Rapport de projet **GAME ATLAS**



ligne courte

SÉBASTIEN HENIQUE  
28 avril 2025

**Sommaire**

**Liste des compétences du référentiel couvertes par le projet 3**

**Résumé du projet en anglais 6**

**Présentation du projet 7**

**Cahier des charges 8**

I. Présentation d’ensemble du projet 8

II. Description de la charte graphique 9

III. Prestations attendues et modalités 10

**Gestion du projet 11**

I. Planning et suivi 11

II. Environnement humain et technique 12

III. Objectifs de qualité 14

**Spécifications fonctionnelles du projet 15**

**Spécifications techniques du projet 16**

**Réalisations 17**

I. Exemple 1 17

II. Exemple 2 20

III. Exemple 3 23

**Présentation d’un jeu d’essai 26**

I. Données en entrée 26

II. Données attendues 27

III. Données obtenues 28

**Veille sur les vulnérabilités de sécurité 29**

I. Besoin d’information 29

II. Mots clés de recherche 30

III. Liste des sites retournés 31

IV. Critères de sélection des sites 32

V. Vulnérabilités trouvées 33

VI. Failles potentielles corrigées 34

**Pratiques DevOps 35**

1. Tests Unitaires 35
2. Tests de non-régression & tests end-to-end
3. Pipeline Github CI/CD
4. Conteneurisation Docker

**Description d’une situation de travail ayant nécessité une recherche 35**

I. Besoin d’information 35

II. Mots clés de recherche 36

III. Liste des sites retournés 37

IV. Critères de sélection des sites 38

V. Solution et mise en œuvre 39

**Conclusion 40**

**Annexes 41**

**Liste des compétences du référentiel couvertes par le projet**

**Activité type 1 « Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité »**

* Maquetter une application

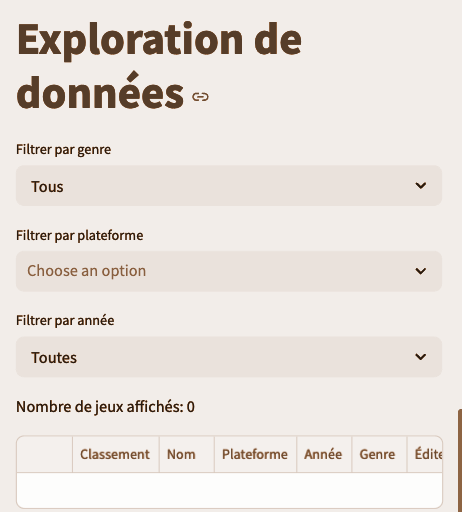
Aperçu écran d'accueil de l’application:





Exploration des données :

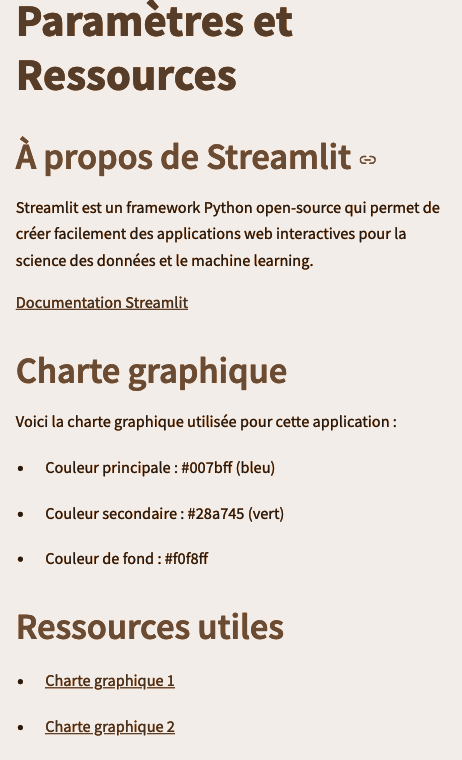




Affiche les différents logos des plateformes de jeux vidéo :

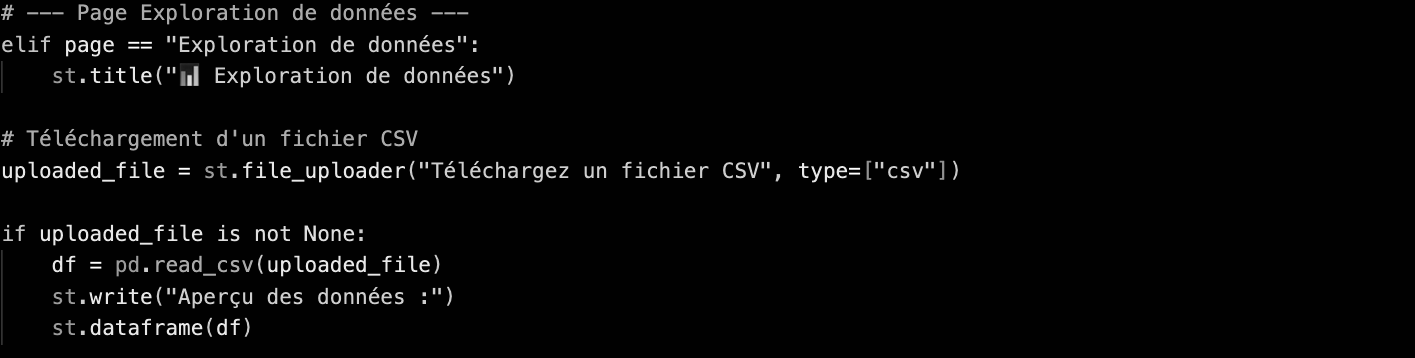


Affiche les paramètres et ressources :



