Projet : Plateforme e-learning "Knowledge Learning"

Développeur : Pierre FAIVRE

Soutenance de projet

Contexte et entreprise

- Entreprise fictive : Knowledge
- Éditeur de livres de formation (musique, informatique, jardinage)
- Objectif : proposer ses formations en ligne
- Contexte : développement d'une plateforme e-learning/e-commerce

Objectifs du projet

- Créer un back-end complet et sécurisé
- Gérer comptes, achats, accès aux contenus
- Assurer l'autonomie de l'utilisateur
- Respecter les exigences techniques et fonctionnelles

Fonctionnalités principales

- Création de compte avec activation par mail
- Connexion utilisateur
- Achat de cursus ou de leçon
- Validation des leçons et obtention de certification
- Gestion backoffice pour les administrateurs
- Sécurité renforcée (CSRF, rôles, mot de passe)

Conception de la base de données

- Méthode utilisée : [Merise]
- Tables principales : Utilisateurs, rôles, leçons, cursus, thèmes, achats, certifications
- Relations entre entités (exemple : un cursus contient plusieurs leçons)
- Champs de traçabilité : created_at, updated_at

Mise en place technique

- Back-end : Node.js avec Express
- Sequelize
- Architecture MVC
- Sécurité : Middleware CSRF, hashage des mots de passe
- Vérification des rôles et permissions (admin, client)

Intégration e-commerce

- Paiement simulé via Stripe sandbox
- Achat d’un contenu = attribution des droits d’accès
- Système de validation des leçons
- Certification automatique à la validation d’un cursus

Tests et qualité

- Tests unitaires (Jest)
- Fonctions testées : Création de compte, activation, connexion, achat, accès sécurisé
- Objectif : valider fonctionnalités et sécurité

Recherches effectuées

- Documentation technique : Express, ORM, sécurité web, Stripe
- Sources : MDN, StackOverflow, blogs spécialisés
- Problèmes rencontrés et solutions apportées
- Bonnes pratiques mises en œuvre

Présentation Front-end

- Création d’un compte
- Activation du compte par mail
- Connexion utilisateur
- Achat de cursus ou leçon
- Validation des leçons et obtention de la certification

Bilan personnel

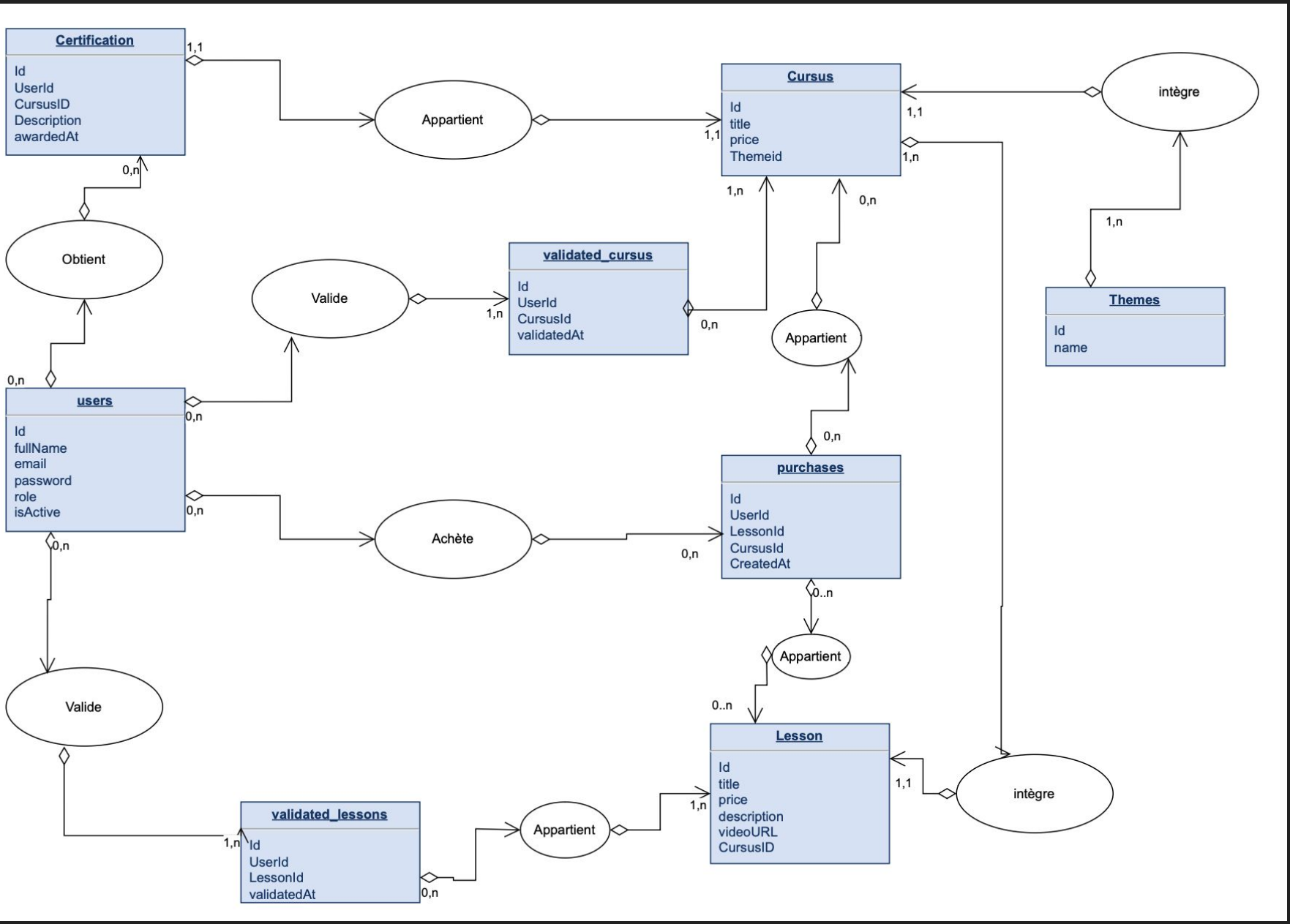
- Compétences acquises : Sécurité web, structuration MVC, base de données relationnelle
- Points forts du projet : Organisation claire, respect des rôles, architecture propre
- Limites : Front-end simplifié (focus back-end)

