Université Abdelmalek Essaâdi
faculté des sciences et techniques de tanger
Département des Mathématiques

Master Modélisation Mathématique et Science de Données

Projet de fin de module

Réalisé par :

Abdesalam Akhdim

Sujet :

Déploiement d'une Infrastructure PKI à Trois Niveaux

---

m

Encadrantes :

— **Pr.LECHHAB OUADRASSI Nihad** **Encadrante**

Année Universitaire 2024-2025

Table des matières

1 Introduction Générale

1.1 Contexte du projet

Dans un monde où la communication numérique est omniprésente, la sécurité des échanges d'informations constitue un enjeu majeur. Qu'il s'agisse de transactions bancaires, d'e-mails professionnels ou d'accès à des services en ligne, les données doivent être protégées contre toute forme de compromission. C'est dans ce contexte que les infrastructures à clé publique, connues sous l'acronyme **PKI** (Public Key Infrastructure), jouent un rôle central.

Une PKI permet de garantir l'authenticité, l'intégrité et la confidentialité des communications à travers l'utilisation de certificats numériques et de signatures électroniques. Elle repose sur des principes cryptographiques robustes qui facilitent la gestion sécurisée des identités numériques. Dans le cadre de ce projet, nous nous sommes intéressés à la conception et la réalisation d'une application web permettant de mettre en place une telle infrastructure PKI.

1.2 Objectifs du projet

Ce projet s'inscrit dans le cadre du **Projet de Fin de Master** du programme **Modélisation Mathématique et Science des Données** (MMSD). L'objectif principal est de développer une application web capable de :

- Générer des clés cryptographiques RSA ;
- Créer des demandes de certificats (CSR) ;
- Signer ces demandes via une autorité de certification intermédiaire ;
- Révoquer des certificats et gérer une CRL ;
- Vérifier la validité des certificats ;
- Fournir un tableau de bord avec des statistiques pertinentes.

L'application repose sur les technologies **Flask**, **Python** et la bibliothèque **cryptography**. Elle vise à illustrer concrètement les concepts théoriques de la cryptographie appliquée, tout en intégrant des principes modernes d'interface utilisateur via **Bootstrap** et **Chart.js**.

1.3 Cahier des charges

Les principales exigences fonctionnelles sont :

- Une interface conviviale permettant un accès simplifié aux fonctionnalités ;
- Une gestion sécurisée des clés, certificats et CRL ;

- Une vérification rigoureuse de l'authenticité et de la validité ;
- Un module de visualisation des certificats et statistiques.

2 Présentation de l'Infrastructure PKI

2.1 Définition et rôle d'une PKI

Une **Infrastructure à Clé Publique (PKI)** est un ensemble cohérent de services, de composants matériels et logiciels, ainsi que de règles organisationnelles visant à assurer la gestion des identités numériques et la sécurisation des communications électroniques. Elle repose sur la **cryptographie asymétrique**, fondée sur une paire de clés : une **clé publique** et une **clé privée**.

Son objectif principal est de :

- garantir l'authenticité des parties en communication ;
- assurer la confidentialité des échanges ;
- maintenir l'intégrité des données ;
- fournir la non-répudiation des transactions.

Grâce aux certificats numériques signés par une autorité de certification (CA), une PKI permet de lier une identité à une clé publique de manière fiable et vérifiable.

2.2 Composants d'une PKI

Une PKI complète est composée de plusieurs éléments interdépendants :

- **Autorité de Certification (CA)** : organe central qui délivre, signe et révoque les certificats. Elle peut être une *Root CA* ou une *Intermediate CA* ;
- **Autorité d'Enregistrement (RA)** : responsable de la vérification d'identité des utilisateurs avant la délivrance du certificat ;
- **Répertoire de certificats** : serveur de publication des certificats et des CRL, généralement accessible via LDAP ou HTTP ;
- **Clés cryptographiques** :
  - Clé privée : utilisée pour signer ou déchiffrer ;
  - Clé publique : utilisée pour vérifier ou chiffrer.
- **Certificats numériques** : fichiers conformes au standard *X.509*, contenant la clé publique, l'identité, la validité, la CA émettrice, etc. ;
- **CRL (Certificate Revocation List)** : liste signée par la CA contenant les certificats révoqués avant expiration.