* Développer des composants d’accès aux données

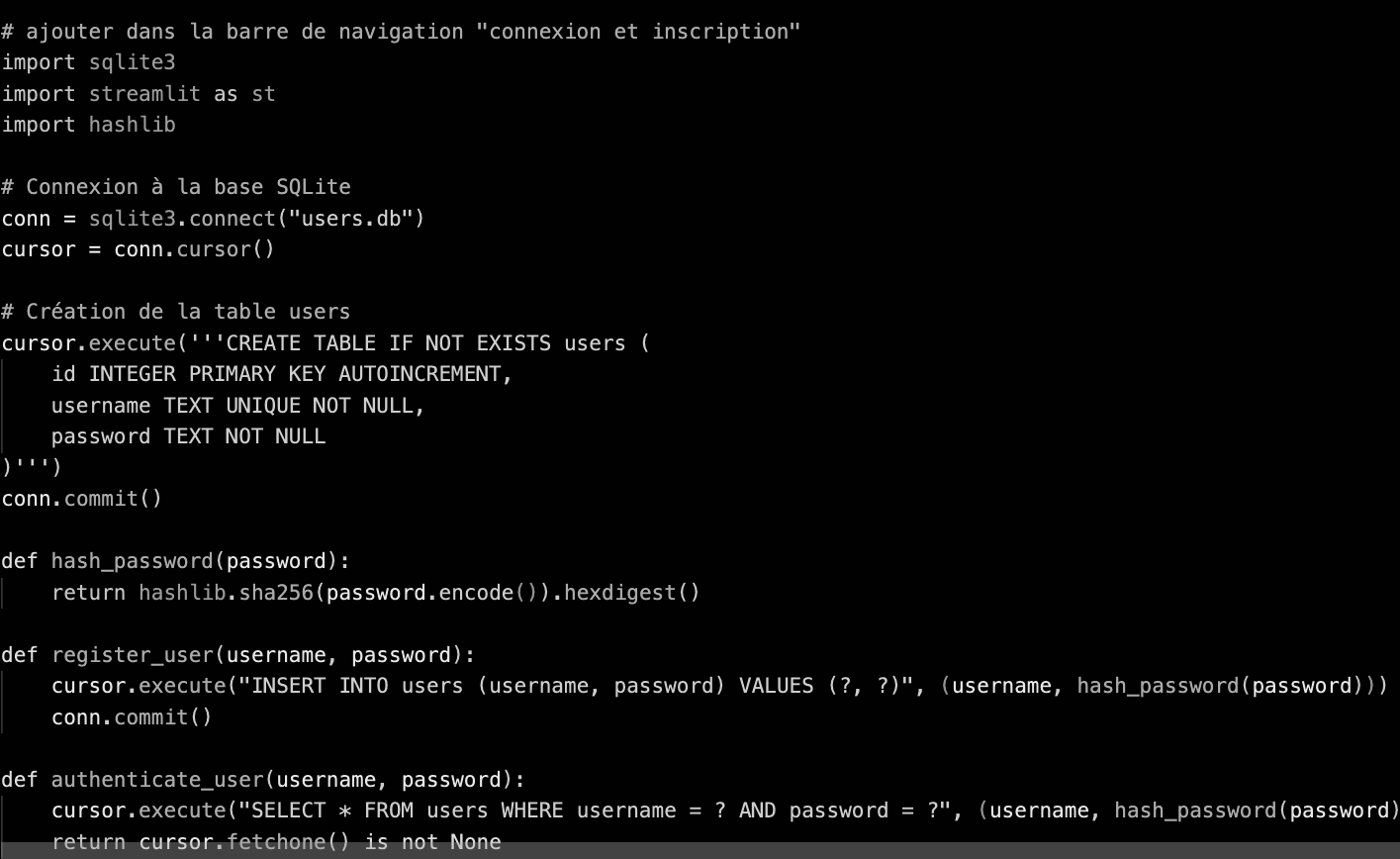
Extrait de code ou je charge un fichier CSV :



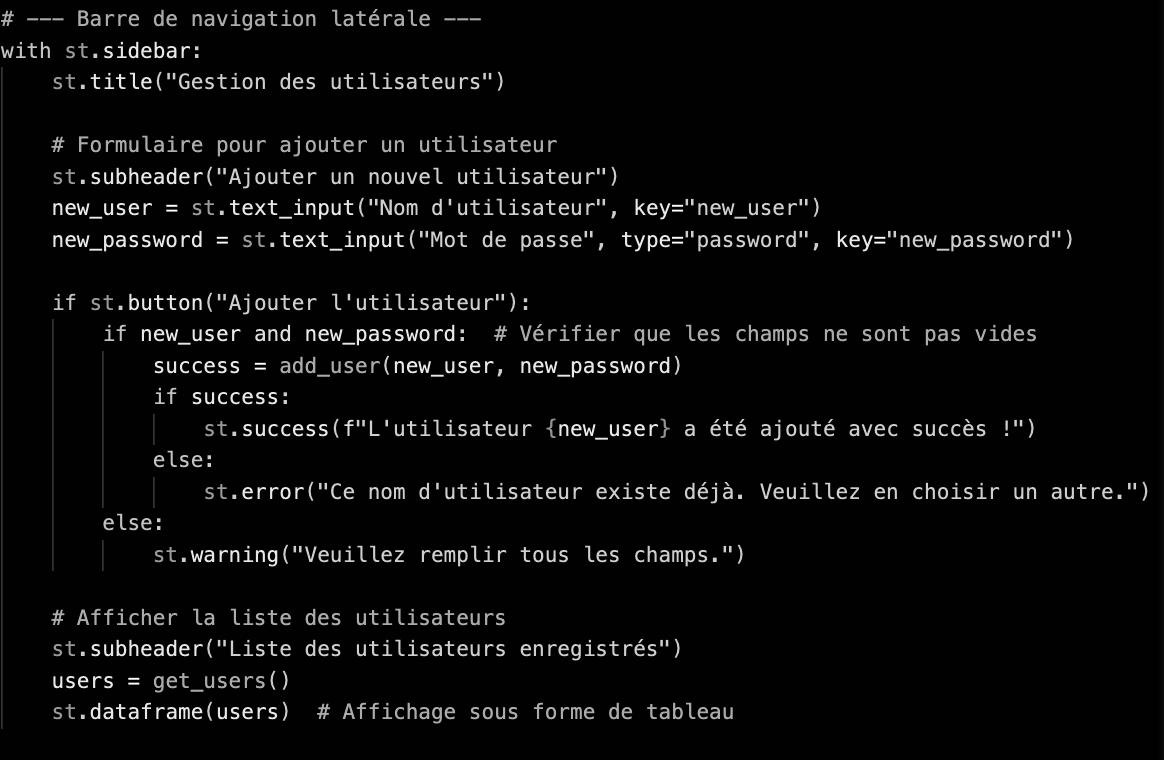
Extrait de code pour filtrer les données et afficher les ventes :