Perspectives d’évolution

- Intégration de quiz
- Suivi de progression personnalisé
- Backoffice enrichi
- Refonte visuelle et responsive design

Conclusion

- Projet conforme au cahier des charges
- Solution fonctionnelle, évolutive et sécurisée
- Expérience complète en développement back-end
- Merci pour votre attention


Certification
Id
UserId
CursusID
Description
awardedAt
1,1
Appartient
Cursus
Id
title
price
Themeid
1,1
1,n
intègre
1,n
0,n
Obtient
Valide
validated_cursus
Id
UserId
CursusId
validatedAt
1,n
0,n
Appartient
Themes
Id
name
0,n
users
Id
fullName
email
password
role
isActive
0,n
0,n
purchases
Id
UserId
LessonId
CursusId
CreatedAt
Achète
0,n
0,n
0..n
Appartient
Valide
0..n
Lesson
Id
title
price
description
videoURL
CursusID
1,1
intègre
validated_lessons
Id
UserId
LessonId
validatedAt
1,n
0,n
Appartient
1,n

[Retour]

```
    const token = jwt.sign(
      {
        id: user.id,
        email: user.email,
        role: user.role
      },
      process.env.JWT_SECRET,
      { expiresIn: '24h' }
    );

    res.cookie('token', token, {
      httpOnly: true,
      sameSite: 'none',
      secure: true,
      maxAge: 1000 * 60 * 60 * 24
    });

    res.status(200).json({ message: 'Login successful', token });
  } catch (error) {
    console.error('Login error:', error);
    res.status(500).json({ message: 'Server error' });
  }
};
```

```
const csrfProtection = csrf({ cookie: true })

router.get('/token', csrfProtection, (req, res) => {
  res.cookie('XSRF-TOKEN', req.csrfToken())
  res.status(200).json({ message: 'CSRF token envoyé' })
})

module.exports = router
```

Retour

```vue
<script setup>
import api from '@/utils/api'

const props = defineProps({
  cursusId: Number,
  amount: Number // valeur en euros (ex: 49)
})

const checkout = async () => {
  try {
    const csrfRes = await api.get('/security/csrf-token', { withCredentials: true })
    const csrfToken = csrfRes.data.csrfToken

    // Crée la session Stripe côté back-end
    const res = await api.post('/stripe/create-checkout-session', {
      cursusId: props.cursusId,
      amount: props.amount
    }, {
      headers: { 'X-CSRF-Token': csrfToken },
      withCredentials: true
    })

    const stripe = await loadStripe(import.meta.env.VITE_STRIPE_PUBLISHABLE_KEY)
    await stripe.redirectToCheckout({ sessionId: res.data.sessionId })

  } catch (err) {
    console.error('Erreur lors du paiement Stripe :', err)
    alert('❌ Paiement échoué')
  }
}
</script>
```

```vue
<ul v-if="cursus.length">
  <li v-for="c in cursus" :key="c.id">
    <router-link :to="`/cursus/${c.id}/lessons`">
      {{ c.title }} – {{ c.price }} €
    </router-link>
    <StripeCursusButton :cursus-id="c.id" :amount="c.price.toString()" />
  </li>
</ul>
```