2.3 Cycle de vie d'un certificat

Le cycle de vie d'un certificat comprend plusieurs étapes clés :

1. Génération d'une paire de clés (clé publique/privée) ;
2. Création d'une demande CSR (Certificate Signing Request) ;
3. Vérification d'identité par la RA ;
4. Signature du certificat par la CA ;
5. Distribution du certificat au demandeur ;
6. Utilisation dans des communications sécurisées ;
7. Expiration ou révocation anticipée (en cas de compromission).

2.4 Exigences de sécurité

Une infrastructure PKI efficace doit satisfaire les exigences de sécurité fondamentales suivantes :

- **Confidentialité** : empêcher l'accès non autorisé aux informations (via le chiffrement) ;
- **Authenticité** : garantir l'identité de l'émetteur grâce aux certificats ;
- **Intégrité** : assurer que les données ne sont ni altérées ni falsifiées (hashing, empreintes) ;
- **Non-répudiation** : empêcher l'auteur d'un message de nier l'avoir envoyé (signature numérique) ;
- **Traçabilité** : journalisation des émissions, utilisations et révocations de certificats.

3 Conception de l'Application Web

3.1 Choix technologiques

Le choix des technologies repose sur les objectifs pédagogiques du projet ainsi que sur la simplicité de mise en œuvre et la robustesse des bibliothèques :

— **Langage principal** : Python 3.11

— **Framework Web** : Flask — léger, extensible, facile à apprendre

— **Bibliothèque de cryptographie** : `cryptography` — puissante et conforme aux standards (X.509, RSA, CRL, etc.)

— **Interface utilisateur** : HTML5, CSS3, Bootstrap 5 — pour une UI réactive et responsive

— **Visualisation** : Chart.js — pour les graphiques statistiques dynamiques

— **Gestion des fichiers** : Système de fichiers local (structure de dossiers `pki/`, `final-certs/`, etc.)

Ces choix permettent une intégration fluide entre le backend et l'interface, tout en assurant la sécurité et la maintenabilité du projet.

3.2 Architecture générale de l'application

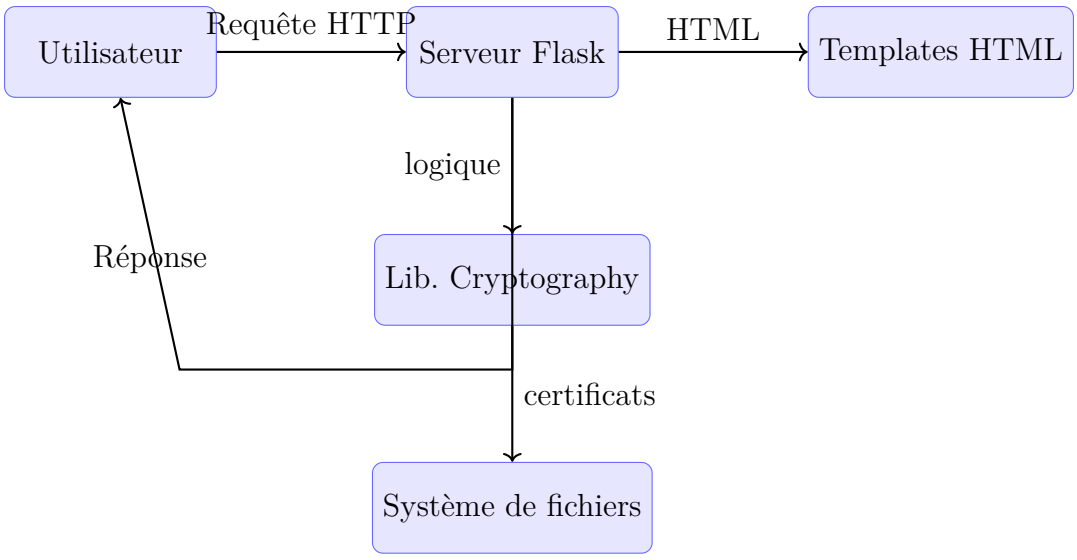
L'application suit une architecture **MVC simplifiée** adaptée à Flask :

— **Modèle** : gestion des fichiers (clés, certificats, CRL)

— **Vue (templates)** : pages HTML avec Jinja2 pour l'affichage dynamique

— **Contrôleur (routes Flask)** : logique métier de génération, signature, vérification, etc.

Schéma global du fonctionnement :



3.3 Diagramme de classes (modèle métier)

Le modèle métier est basé sur des entités représentées par fichiers ou objets manipulés :

— **CSR** : contient le nom commun, les données brutes et la méthode d'enregistrement.

— **Certificat** : nom, date de validité, statut de révocation, avec fonctions de signature et de révocation.