Extrait de code ou je créer la tables des utilisateurs :

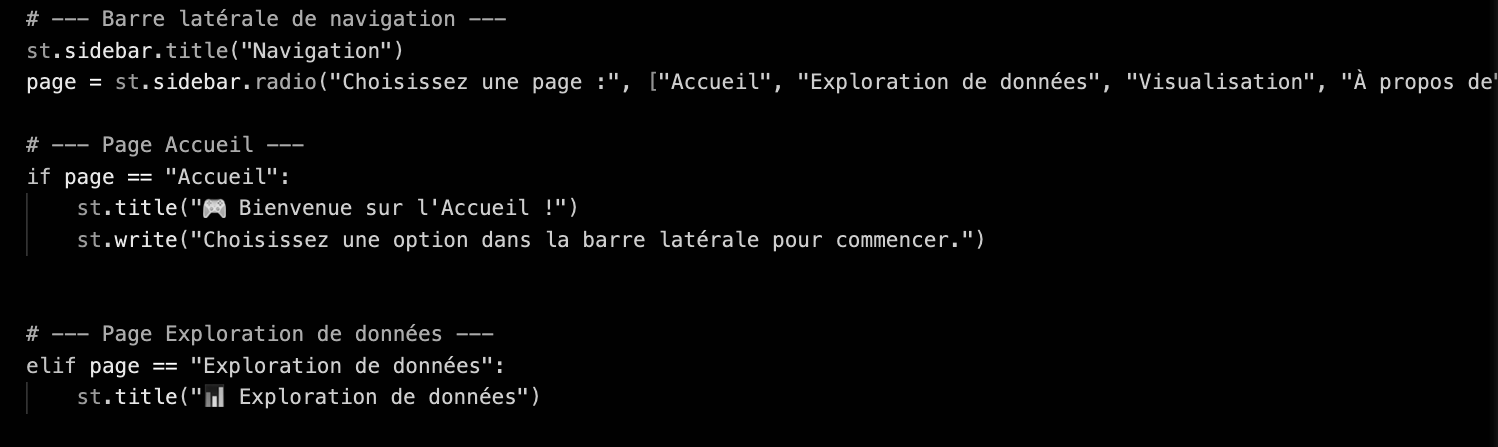


Extrait de code pour insérer un nouvel utilisateur :