Retour

```js
router.post('/create-checkout-session', async (req, res) => {
  const { amount, successUrl, cancelUrl } = req.body;

  try {
    const session = await stripe.checkout.sessions.create({
      mode: 'payment',
      payment_method_types: ['card'],
      line_items: [{
        price_data: {
          currency: 'eur',
          product_data: {
            name: 'Achat leçon ou cursus',
          },
          unit_amount: Math.round(parseFloat(amount) * 100),
        },
        quantity: 1,
      }],
      success_url: successUrl,
      cancel_url: cancelUrl,
    });

    res.json({ id: session.id });
  } catch (err) {
    console.error('Erreur Stripe :', err);
    res.status(500).json({ error: 'Erreur création session Stripe' });
  }
});
```

```
exports.validateLesson = async (req, res) => {
  const userId = req.user.id;
  const lessonId = req.params.id;

  try {
    const lesson = await Lesson.findByPk(lessonId);
    if (!lesson) return res.status(404).json({ message: "Leçon introuvable." });

    const already = await ValidatedLesson.findOne({
      where: { UserId: userId, LessonId: lessonId }
    });
    if (already) {
      return res.status(409).json({ message: "Leçon déjà validée." });
    }

    await ValidatedLesson.create({ UserId: userId, LessonId: lessonId });

    // 🔁 Vérifie les autres leçons du cursus
    const allLessons = await Lesson.findAll({ where: { CursusId: lesson.CursusId } });
    const lessonIds = allLessons.map(l => l.id);

    const validated = await ValidatedLesson.findAll({
      where: { UserId: userId, LessonId: lessonIds }
    });

    const purchased = await Purchase.findAll({
      where: { UserId: userId, LessonId: lessonIds }
    });

    const allValidated = validated.length === lessonIds.length;
    const allPurchased = purchased.length === lessonIds.length;

    if (allValidated && allPurchased) {
  // ✅ Crée ou trouve la validation de cursus
  const [validatedCursus] = await ValidatedCursus.findOrCreate({
    where: { UserId: userId, CursusId: lesson.CursusId }
  });
```

Retour

```
exports.validateCursus = async (req, res) => {
  const userId = req.user.id;
  const cursusId = req.params.id;

  try {
    const lessons = await Lesson.findAll({ where: { CursusId: cursusId } });

    for (const lesson of lessons) {
      await Purchase.findOrCreate({
        where: { UserId: userId, LessonId: lesson.id }
      });

      await ValidatedLesson.findOrCreate({
        where: { UserId: userId, LessonId: lesson.id }
      });
    }

    await ValidatedCursus.findOrCreate({
      where: { UserId: userId, CursusId: cursusId }
    });

    await Certification.findOrCreate({
      where: { UserId: userId, CursusId: cursusId }
    });

    res.status(200).json({ message: 'Cursus validé avec succès' });

  } catch (err) {
    console.error('Erreur validation cursus :', err);
    res.status(500).json({ message: 'Erreur serveur lors de la validation du cursus.' });
  }
};
```

Retour

```
test('✅ Connexion utilisateur', async () => {
  const res = await request(app)
    .post('/auth/login')
    .set('X-CSRF-Token', csrfToken)
    .send({
      email: 'test@example.com',
      password: 'Azerty123!',
    });

  expect([200, 201]).toContain(res.statusCode);
  expect(res.body.token).toBeDefined();
});

test('✅ Achat de cursus via POST direct', async () => {
  const res = await agent
    .post('/buy/cursus/1')
    .set('X-CSRF-Token', csrfToken)
    .set('Authorization', `Bearer ${jwtToken}`);

  expect(res.statusCode).toBe(201);
});

test('✅ Achat de leçon via POST direct', async () => {
  const res = await agent
    .post('/buy/lesson/1')
    .set('X-CSRF-Token', csrfToken)
    .set('Authorization', `Bearer ${jwtToken}`);

  expect(res.statusCode).toBe(201);
});
```

```
Ran all test suites.
pierre@MacBook-Air-de-Pierre knowledge-learning % npm run test

> knowledge-learning@1.0.0 test
> NODE_ENV=test jest

  console.log
    📦 Base utilisée par Sequelize : sqlite:./test.db

      at Object.log (config/db.js:7:9)

  console.log
    POST session ID: IRIDvl68Oki0lSheI652kmgMpeaLC24t

      at log (routes/purchase.routes.js:11:13)

  console.log
    Cookies: undefined

      at log (routes/purchase.routes.js:12:13)

  console.log
    { UserId: 1, LessonId: 1 }

      at log (controllers/purchase.controller.js:15:13)

 PASS  test/api.test.js
  🧪 Tests des fonctionnalités d'achat (Stripe intégré)
    ✓ ✅ Création de compte utilisateur (1532 ms)
    ✓ ✅ Connexion utilisateur (116 ms)
    ✓ ✅ Achat de cursus via POST direct (16 ms)
    ✓ ✅ Achat de leçon via POST direct (25 ms)
    ✓ ✅ Récupération des leçons achetées (14 ms)
    ✓ ✅ Récupération des cursus achetés (9 ms)

Test Suites: 1 passed, 1 total
Tests:       6 passed, 6 total
Snapshots:   0 total
Time:        2.754 s
Ran all test suites.
pierre@MacBook-Air-de-Pierre knowledge-learning %
```

