

DevSecOps : Programmation algorithme sécurisée

Kick-off Projet - Application de gestion pour e-commerçants

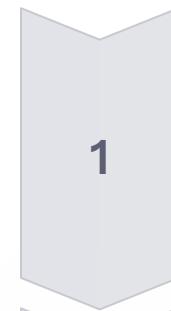
Yannis Benbourahla, Co-dirigeant Novicore

school@novicore.fr





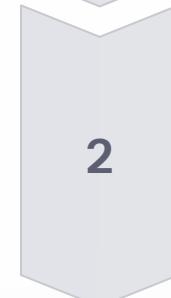
Qu'est-ce que DevSecOps ?



Approche Traditionnelle

Code → Test → Déploie

La sécurité arrive trop tard, nécessitant des corrections coûteuses et chronophages après développement.



Approche DevSecOps

Code sécurisé → Test (sécurité incluse) → Déploie sécurisé

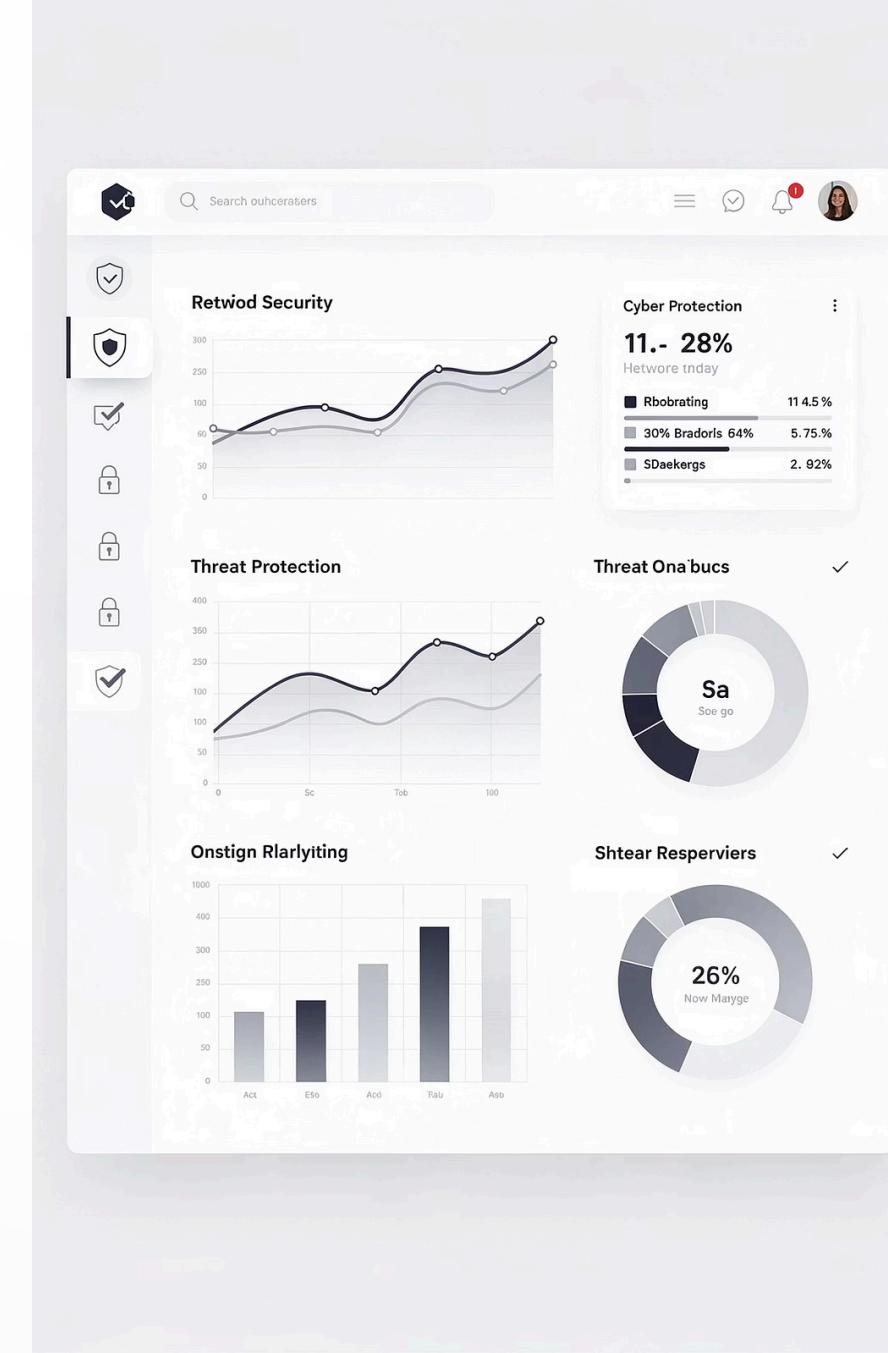
Intégration de la sécurité dès la conception, réduisant les vulnérabilités et accélérant le déploiement.