— **CA** : contient les clés, et la capacité à signer un CSR.

Relations :

— `CSR` est signé par la `CA`.

— `CA` émet un `Certificat`.

— `Certificat` peut être inscrit dans une `CRL`.

3.4 Modèle de données

L'application n'utilise pas de base de données relationnelle. Les données sont organisées par **structure de fichiers hiérarchique** :

— `pki/intermediateCA/` : contient la clé et le certificat de la CA intermédiaire

— `pki/final-certs/` : stocke les clés privées, CSR et certificats finaux

— `pki/crl/` : contient les fichiers CRL (en format PEM)

Exemple de structure :

```
pki/
 intermediateCA/
    intermediate.cert.pem
    intermediate.key.pem
 final-certs/
    exemple.csr.pem
    exemple.crt.pem
    exemple.key.pem
 crl/
     intermediate.crl.pem
```

Cette organisation facilite la portabilité, la lisibilité et la manipulation des opérations cryptographiques sans dépendre d'une base de données externe.

4 Implémentation du Projet

Ce chapitre présente en détail les principales fonctionnalités implémentées dans l'application web. Chaque fonctionnalité correspond à une étape importante de la gestion d'une infrastructure PKI.

4.1 Génération de la clé privée et CSR

L'utilisateur peut générer une paire de clés RSA (2048 bits) ainsi qu'une **demande de certificat** (CSR — *Certificate Signing Request*) via un formulaire web.

— La clé privée est enregistrée localement dans le dossier `final-certs/`.

— Le CSR contient la clé publique et les informations d'identité.

— Il est enregistré au format `PEM`.

Extrait de code :

```
key = rsa.generate_private_key(public_exponent=65537, key_size=2048)
csr = x509.CertificateSigningRequestBuilder().subject_name(x509.Name([...]))\
    .sign(key, hashes.SHA256())
```

Un message de succès confirme la création du CSR et de la clé privée.

4.2 Signature par l'autorité intermédiaire

Les demandes CSR générées peuvent être signées par la **CA intermédiaire** afin de produire des certificats finaux valides.

— La CA charge son certificat et sa clé privée.

— Le CSR est vérifié, puis signé.

— Le certificat contient la durée de validité (365 jours par défaut) et les extensions nécessaires.

Extrait de code :

```
cert = x509.CertificateBuilder()\
    .subject_name(csr.subject)\
    .issuer_name(ca_cert.subject)\
    .public_key(csr.public_key())\
    .serial_number(x509.random_serial_number())\
    .not_valid_before(datetime.utcnow())\
    .not_valid_after(datetime.utcnow() + timedelta(days=365))\
    .sign(ca_key, hashes.SHA256())
```

4.3 Révocation et gestion de la CRL

L'application permet de révoquer un certificat et d'ajouter son numéro de série à la **CRL** (Certificate Revocation List).

— Les certificats révoqués sont chargés à partir de la CRL existante.

— Une nouvelle entrée est ajoutée avec la date de révocation et une raison (ex : `key_compromise`).

— La CRL est ensuite signée par la CA et sauvegardée.

Cette méthode garantit que **toutes les révocations précédentes sont conservées** de manière persistante.

4.4 Vérification de certificats

L'utilisateur peut vérifier un certificat à partir de l'interface. L'application effectue plusieurs contrôles automatiques :

— **Validité temporelle** : certificat non expiré et actif.

— **Authenticité de l'émetteur** : signé par une CA autorisée.

— **Appartenance à la CRL** : non révoqué.

Résultats possibles :

— Certificat valide 2705

— Certificat expiré 23f3

— Certificat révoqué 274c

4.5 Tableau de bord et statistiques

L'application fournit un tableau de bord interactif pour la supervision de l'infrastructure PKI :

— Nombre de certificats signés

— Nombre de demandes CSR en attente

— Nombre de certificats révoqués

— Nombre de certificats valides

Ces données sont affichées via :

— **Chart.js** pour des diagrammes dynamiques (barres, camemberts, etc.)

— **Cartes Bootstrap** colorées pour une visualisation rapide

Exemple de données visualisées :

```
labels: ['Signés', 'CSR', 'Révoqués', 'Valides']
data: [12, 4, 3, 9]
```

Ce module permet une **analyse rapide et efficace** de l'état global du système PKI.