Retour

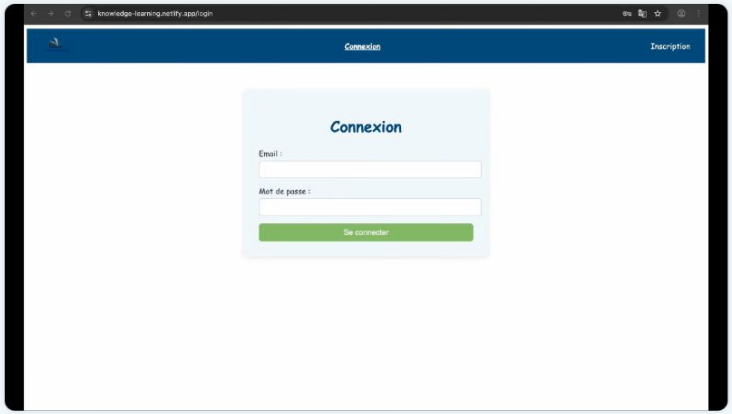
knowledge-learning.netlify.app

🎓 Bienvenue sur Knowledge Learning

La plateforme de formation qui vous permet d'apprendre à votre rythme. Achetez des leçons ou des cursus complets, suivez vos progrès, validez vos acquis, et obtenez des certifications 📜.