"Sécurité dès le départ, pas à la fin"

Projet Global : Une Application Complète et Sécurisée

Ce projet est conçu pour approfondir l'apprentissage de Python en intégrant des aspects essentiels de la programmation moderne, de la sécurité des applications et de l'expérience utilisateur.

Vous développerez une application de gestion de produits pour commerçants, avec lecture de fichiers, ajout, tri et recherche. Le projet évoluera vers une version multi-utilisateurs avec sécurisation des données (détection de mots de passe compromis via API). Une interface graphique (Tkinter ou PyQt), la gestion des commandes et l'affichage de statistiques (Matplotlib, Seaborn) viendront enrichir l'outil.





Les 3 Concepts Clés du Projet



Hachage & Salage

Transformation irréversible des mots de passe avec **SHA-256** combinée à un **salt aléatoire unique** pour chaque utilisateur. Cette approche garantit qu'un même mot de passe génère des empreintes différentes, empêchant les attaques par rainbow tables.



Vérification Locale

Contrôle des mots de passe contre une base de **plus de 600 millions** de credentials compromis, disponible gratuitement. Cette vérification locale protège la confidentialité tout en assurant une sécurité maximale.



Architecture Modulaire

Security by Design : conception d'une architecture en couches avec séparation des responsabilités, principe du moindre privilège et isolation des composants critiques pour une sécurité robuste et maintenable.

Architecture Globale

Notre application suit une architecture en couches qui sépare clairement les responsabilités et facilite la maintenance, les tests et l'évolution du système.



- Cette architecture permet d'isoler les composants sensibles et d'appliquer des contrôles de sécurité à chaque niveau, conformément aux principes DevSecOps.

Les 7 Modules à Développer

Le projet est structuré en 8 modules interconnectés qui couvrent l'ensemble des compétences DevSecOps requises pour développer une application professionnelle sécurisée.

#	Module	Description	Priorité
1	Gestion des produits	CRUD complet en CSV	Haute
2	Authentification sécurisée	Hachage SHA-256 + Salt	Critique
3	Vérification mots de passe	Contrôle base compromis	Critique
4	Interface graphique	GUI Tkinter/PyQt	Haute
5	Commandes & Statistiques	Gestion + Visualisations	Moyenne
6	API REST	Flask/FastAPI endpoints	Moyenne
7	Sécurité & Audit	Bandit, Pylint, Safety	Critique

Module 1 - Gestion des Produits

Qu'est-ce que c'est ?

Système **CRUD** (Create, Read, Update, Delete) permettant de gérer l'inventaire complet des produits via une interface en ligne de commande.