5 Interface Utilisateur

Ce chapitre est consacré à la présentation de l'interface graphique de l'application ainsi qu'à l'analyse de son ergonomie et de ses choix esthétiques. L'objectif est de proposer une expérience utilisateur fluide et intuitive tout en mettant en valeur les fonctionnalités offertes par le système PKI.

5.1 Présentation des interfaces

L'application comporte plusieurs pages claires et organisées selon les actions principales du cycle de vie d'un certificat. Voici les interfaces disponibles :

— **Accueil (index)** : synthèse des actions possibles avec des liens directs.

— **Génération de CSR** : formulaire pour générer la clé privée et la demande de certificat.

— **Consultation des CSR** : liste des demandes en attente avec bouton de signature.

— **Liste des certificats** : affichage des certificats signés avec options de téléchargement ou de révocation.

— **Vérification** : interface pour vérifier l'état d'un certificat (valide, expiré, révoqué).

— **Dashboard** : visualisation graphique des statistiques globales.

Chaque interface adopte une **mise en page centrée**, des **icônes explicites** et des **boutons colorés** afin de guider efficacement l'utilisateur dans ses actions.

5.2 Navigation et ergonomie

La navigation repose sur une **barre de menu principale** fixe, accessible sur toutes les pages. Elle comporte les entrées suivantes :

Accueil	Dashboard	Générer CSR	Demandes CSR	Certificats	Vérification

Les interactions sont fluides et se réalisent en **quelques clics**. Chaque action déclenche des **messages dynamiques** (succès ou erreur) grâce au système `flash()` de Flask.

L'utilisation de formulaires simples, de **tableaux réactifs** et de **cartes Bootstrap** contribue à une expérience utilisateur claire, rapide et sans ambiguïté.

5.3 Intégration de Bootstrap et Chart.js

L'interface repose sur l'intégration conjointe de deux bibliothèques front-end majeures :

— **Bootstrap 5**, utilisé pour :

  — La mise en page responsive (grilles, conteneurs flexibles)

  — Les composants réutilisables (formulaires, boutons, alertes, cartes, tables)

  — L'adaptabilité sur appareils mobiles

— **Chart.js**, intégré dans la page `dashboard.html`, permet d'afficher dynamiquement les statistiques suivantes :

  — Nombre de certificats signés

  — Nombre de CSR en attente

  — Nombre de certificats révoqués

  — Nombre de certificats valides

Extrait de configuration Chart.js :

```
labels: ['Signés', 'CSR', 'Révoqués', 'Valides'],
data: [12, 4, 3, 9]
```

La combinaison de Bootstrap et Chart.js permet de proposer une interface à la fois **moderne**, **ergonomique** et **maintenable**.

6 Tests et Validation

Ce chapitre vise à démontrer que l'application développée fonctionne conformément aux exigences spécifiées dans le cahier des charges. Les tests réalisés couvrent les fonctionnalités principales de l'infrastructure PKI : génération, signature, révocation, vérification et visualisation.

6.1 Scénarios de test

Les scénarios suivants ont été définis et exécutés pour évaluer le bon fonctionnement des fonctionnalités critiques de l'application :

6.2 Scénarios de test

Les scénarios suivants ont été définis et exécutés pour évaluer le bon fonctionnement des fonctionnalités critiques de l'application :

— **Test 1 — Génération de CSR**
Entrée : nom de domaine valide
Résultat attendu : fichiers `.key.pem` et `.csr.pem` créés

— **Test 2 — Signature d'un CSR**
Entrée : CSR valide
Résultat attendu : certificat `.crt.pem` signé par la CA

— **Test 3 — Révocation d'un certificat**
Entrée : nom d'un certificat existant
Résultat attendu : ajout à la CRL sans suppression des entrées précédentes

— **Test 4 — Vérification d'un certificat actif**
Entrée : certificat valide
Résultat attendu : message confirmant la validité du certificat

— **Test 5 — Vérification d'un certificat révoqué**
Entrée : certificat déjà révoqué
Résultat attendu : détection correcte de la révocation

— **Test 6 — Tableau de bord**
Entrée : données simulées
Résultat attendu : affichage correct des statistiques dans les cartes et graphiques

6.3 Résultats attendus et obtenus

Fonctionnalité	**Résultat Attendu**	**Résultat Observé**	**Statut**
Génération de CSR	CSR créé avec clé privée	Conforme	Succès
Signature CSR	Certificat signé par la CA intermédiaire	Conforme	Succès
Révocation certificat	Ajout à la CRL sans effacer les anciennes entrées	Conforme	Succès
Vérification certificat	État déterminé selon validité et révocation	Conforme	Succès
Statistiques dashboard	Affichage correct avec Chart.js	Conforme	Succès