Explorer les formations | Créer un compte | Se connecter

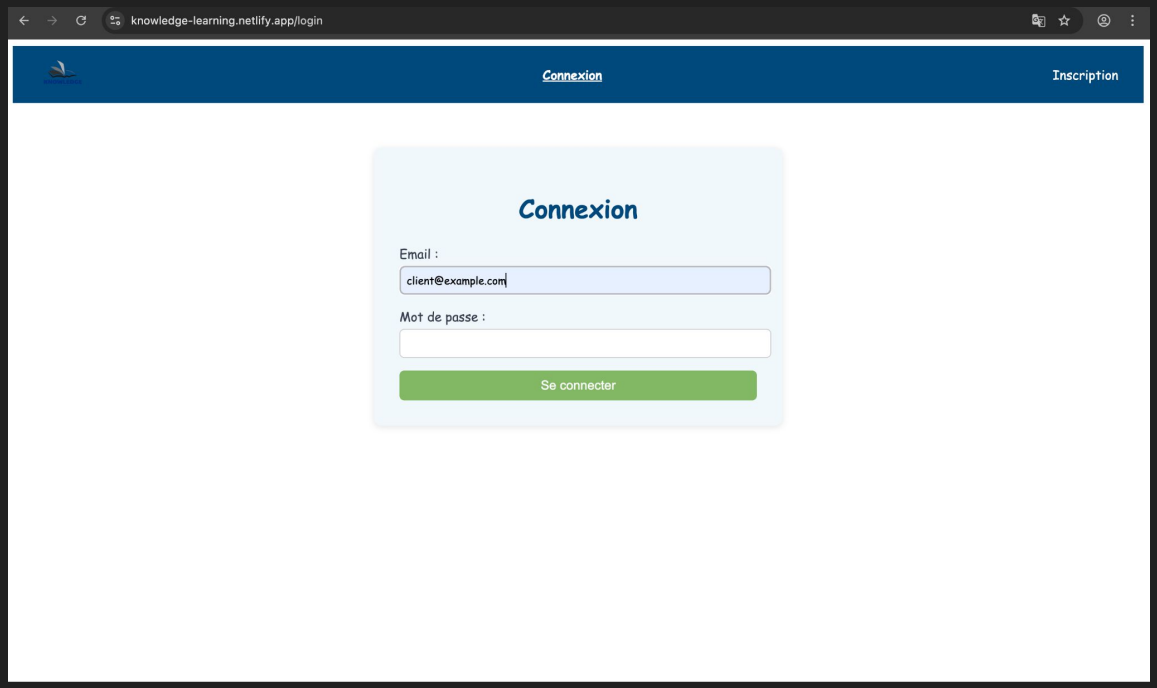
📽️ Découvrez le fonctionnement en vidéo

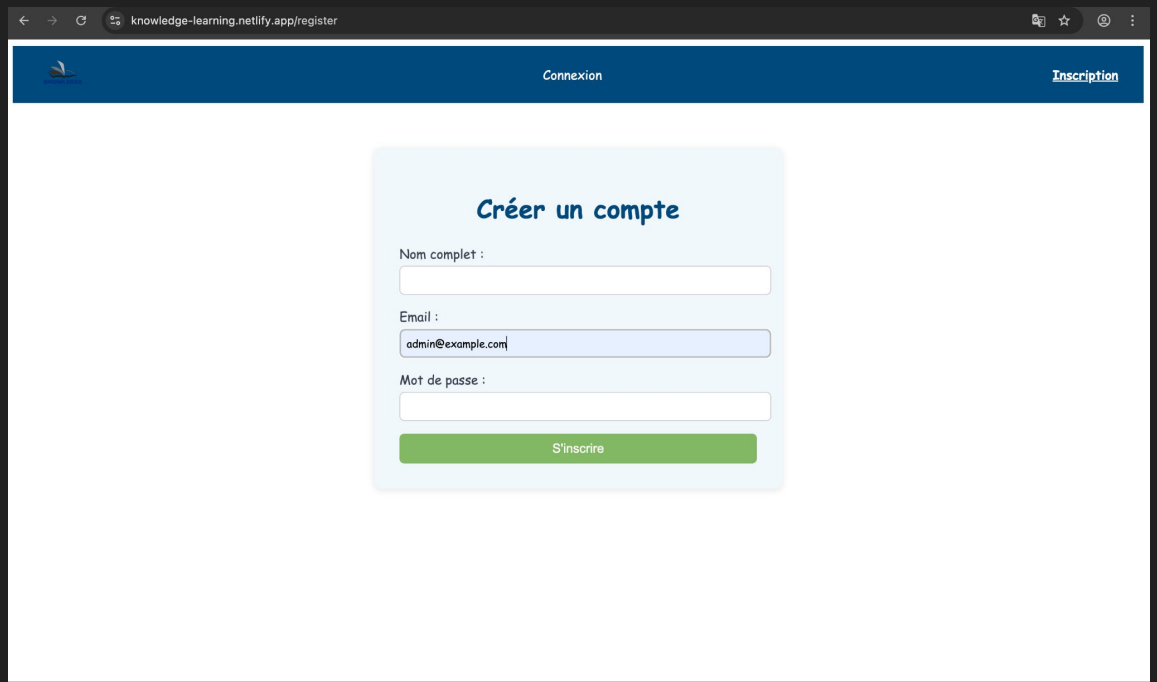


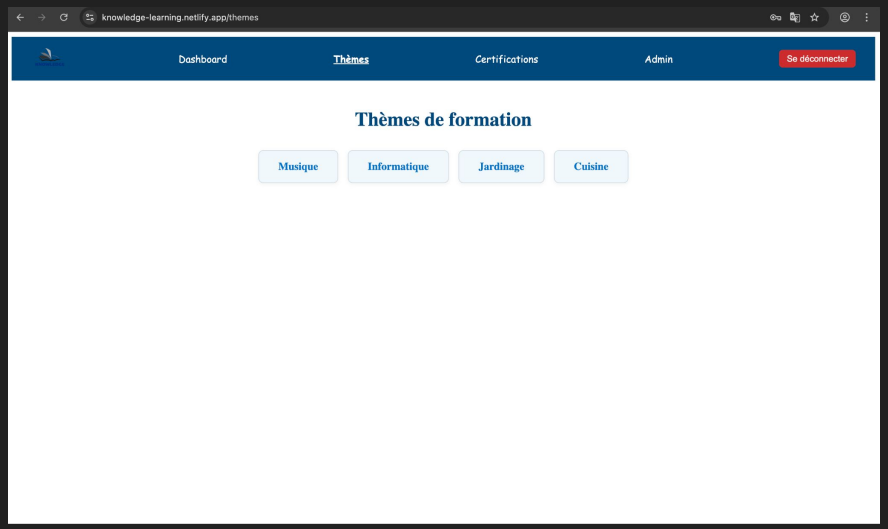
knowledge-learning.netlify.app/auth/verify/a8a9fe03-3baf-49f0-9055-da995309fab7