Livrables

- Code Python structuré
- Suite de tests unitaires
- Fichier CSV exemple
- Documentation utilisateur

Ce qu'il faut faire

01

CLI avec menu interactif pour navigation intuitive

02

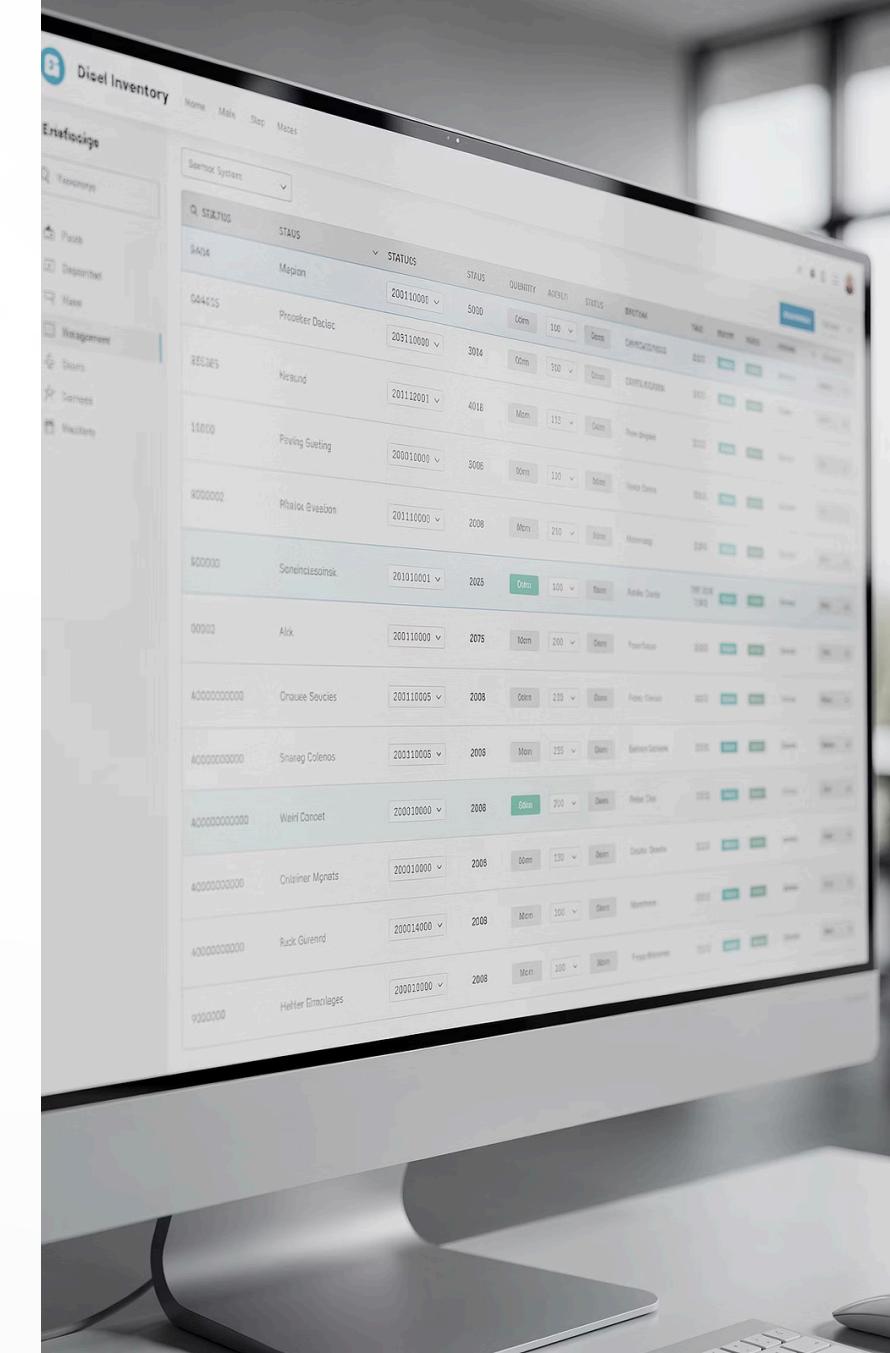
Structurer les données : id, nom, description, prix, quantité

03

Opérations de lecture/écriture CSV avec gestion d'erreurs

04

Tester les 4 opérations CRUD avec cas limites





Module 2 - Authentification Sécurisée

Qu'est-ce que c'est ?