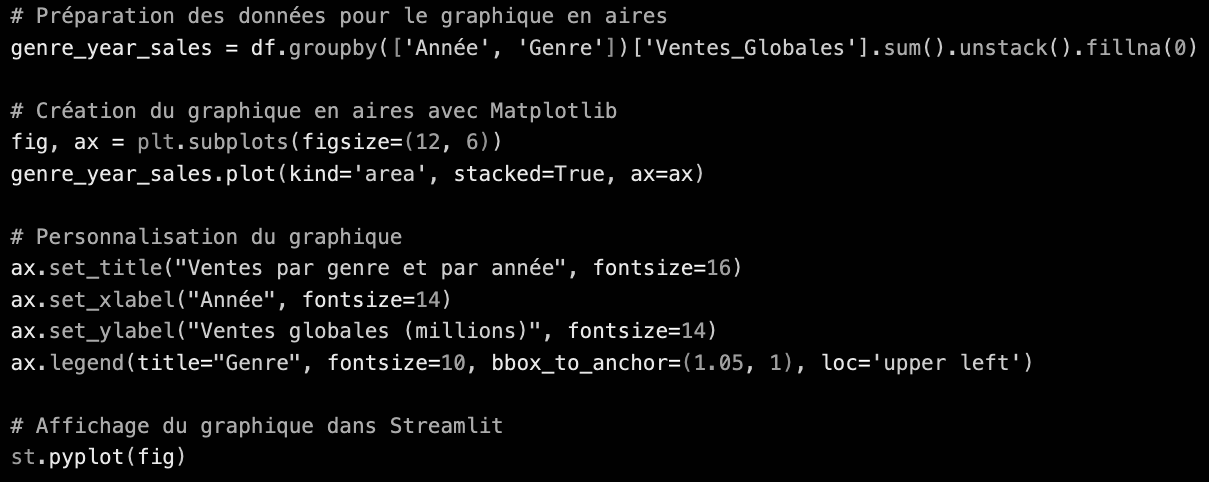


* Développer la partie *front-end* d’une interface utilisateur web

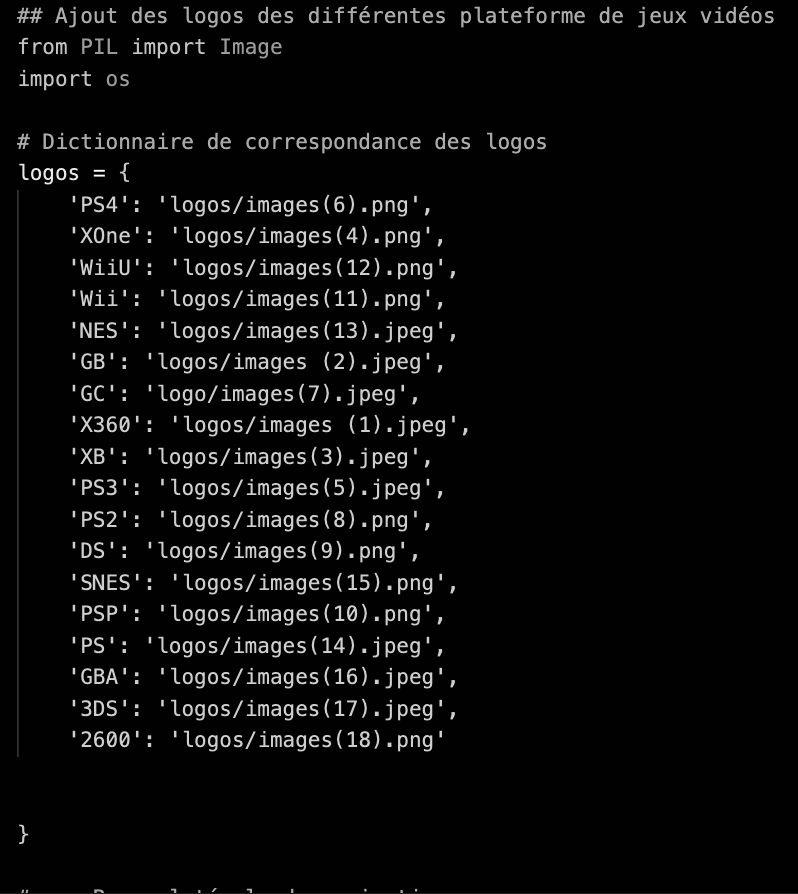
Extrait de code montrant la création de la barre de navigation :

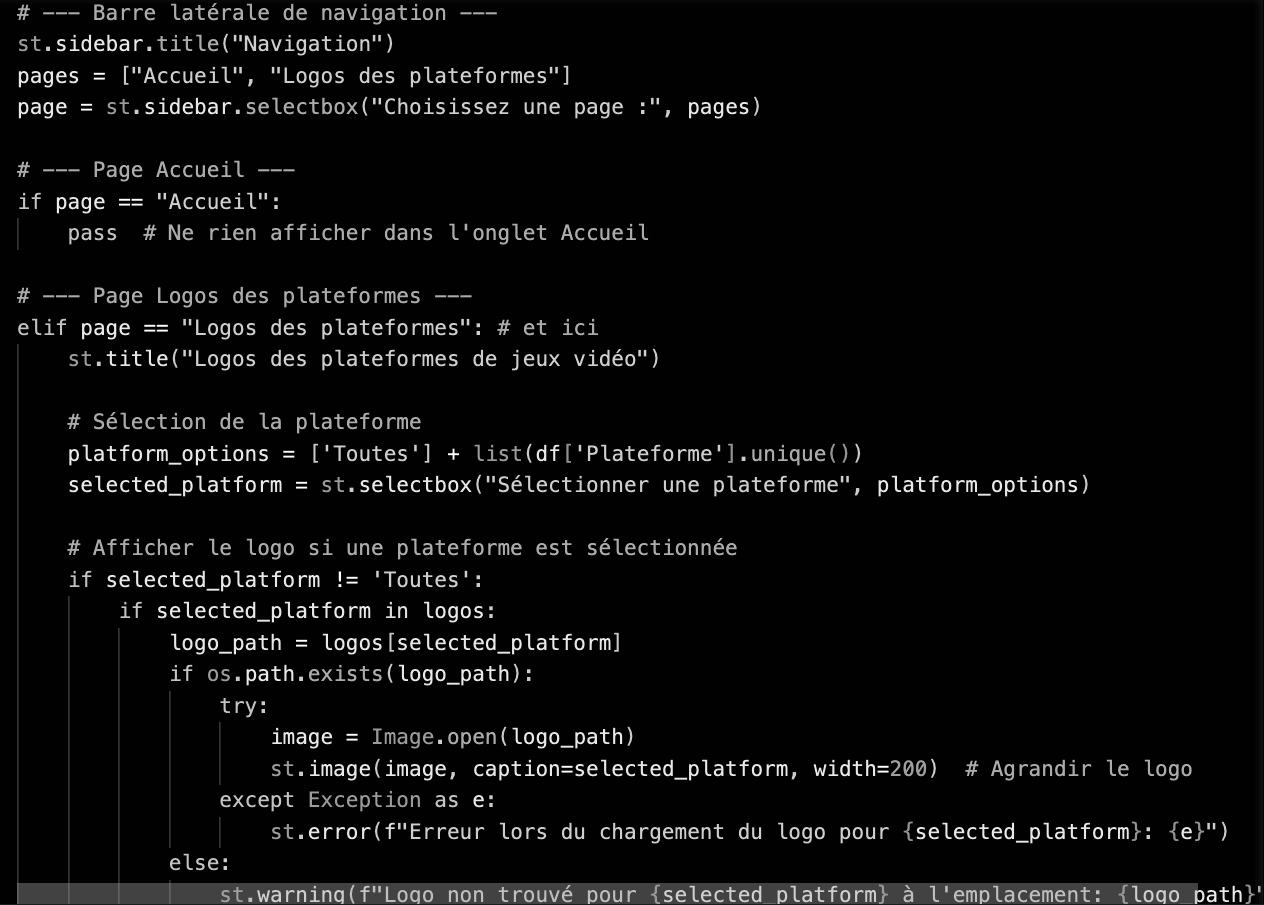


Extrait de code qui montre un graphique interactif :