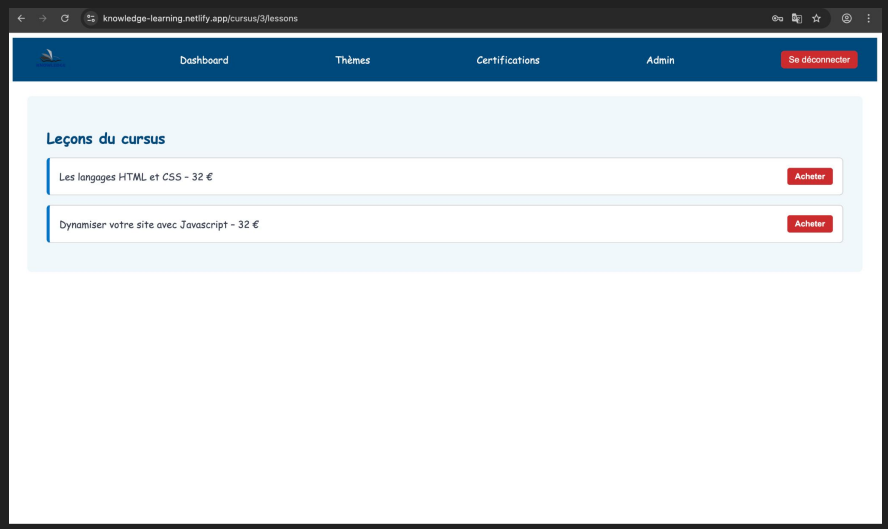
Connexion Inscription

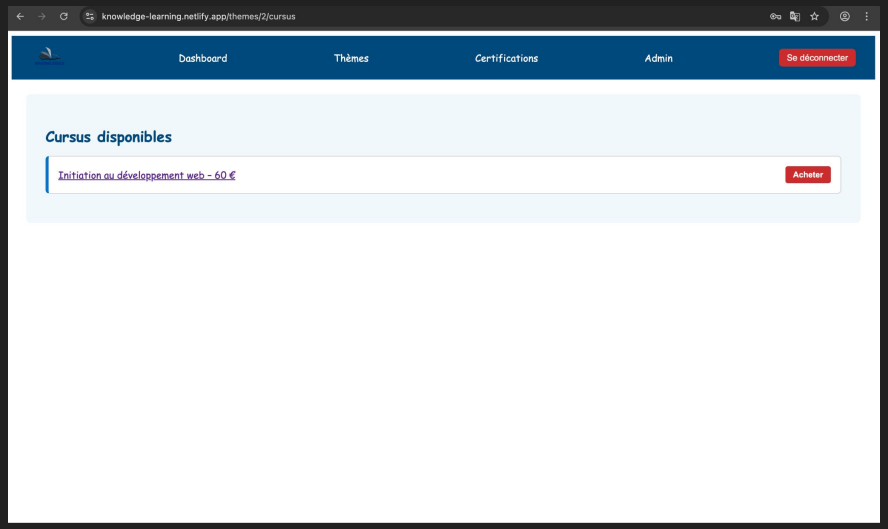
❌ **Lien invalide ou expiré.**

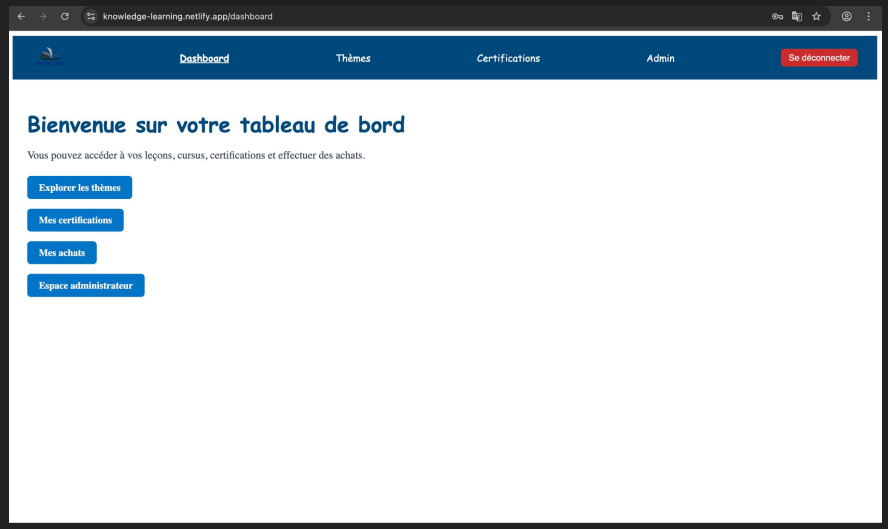

knowledge-learning.netlify.app/login
Connexion
Inscription
Connexion
Email :
client@example.com
Mot de passe :
Se connecter


knowledge-learning.netlify.app/register
Connexion
Inscription
Créer un compte
Nom complet :
Email :
admin@example.com
Mot de passe :
S'inscrire