Système complet de **login** et **enregistrement** utilisant les meilleures pratiques cryptographiques pour protéger les identifiants utilisateurs.

Composants de sécurité

- **Hachage SHA-256** : algorithme cryptographique robuste
- **Salt aléatoire** : unique pour chaque utilisateur
- **Base utilisateurs** : stockage sécurisé (CSV ou dict)
- **Logs détaillés** : traçabilité complète des accès

Fonctionnalités requises

1 Création de compte

Validation des mots de passe, génération du salt, stockage sécurisé

2 Connexion

Vérification des credentials hashés avec comparaison temporelle constante

3 Audit de sécurité

Enregistrement des tentatives échouées, détection d'anomalies

Module 3 - Vérification Mots de Passe Compromis

Protégez vos utilisateurs en vérifiant si leurs mots de passe figurent parmi les **600+ millions de credentials compromis** dans des fuites de données publiques.

API api.pwnedpasswords.com

Mode online – Toujours à jour

- API gratuite sans clé
- Base actualisée en temps réel
- Aucun stockage local requis
- Protection par k-anonymité

Inconvénient : Nécessite connexion Internet

Processus de vérification

1

Hacher le mot de passe

2

Rechercher dans la base

3

Alerter si compromis

Module 4 - Interface Graphique

Qu'est-ce que c'est ?

Interface graphique utilisateur (GUI) professionnelle développée avec **Tkinter** ou **PyQt5**, offrant une expérience utilisateur fluide et intuitive pour gérer l'ensemble de l'application.

Fonctionnalités principales

1

Fenêtre principale

Layout responsive avec navigation claire et moderne

2

Actions produits

Boutons : Ajouter, Voir, Modifier, Supprimer

3

Liste interactive

Affichage dynamique avec tri et filtres

4

Détails produit

Vue complète au clic avec édition inline

Éléments de sécurité

- **Login intégré**

Écran d'authentification sécurisé au démarrage

- **Alertes visuelles**

Notifications de sécurité avec codes couleur

- **Session utilisateur**

Gestion du timeout et déconnexion propre

- **Validation input**

Contrôles en temps réel des données saisies

Module 5 - Commandes & Statistiques

Partie A - Gestion des Commandes

Système complet de gestion du cycle de vie des commandes avec validation des données et contrôle d'intégrité.

- **CRUD complet** : Create, Read, Update, Delete
- **Vérification stock** : Contrôle des quantités disponibles
- **Mise à jour automatique** : Décrément stock post-commande
- **Validation données** : Contrôles d'intégrité et cohérence

Partie B - Statistiques & Visualisations

Tableaux de bord analytiques avec graphiques interactifs pour piloter l'activité commerciale.



Produits vendus

Volume par référence



Revenus

CA total et marges



Évolution ventes

Courbes temporelles



Top 5 produits

Meilleures ventes



Clients actifs

Base utilisateurs

Module 6 - API REST Maison

Développez une **API RESTful professionnelle** avec Flask ou FastAPI permettant l'accès programmatique aux ressources de l'application via des endpoints HTTP standardisés.

Endpoints à implémenter

Méthode	Endpoint	Fonction
GET	/api/products	Liste tous les produits avec pagination
GET	/api/products/:id	Détails d'un produit spécifique
POST	/api/products	Créer un nouveau produit (auth requise)
PUT	/api/products/:id	Modifier un produit existant (auth requise)
DELETE	/api/products/:id	Supprimer un produit (auth requise)
POST	/api/auth/login	Authentification et génération token JWT
GET/POST	/api/orders	Gestion des commandes
GET	/api/stats	Statistiques agrégées

Sécurité

Authentification par **token JWT**, validation des entrées, rate limiting, CORS configuré

Logging

Traçabilité complète des requêtes, erreurs détaillées, métriques de performance

Tests

Suite de tests automatisés pour chaque endpoint avec différents scénarios

Module 7 - Sécurité & Audit

Qu'est-ce que c'est ?

