Migrer vers PostgreSQL si > 100 req/sec
```

Problème: "CORS error"

```
// Backend: Vérifier origin autorisé
app.use(cors({
  origin: 'http://localhost:3000', // Doit matcher frontend URL
  credentials: true
}));

// Frontend: Vérifier URL API
const response = await fetch('http://localhost:3001/api/trainings', { ... });
```

10.2 Logs & Monitoring

Logs Backend (Production)

```
// Installer Winston
npm install winston

// logger.ts
import winston from 'winston';

export const logger = winston.createLogger({
  level: 'info',
  format: winston.format.json(),
  transports: [
    new winston.transports.File({ filename: 'error.log', level: 'error' }),
    new winston.transports.File({ filename: 'combined.log' })
  ]
});

// Utiliser dans routes
logger.info('User registered', { userId: user.id, email: user.email });
logger.error('Login failed', { email, error: err.message });
```

Monitoring (Sentry)

```
# Installer Sentry SDK
npm install @sentry/react @sentry/node

# Frontend (index.tsx)
import * as Sentry from '@sentry/react';
```

```
Sentry.init({
  dsn: 'https://xxx@sentry.io/yyy',
  environment: 'production'
});

# Backend (index.ts)
import * as Sentry from '@sentry/node';

Sentry.init({ dsn: '...' });

app.use(Sentry.Handlers.errorHandler());
```

10.3 Backup & Restore

Backup Base de Données

```
# Manuel
cp server/cyber-solution.db backups/cyber-solution-$(date +%Y%m%d).db

# Automatique (cron job Linux)
0 2 * * * cp /path/to/cyber-solution.db /backups/db-$(date +\%Y\%m\%d).db

# Windows (Task Scheduler PowerShell)
Copy-Item "C:\cyber-solution\server\cyber-solution.db" "C:\backups\db-$(Get-Date -
Format 'yyyyMMdd').db"
```

Restore

```
# Arrêter backend
pm2 stop cyber-solution-backend

# Restaurer backup
cp backups/cyber-solution-20260128.db server/cyber-solution.db

# Relancer backend
pm2 start cyber-solution-backend
```

10.4 Mises à Jour

Update Dépendances

```
# Check outdated
npm outdated

# Update mineur (recommandé)
```

```
npm update

# Update majeur (avec précaution)
npm install react@latest
npm install express@latest

# Check breaking changes
npm install -g npm-check-updates
ncu --upgrade
npm install
```

Migration Base de Données

```
-- Ajouter colonne (exemple)
ALTER TABLE users ADD COLUMN phone TEXT;

-- Modifier structure (SQLite limitation: recréer table)
-- 1. Créer nouvelle table
CREATE TABLE users_new (
  id INTEGER PRIMARY KEY,
  email TEXT UNIQUE NOT NULL,
  password_hash TEXT NOT NULL,
  company_name TEXT NOT NULL,
  phone TEXT,  -- Nouvelle colonne
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);

-- 2. Copier données
INSERT INTO users_new (id, email, password_hash, company_name, created_at)
SELECT id, email, password_hash, company_name, created_at FROM users;

-- 3. Supprimer ancienne table
DROP TABLE users;

-- 4. Renommer
ALTER TABLE users_new RENAME TO users;

-- 5. Recréer index
CREATE UNIQUE INDEX idx_users_email ON users(email);
```

Annexes

A. Commandes Utiles

```
# Frontend
npm run dev                 # Lancer dev server
npm run build               # Build production
npm run preview             # Preview build
```

```
npm run lint                # Linter

# Backend
npx tsx server/index.ts     # Lancer backend
node server/check-db.js     # Vérifier BDD
sqlite3 server/cyber-solution.db # Console SQLite

# Database
sqlite3 cyber-solution.db ".dump" > backup.sql  # Export SQL
sqlite3 cyber-solution.db < backup.sql          # Import SQL

# Git
git log --oneline --graph   # Historique visuel
git stash                   # Sauvegarder modifications
git stash pop               # Restaurer modifications

# Node.js
npm list                    # Liste dépendances
npm prune                   # Nettoyer node_modules
npm ci                      # Clean install (CI/CD)
```

B. Ressources Externes

Documentation:

- React: https://react.dev
- TypeScript: https://www.typescriptlang.org/docs
- Express: https://expressjs.com
- SQLite: https://www.sqlite.org/docs.html
- Tailwind CSS: https://tailwindcss.com/docs

Outils:

- Vite: https://vitejs.dev
- Vitest: https://vitest.dev
- Playwright: https://playwright.dev
- Sentry: https://docs.sentry.io

Sécurité:

- OWASP Top 10: https://owasp.org/www-project-top-ten
- JWT Best Practices: https://jwt.io/introduction
- bcrypt Guide: https://github.com/kelektiv/node.bcrypt.js

NIS2 & RGPD:

- Directive NIS2: https://digital-strategy.ec.europa.eu/en/policies/nis2-directive
- ANSSI Guides: https://www.ssi.gouv.fr
- CNIL RGPD: https://www.cnil.fr/fr/reglement-europeen-protection-donnees

---

Glossaire

- **API:** Application Programming Interface
- **JWT:** JSON Web Token (authentification stateless)
- **CORS:** Cross-Origin Resource Sharing
- **MCD:** Modèle Conceptuel de Données
- **MLD:** Modèle Logique de Données
- **MVP:** Minimum Viable Product
- **NIS2:** Network and Information Security Directive (EU 2022/2555)
- **RGPD:** Règlement Général sur la Protection des Données (GDPR)
- **SaaS:** Software as a Service
- **SPA:** Single Page Application
- **CRUD:** Create, Read, Update, Delete
- **ORM:** Object-Relational Mapping
- **CI/CD:** Continuous Integration / Continuous Deployment
- **DPO:** Data Protection Officer
- **RSSI:** Responsable de la Sécurité des Systèmes d'Information
- **DSI:** Directeur des Systèmes d'Information

Version: 1.0
Date: 28 Janvier 2026
Auteurs: Équipe Cyber Solution
Contact: dpo@cyber-solution.fr