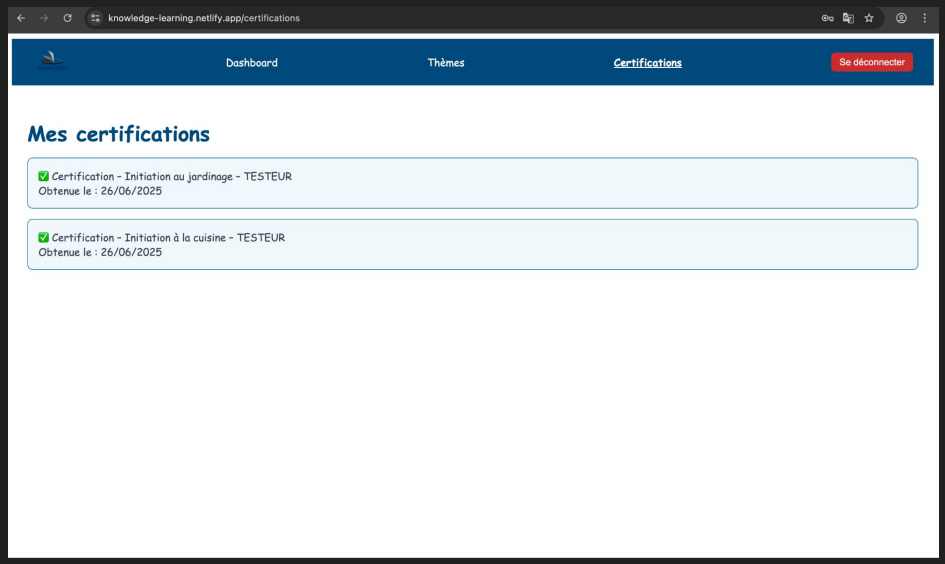


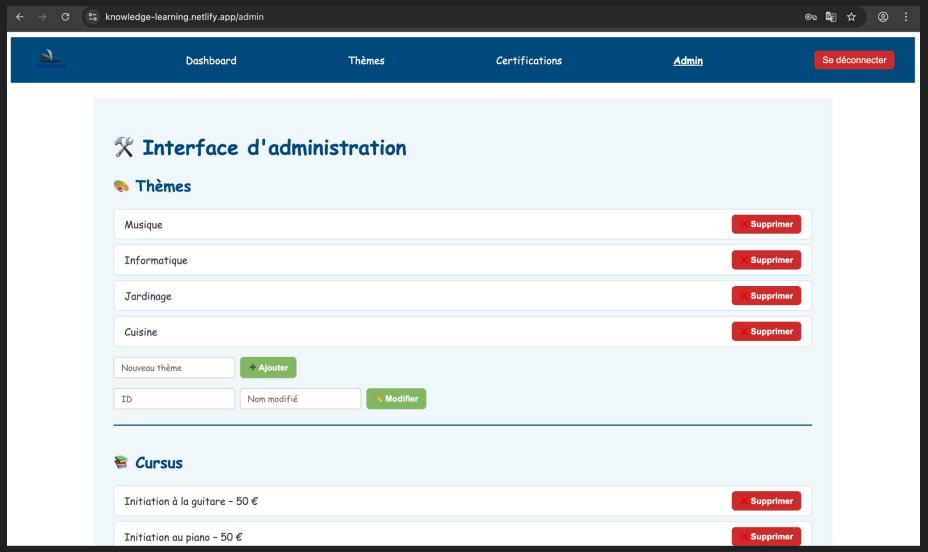


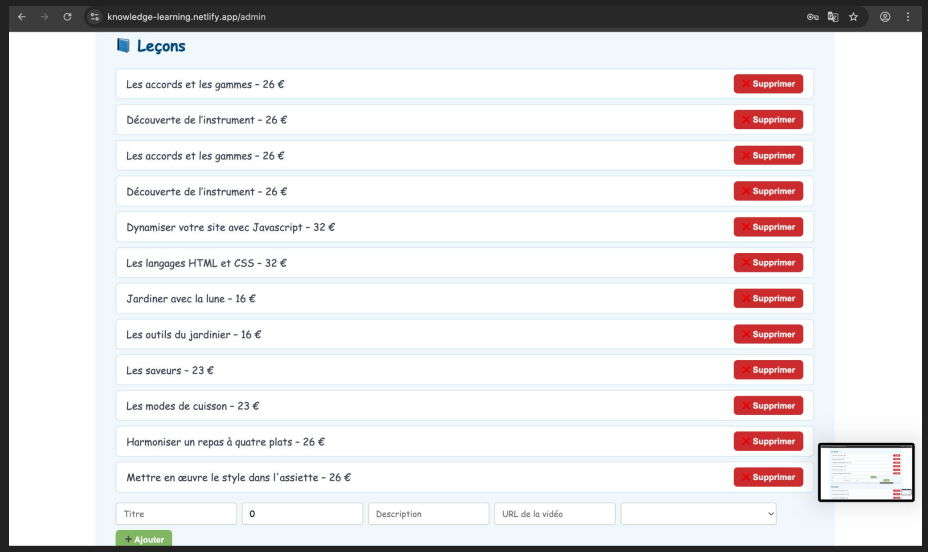


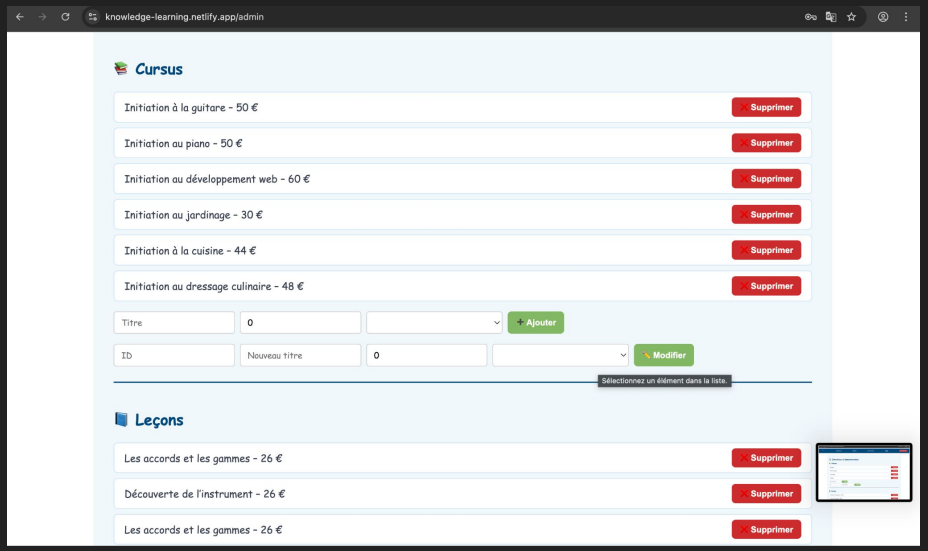