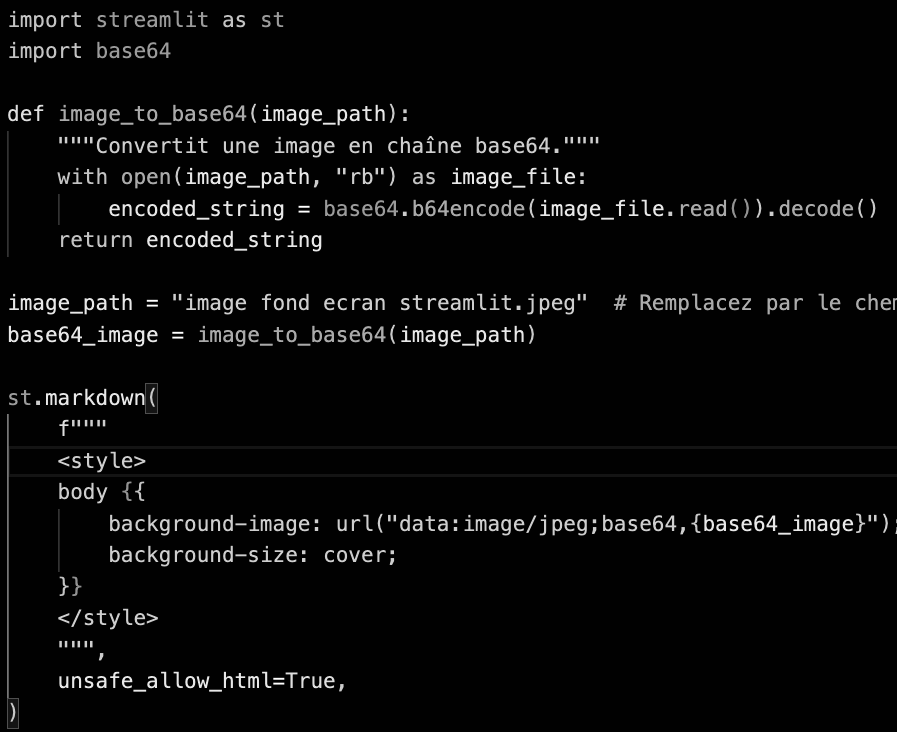


Extrait de code pour visualiser les logos des plateformes de jeux vidéo :



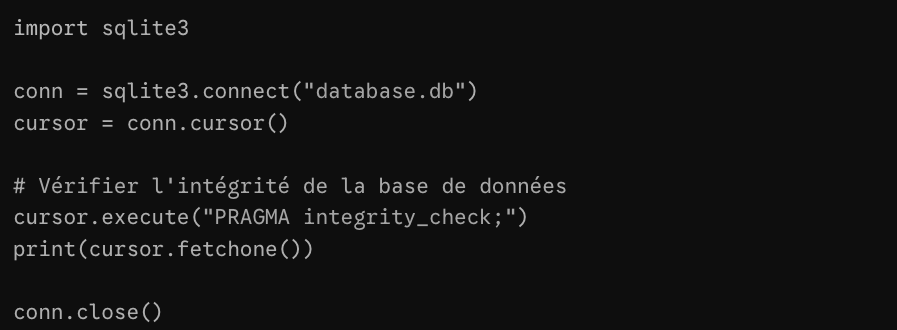


Extrait de code qui ajoute une image en fond d’écran à l’application :

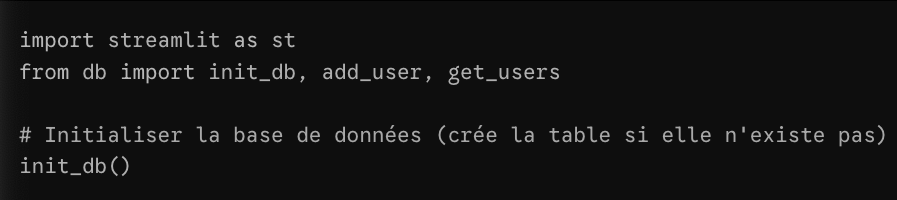


* Développer la partie *back-end* d’une interface utilisateur web

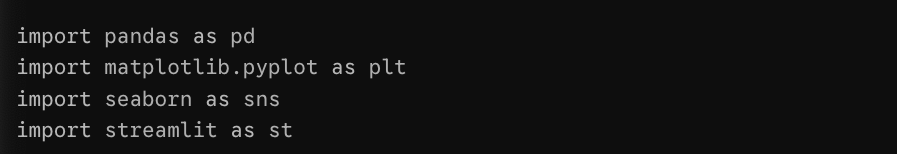
Extrait de code de gestion de la base de données SQLite :



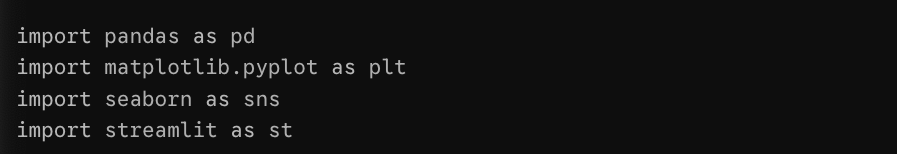
Extrait de code des fonctions de gestion des utilisateurs :



Extrait de code pour le traitement des données pour les visualisations :



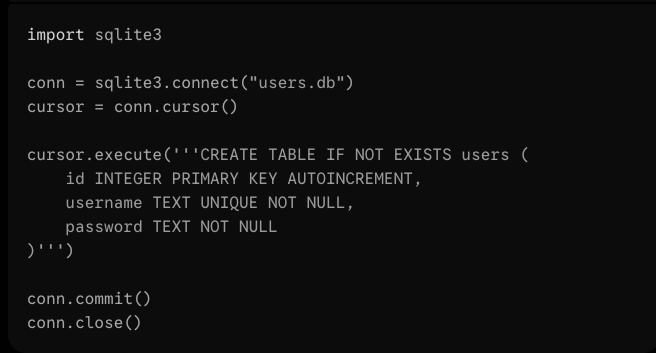
Extrait de code pour le filtrage des données :



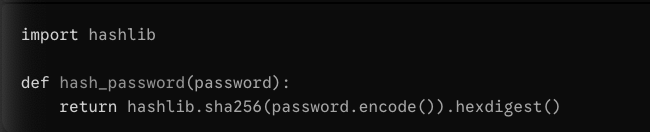
**Activité type 2 « Concevoir et développer la persistance des données en intégrant les recommandations de sécurité »**

* Mettre en place une base de données

Exemple de capture d’écran (création de la table **users**) :