7 Conclusion Générale

7.1 Bilan du projet

Ce projet a permis la conception et le développement d'une application web simulant une infrastructure à clé publique (**PKI**) complète et fonctionnelle. L'intégration des bi-

bliothèques `cryptography`, `Flask` et `Bootstrap` a permis de concrétiser les principales fonctionnalités suivantes :

— Génération de paires de clés RSA privées/publiques

— Création et signature de demandes de certificats (CSR)

— Gestion et visualisation des certificats signés

— Révocation de certificats avec production d'une CRL dynamique

— Vérification de la validité et de l'authenticité des certificats

— Affichage statistique via des graphiques dynamiques (Chart.js)

Ce travail a aussi permis d'aborder les bonnes pratiques de sécurité numérique et de fournir une interface intuitive, même pour un public non spécialiste. Il a représenté une excellente opportunité de mise en œuvre des concepts de cryptographie appliquée, de développement web et d'architecture logicielle sécurisée.

7.2 Limites du travail

Malgré les résultats satisfaisants, plusieurs limitations ont été identifiées :

— L'application repose uniquement sur un **système de fichiers local**, sans base de données relationnelle.

— Le modèle PKI ne gère pas encore une hiérarchie complète de certification (*Root CA* + *Intermediate CA* multiples).

— Aucune **authentification ou autorisation utilisateur** n'est actuellement implémentée.

— La **liste de révocation (CRL)** est statique ; une évolution vers le protocole *OCSP* serait une amélioration notable.

Ces limites constituent autant de pistes d'amélioration pertinentes pour les développements futurs du projet, notamment dans une perspective de mise en production ou d'intégration à un système plus vaste.

En somme, le projet a rempli ses objectifs pédagogiques et techniques, tout en ouvrant la voie à des extensions intéressantes dans les domaines de la cybersécurité, de l'administration des certificats et des services web sécurisés.

A Annexes

A.1 Code source principal : `app_cryptography.py`

Listing 1 – Extrait de la fonction de génération de CSR

```
@app.route('/generate', methods=['GET', 'POST'])
def generate():
    if request.method == 'POST':
        cn = request.form.get('common_name')
        validity = int(request.form.get('validity', '365'))

        key = generate_key()
        csr = create_csr(key, cn)

        save_key_to_file(key, f"pki/final-certs/{cn}.key.pem")
        save_csr_to_file(csr, f"pki/final-certs/{cn}.csr.pem")

        flash("CSR g n r e avec succ s.", "success")
        return redirect(url_for('index'))

    return render_template('generate.html')
```

A.2 Exemple de certificat généré (format PEM)

Certificat PEM signé

```
-----BEGIN CERTIFICATE-----
MIIDZTCCAk2gAwIBAgIUDm3RZu5V...
...
A9ixxK2dE9gdo3LjaoUk0nY=
-----END CERTIFICATE-----
```

Ce certificat a été signé par l'autorité intermédiaire. Il contient la clé publique, l'identité du titulaire et les extensions nécessaires pour son usage dans une infrastructure de confiance.