Thèmes - Cursus - Leçons et Dashboard

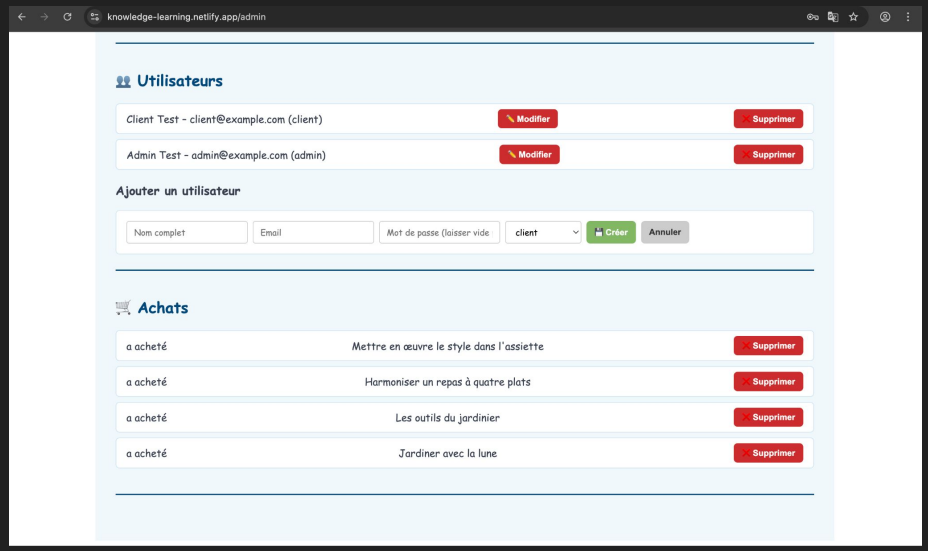
Mes achats et certifications











Interface Admin

[Retour]