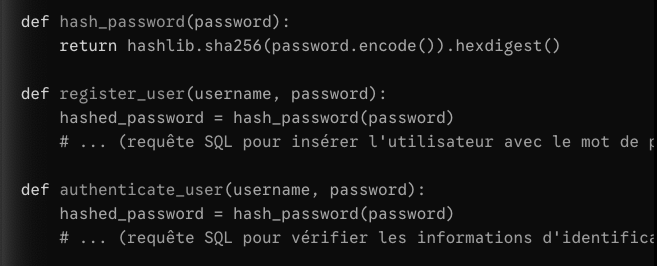
Exemple de capture d’écran (hachage des mots de passe) :



**Activité type 3 « Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité »**

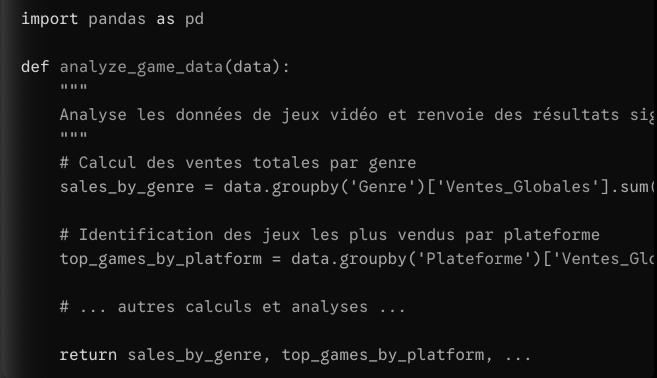
* Concevoir une application

Capture d’écran du code de la couche métier (gestion des utilisateurs) :



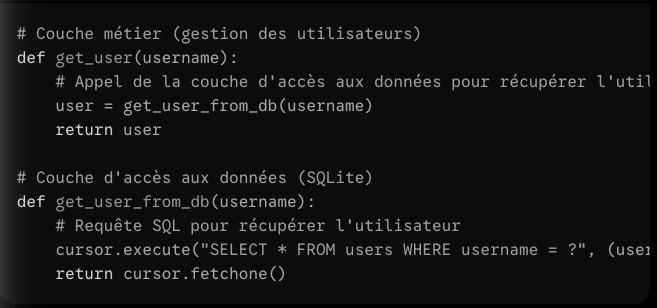
* Développer des composants métier

Capture d’écran de la fonction **analyse\_game\_data** (logique métier) :

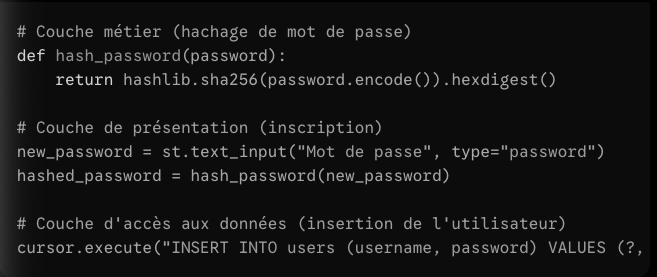


* Construire une application organisée en couches

Capture d'écran de l’interaction entre la couche métier et la couche d’accès aux données :



Capture d’écran de l’utilisation de composants réutilisables entre les couches :



**Résumé du projet en anglais**

Project Summary: Video Game Sales Data Analysis Streamlit Application

This Streamlit application provides a user-friendly interface to analyze and visualize video game sales data. Users can upload CSV files, filter data by genre, platform, and year, and explore descriptive statistics and visualizations. The application utilizes Pandas for data manipulation and Streamlit for interactive visualizations, empowering users to efficiently gain insights into sales trends and patterns.

**Présentation du projet**

Ce projet consiste en une application Streamlit complète et interactive conçue pour l’analyse et la visualisation des données de ventes de jeux vidéo. Il offre une interface utilisateur conviviale permettant aux utilisateurs de télécharger des fichiers CSV, de filtrer les données par divers critères, et d’explorer des statistiques descriptives et des visualisations graphiques.

**Cahier des charges**

1. **Présentation d’ensemble du projet**

Fonctionnalités principales:

Chargement de données flexible :

* Les utilisateurs peuvent télécharger leurs propres fichiers CSV contenant des données de ventes de jeux vidéo.

Exploration de données interactive :

* Affichage des données sous forme de tableaux interactifs
* Calcul et affichage des statistiques descriptives (moyenne, médiane, etc…).
* Identification et affichage des valeurs manquantes.
* Filtrage interactif des données par genre, plateforme, année et plage de ventes.

Visualisations graphiques riches :

* Graphiques en barres pour comparer les ventes par genre et par éditeur.
* Graphiques linéaires pour visualiser l’évolution des ventes au fil du temps
* Graphiques en aires pour montrer la répartition des ventes par genre et par année.
* Graphiques en camembert pour visualiser les plateformes les plus utilisées.
* Graphique en violon pour les ventes par plateforme
* Diagramme de Sankey pour visualiser les flux de ventes entre genres, plateformes et éditeurs.
* Carte interactive des ventes par régions.
* Graphique de série chronologique interactif des ventes par genre.

Personnalisation de l’interface :

* Utilisation de CSS pour personnaliser l’apparence des titres, sous-titres et autres éléments de l’application.
* Ajout de logo pour les différentes plateformes.
* Gestion d’une base de données d’utilisateur, et de leurs authentifications.

Navigation conviviale :

* Barre de navigation latérale pour un accès facile aux différentes pages de l’application

Gestion des utilisateurs :

* Une base de données d’utilisateurs à été mise en place avec la possibilité d’en ajouter et de les consulter
* Un système d’authentification à été mis en place, avec inscription et connexion

Technologies utilisées :

* Streamlit: pour la création de l’interface utilisateur interactive.
* Pandas : pour la manipulation et l’analyse des données.
* Matplotlib et Seaborn : pour les visualisations graphiques statiques.
* Plotly express et plotly graph objets : pour les graphiques interactifs.
* SQLite : pour la base de données.
* CSS : pour la personnalisation de l’apparence.
* Pillow(PIL) : pour le traitement des images.