A.3 Captures d'écran de l'interface utilisateur

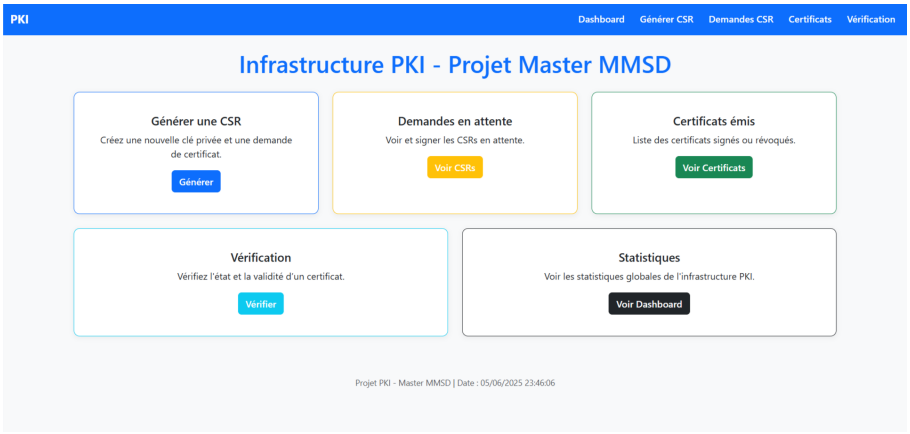


FIGURE 1 – Interface

FIGURE 2 – Interface de génération d'un CSR

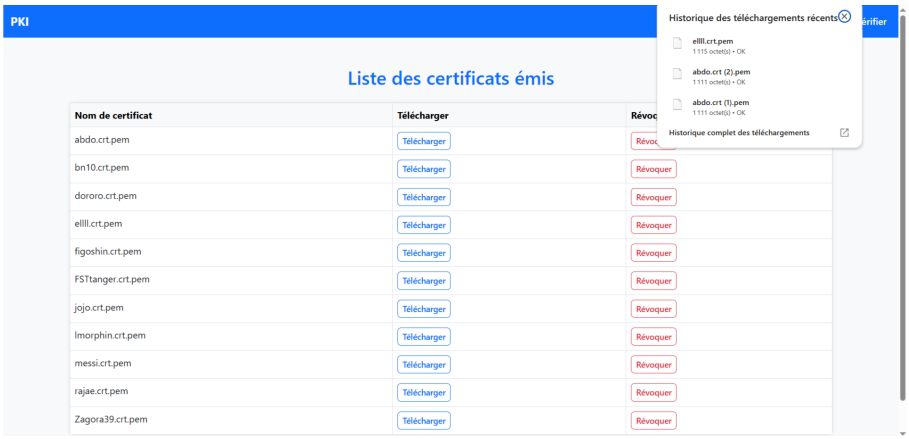


FIGURE 3 – Liste des certificats émis

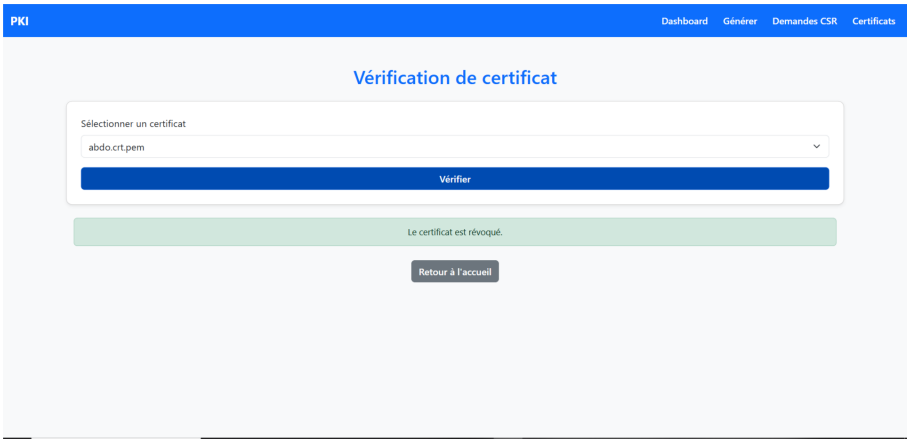


FIGURE 4 – Vérification d'un certificat

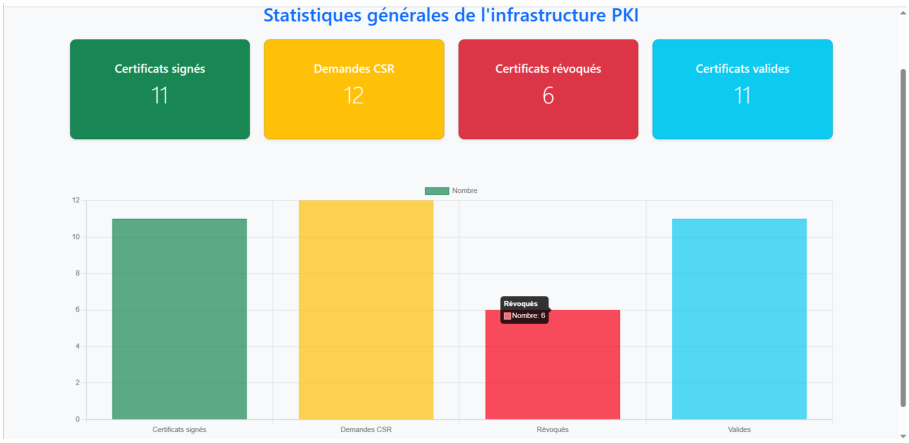


FIGURE 5 – Dashboard statistique

Ces éléments visuels permettent d'illustrer concrètement le fonctionnement et l'ergonomie de l'application web PKI développée.