Analyse approfondie du code source pour identifier et corriger les **vulnérabilités de sécurité**, améliorer la qualité du code et vérifier les dépendances contre les CVE connus.



Bandit

Scanner spécialisé dans la détection des **failles de sécurité Python** : injections, déserialisations dangereuses, cryptographie faible, hardcoded secrets



Pylint

Analyse de la **qualité du code** : conventions PEP8, complexité cyclomatique, code smell, bugs potentiels, maintenabilité



Safety

Vérification des **vulnérabilités des dépendances** : détection des packages avec CVE, mises à jour recommandées, alertes de sécurité

Checklist de sécurité



Secrets Management

Vérifier l'absence de credentials, tokens, clés API en dur dans le code



Input Validation

Tester tous les cas limites, injections SQL/XSS, dépassements de buffer



Error Handling

S'assurer que les messages d'erreur ne révèlent pas d'informations sensibles

À Quoi Ça Sert ?

Ce projet DevSecOps vous permet d'acquérir un **ensemble complet de compétences professionnelles** recherchées par les entreprises technologiques modernes.

Domaine	Compétences Acquises
Sécurité	Hachage cryptographique, salage, vérification credentials compromis, principe du moindre privilège, audit de sécurité, gestion des secrets
Python	Structures de données avancées, programmation modulaire, gestion d'erreurs robuste, tests unitaires, programmation orientée objet
Git	Gestion de branches, commits sémantiques, pull requests, collaboration en équipe, résolution de conflits, workflows GitFlow
BDD	Manipulation CSV, persistence des données, transactions, intégrité référentielle, optimisation des requêtes
GUI	Tkinter/PyQt, event handling, layouts responsive, binding de données, patterns MVC
API	Architecture REST, HTTP/HTTPS, authentification par tokens, documentation OpenAPI, versioning d'API
DevOps	Logging structuré, monitoring applicatif, audit automatisé, quality assurance, CI/CD basics
Professionnel	Documentation technique, communication claire, présentation de démo, travail d'équipe agile

Stack Technologique

L'ensemble des technologies et outils que vous utiliserez pour construire cette application DevSecOps professionnelle.



Python 3

Langage principal pour la logique métier, manipulation de données et orchestration des modules



Git/GitHub

Versioning du code, collaboration en équipe, gestion des branches et historique complet



Hashlib

Bibliothèque standard Python pour le hachage cryptographique sécurisé (SHA-256)



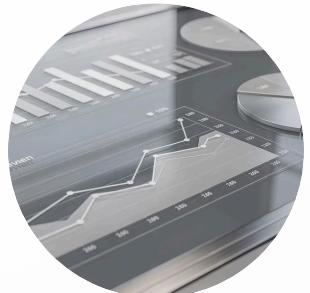
CSV

Format de stockage des données simple, portable et facilement manipulable



PyQt/Tkinter

Frameworks pour créer des interfaces graphiques utilisateur professionnelles et responsive



Matplotlib/Seaborn

Bibliothèques de visualisation pour générer des graphiques et tableaux de bord analytiques



Flask/FastAPI

Frameworks web légers pour construire des API REST modernes et performantes



Bandit/Pylint/Safety

Suite d'outils d'analyse statique pour garantir qualité du code et sécurité

Ressources Clés

Une sélection de ressources essentielles pour vous accompagner tout au long du développement de votre projet DevSecOps.