Objectifs du projet :

* Fournir un outil d’analyse de données de ventes de jeux vidéo accessible et facile à utiliser.
* Permettre aux utilisateurs de découvrir des tendances et des modèles dans les données de ventes.
* Offrir une expérience de visualisation de données riche et interactive.
* Ce projet est conçu pour être utile aux passionnés de jeux vidéo, aux analystes de données et à toute personne intéressée par l’exploration des données de ventes de jeux vidéo.

**II. Description de la charte graphique**

La charte graphique de cette application Streamlit est conçue pour offrir une expérience utilisateur à la fois agréable et intuitive, tout en mettant en valeur les données de ventes de jeux vidéo de manière claire et efficace. Voici une description détaillé de ses principaux éléments :

Couleurs :

Couleur de fond :

* Une couleur claire et douce (#f0f8ff, un bleu très clair) est utilisée pour le fond de l’application, créant une atmosphère aérée et facilitant la lecture des données.

Couleurs des titres :

* Les titres principaux (h1) sont mis en évidence avec un bleu vif (#007bff), attirant l’attention sur les sections importantes.
* Les sous-titres (h2 et h3) dans la barre de navigation sont présentés dans un vert agréable ( #28a745), offrant une distinction visuelle tout en restant harmonieux.
* Les paragraphes de sous-titres sont eux affiché en gris (#6c757d).

Couleur des boutons :

* Les boutons sont conçus avec un vert vibrant (#4CAF50) et un texte blanc, assurant une bonne visibilité et incitant à l’action.

Palette de couleurs pour les graphiques :

* Des palettes de couleurs variées sont utilisées pour les graphiques ( Seaborn, Plotly), telles que “viridis” et des couleurs personnalisées, pour différencier les catégories de données et rendre les visualisations attrayantes.
* Les diagrammes de Sankey ont des couleurs personnalisées pour différencier chaque couche de données.
* Les cartes choroplèthes utilisent un dégradé de couleur en fonction de la valeur représentée.

Typographie :

* Bien que le code fourni ne spécifie pas explicitement les polices, Streamlit utilise par défaut des polices sans-serif pour une lisibilité optimale sur les écrans.
* Les tailles de police sont adaptées pour les titres, sous-titres et textes, assurant une hiérarchie visuelle claire.

Éléments graphiques :

Logos de plateformes :

* Des logos des différentes plateformes de jeux vidéo sont intégrés pour ajouter une dimension visuelle et contextuelle à l’application.

Graphiques interactifs :

* Une variété de graphiques interactifs ( barres, lignes, camemberts, cartes, etc..) sont utilisés pour permettre aux utilisateurs d’explorer les données de manière dynamique.

Diagramme de Sankey :

* Un diagramme de sankey est utilisé pour représenter les flux de données entre les genres, les plateformes et les éditeurs de jeux vidéo.

Mise en page :

Barre de navigation latérale :

* Une barre de navigation latérale est utilisée pour organiser les différentes pages de l’application, facilitant l’accès aux fonctionnalités.

Onglets :

* Les onglets sont utilisés pour séparer les sections d’exploration et de visualisation de données, améliorant l’organisation du contenu.

Espacement :

* Des marges et des espacements appropriés sont utilisés pour éviter le surchargement visuel et améliorer la lisibilité.

Interactivité :

* Des effets de survol et des changements de couleur sont utilisés pour ameliorer l’interactivité de l’application.

Style général :

* L’application adopte un style moderne et épuré, mettant l’accent sur la clarté et la fonctionnalité.
* L’utilisation de couleurs vives et de graphiques interactifs rend l’application attrayante et engageante.

En résumé, la charte graphique de cette application Streamlit est conçue pour offrir une expérience équilibrée, combinant esthétique et fonctionnalité pour une analyse efficace des données de ventes de jeux vidéo.

**III. Prestations attendues et modalités**

**1. Déploiement :**

Déploiement local :

* Prestations : Accès gratuit et illimité à l’application pour les utilisateurs locaux.
* Modalités: L’application est exécutée sur votre ordinateur personnel. Les utilisateurs doivent installer Python et les bibliothèques nécessaires (Streamlit, Pandas, etc…) pour l’exécuter.

Déploiement sur le cloud( Streamlit Community Cloud, GitHub):

* Modalités : L’application est hébergée sur un serveur distant, accessible via un navigateur web. Les utilisateurs n’ont pas besoin d’installer de logiciel.

Prestations :

* Accessibilité : L’application est accessible à tout moment et de n'importe où.
* Partage facile: Il suffit de partager un lien pour permettre à d’autres d’utiliser l’application.
* Scalabilité : Les plateformes cloud peuvent gérer un grand nombre d’utilisateurs.

**2. Accès et utilisation :**

Accès gratuit et public :

* Prestations : Large diffusion et accessibilité.
* Modalités : L’application est accessible à tous les utilisateurs sans inscription ni paiement.

Accès restreint (inscription authentification) :

* Prestations :
* Sécurité et contrôle d’accès et protection des données.
* Personnalisation : Possibilité d’offrir des fonctionnalités personnalisées aux utilisateurs enregistrés
* Modalités : Les utilisateurs doivent s’inscrire et se connecter pour utiliser l’application

**3. Fonctionnalités et services :**

Analyse de données :

* Modalités: Téléchargement de fichiers CSV, filtres interactifs, graphiques interactifs.
* Prestations : Visualisation et exploration de données de ventes de jeux vidéo.

Gestion des utilisateurs :

* Modalités : Base de données SQLite, formulaires d’inscription et de connexion.
* Prestations : Création et gestion de comptes utilisateurs, authentification.

Personnalisation :

* Modalités: Utilisation de code CSS, intégration de logos.
* Prestations : Amélioration de l’apparence et de l’expérience utilisateur.