	Documentation Officielle Python https://docs.python.org/3/ Référence complète du langage, bibliothèque standard, tutoriels et guides avancés pour maîtriser Python 3
	Have I Been Pwned - Passwords API https://api.pwnedpasswords.com/range/5BAA6 Accès à la base de données de mots de passe compromis et documentation de l'API pour vérification sécurisée
	PyQt5 Documentation https://www.riverbankcomputing.com/static/Docs/PyQt5/ Guide complet pour créer des interfaces graphiques modernes et professionnelles avec PyQt5
	Flask Framework https://flask.palletsprojects.com/ Documentation officielle Flask pour développer des applications web et API REST légères et extensibles

Livres Recommandés

- "Automate the Boring Stuff with Python"
- "Python Crash Course"
- "Clean Code in Python"

Tutoriels Essentiels

- "How To Hash Passwords In Python"
- "Real Python: API Requests"
- "Building REST APIs with Flask"

Outils en Ligne

- Python Tutor (visualisation)
- Repl.it (IDE en ligne)
- GitHub Learning Lab

Liberté d'Organisation

"Votre projet, vos règles, votre rythme"

Ce projet vous offre une **flexibilité totale dans l'organisation** de votre travail. Il n'y a pas de timeline rigide imposée par semaine - vous êtes maîtres de votre planning.

Organisation Libre

Gérez votre équipe de **4 personnes** comme vous le souhaitez : approche agile avec sprints, méthode waterfall séquentielle, ou hybride selon vos préférences

7 Modules à Livrer

Objectif clair : compléter les 7 modules définis. La seule contrainte est la qualité du livrable final, pas le chemin pour y parvenir

Approche Flexible

Les modules peuvent être développés **en parallèle** par différents membres ou **séquentiellement** selon les dépendances identifiées

Conseils d'organisation

- Identifiez les dépendances entre modules (ex: Auth avant GUI) → Utilisez Git/Github efficacement pour la collaboration
- Répartissez les tâches selon les compétences de chacun → Priorisez les modules critiques (Auth, Sécurité)
- Définissez des points de synchronisation réguliers → Testez continuellement, ne reportez pas à la fin

- Focus principal :** Qualité et sécurité du code. Prenez le temps nécessaire pour bien faire les choses. Un code sécurisé et bien documenté vaut mieux qu'une livraison rapide mais approximative.

Points Clés à Retenir

1. Hashing ≠ Encryption

Le hachage est une **transformation à sens unique** irréversible. L'encryption est bidirectionnelle. Pour les mots de passe, toujours hacher, jamais encrypter.

2. Démo Live : Hash en Action

Testez dans Python REPL :

```
import  
hashlib;  
hashlib.sha256(b"password").hexdigest()
```

Observez comment le même input donne toujours le même hash.

3. Enjeux Réels

Amendes RGPD jusqu'à 4% du CA mondial, **réputation** dégradée irrémédiablement, **perte de clients** et de confiance. La sécurité n'est pas optionnelle.

4. Ressources Disponibles

Documentation Python, tutoriels, forums Stack Overflow, GitHub. **Explorez librement**, apprenez par la pratique, n'hésitez pas à chercher.

5. Questions Bienvenues

Aucune question n'est stupide. **Posez-les tôt** plutôt que de bloquer.

6. Projet Professionnel

Ce n'est pas un exercice académique. C'est un **vrai projet** que vous pourriez présenter en entretien. **Rendez-le professionnel** : code propre, docs complètes, démo soignée.

Rappel Sécurité

- Toujours saler les hash
- Jamais de secrets en dur
- Logger les accès

Bonnes Pratiques

- Tests unitaires systématiques
- Code review en équipe
- Commits atomiques clairs

Livrables

- Code fonctionnel testé
- Documentation complète

1 Application Professionnelle DevSecOps

"Sécurité dès le départ - Pas à la fin"

Vous avez maintenant toutes les clés pour réussir ce projet ambitieux. Rappelez-vous que la sécurité n'est pas une fonctionnalité qu'on ajoute à la fin, c'est un **état d'esprit** qui doit guider chaque ligne de code que vous écrivez.

7

Modules à maîtriser

De la gestion de produits à l'audit de sécurité

600M+

Mots de passe compromis

Base de données pour protéger vos utilisateurs

100%

Approche DevSecOps

Sécurité intégrée dès la conception

Prochaines Étapes

1. Former vos équipes de 4
2. Choisir votre approche d'organisation
3. Cloner le repository de démarrage
4. Commencer par les modules prioritaires

Bonne chance et excellent développement ! 