**Gestion du projet**

1. **Planning et suivi**

**Semaine 1 (13-19 janvier 2020)**

Module 1 : Fondamentaux de la programmation

**Semaine 2 (20-26 janvier 2025)**

Module 2 : Développement applicatif et suivi / choix du projet fil rouge

**Semaine 3 (27 janvier - 2 février 2025)**

Module 3 : Module de données en SQL

**Semaine 4 (3-9 février 2025)**

Module 4 : Modules de données en Python

**Semaine 5 (10-16 février 2025)**

Module 5 : Data Visualisation

**Semaine 6 (17-23 février 2025)**

Module 6 : Web en scraping en Python

**Semaine 7 (3-9 mars 2025)**

Module 7 : Développement applicatif avec Streamlit

**Semaine 8 (10 mars 2025)**

Module 8 : Certification- Portfolio- Carrière





1. **Environnement humain et technique**
2. Environnement humain

Ce projet sera réalisé au sein de la promotion Van Halen, constituée de plusieurs apprenants. L’équipe encadrante de DATA ROCKSTARS accompagnera et assurera le suivi de la réalisation de ce projet tout au long du bootcamp.

b) Environnement technique

1. Langage de programmation:

* Python : Langage principal utilisé pour développer l’application Streamlit

1. Bibliothèques et frameworks :

* Streamlit: Framework Python open-source pour la création d’application web interactive pour la science des données et le machine learning.
* Pandas : Bibliothèque Python pour la manipulation et d’analyse de données tabulaires
* Matplotlib et Seaborn : Bibliothèques Python pour la création de visualisations graphiques statiques.
* Plotly Express et Plotly Graph Objects : Bibliothèques Python pour la création de graphiques interactifs.
* SQLite : Système de gestion de base de données relationnels embarqué, utilisé pour la gestion des utilisateurs.
* Pillow (PIL) : Bibliothèque Python pour le traitement d’images.
* Hashlib : Bibliothèque Python pour le hachage de mot de passe.

1. Base de données :

* SQLite : Utilisée pour stocker les informations des utilisateurs (noms d’utilisateur et mot de passe)

1. Interface utilisateur :

* HTML et CSS : Utilisés pour personnaliser l’apparence de l’application Streamlit

1. Déploiement :

* Streamlit Community Cloud : Plateforme d’hébergement gratuite pour les applications Streamlit.
* Déploiement local : Exécution de l’application sur un ordinateur local.

1. Système d’exploitation :

L’application peut être exécutée sur différents systèmes d’exploitation; tels que:

* Windows
* macOS
* Linux

1. **Objectifs de qualité**
2. Fonctionnalité **:**

* Exactitude des données : L’application doit afficher des données précises et à jour.
* Fonctionnalités complètes : L’application doit offrir toutes les fonctionnalités prévues, telles que le chargement de fichiers, le filtrage des données et la visualisation de graphiques.
* Facilité d’utilisation : L’application doit être intuitive et facile à utiliser, même pour les utilisateurs non techniques.

1. Performance :

* Temps de chargement rapides: L’application doit se charger rapidement, même avec de grands ensembles de données.
* Réactivité : L’application doit répondre rapidement aux interactions de l’utilisateur.
* Evolutivité : L’application doit pouvoir gérer un grand nombre d'utilisateurs et de données

1. Fiabilité :

* Disponibilité: L’application doit être disponible 24h/24 et 7j/7.
* Stabilité : L’application ne doit pas planter ou se bloquer.
* Sécurité : L’application doit protéger les données des utilisateurs contre les accès non autorisés.

1. Maintenabilité :

* Code propre et organisé : Le code de l’application doit être facile à lire et à comprendre.
* Documentation complète : L’application doit être accompagnée d’une documentation complète pour faciliter la maintenance et les mises à jour.
* Modularité : L’application doit être conçue de manière modulaire pour faciliter l’ajout de nouvelles fonctionnalités.

1. Expérience utilisateur (UX) :

* Interface utilisateur claire et intuitive : L’interface utilisateur doit être facile à comprendre et à utiliser.
* Visualisations graphiques efficaces : Les graphiques doivent être clairs, informatifs et esthétiques.
* Personnalisation : L’application doit permettre aux utilisateurs de personnaliser l’affichage des données et des graphiques

**Spécifications fonctionnelles du projet**

1. Navigation et interface latérale :

Barre de navigation latérale :

* Permet aux utilisateurs de naviguer entre différentes pages de l’application.
* Les pages incluent “Accueil”, “Logos des plateformes” et “Paramètres”

Page “Accueil” :

* Page d'accueil de l’application.

Page “Logos des plateformes” :

* Affiche les logos des différentes plateformes de jeux vidéo.
* Permet aux utilisateurs de sélectionner une plateforme spécifique pour afficher son logo
* Gère les cas ou un logo n’est pas trouvé ou n’est pas disponible

Page “Paramètres” :

* Fournit des informations sur Streamlit et la charte graphique de l’application
* Inclut des liens vers des ressources utiles.

Personnalisation de l’interface :

* Utilisation de CSS pour personnaliser l’apparence de l’application (couleurs de fond, styles de blocs, etc..)
* Ajout d’effets interactifs (survol, clic) sur les éléments de l’interface
* Ajout d’une image de fond d'écran à l’application

1. Gestion des données utilisateur :

Collecte de données utilisateur :

* Permet de collecter les noms d’utilisateur et les adresses e-mail.
* Stocke les données dans une base de données SQLite.

Consentement de l’utilisateur :

* Demande le consentement de l’utilisateur pour l’utilisation de ses données.

Droit des utilisateurs :

* Permet aux utilisateurs d’afficher et de supprimer leurs données personnelles.

1. Fonctionnalités supplémentaires :

Affichage de logos de plateformes :

* Affiche les logos des plateformes de jeux vidéo.
* Permet de sélectionner une plateforme spécifique.
* Gestion des erreurs liées aux images.

**Spécifications techniques du projet**

Interface et utilisateur :

* HTML et CSS: Pour la personnalisation de l’apparence de l’application.
* JavaScript: Pour l’ajout d’interactivité à l’interface.
* Barre de navigation latérale : Pour la navigation entre les pages.
* Eléments Streamlit : Textes, sélecteurs, boutons, images, etc.

Base de données :

SQLite **:**

* Stockage des données utilisateur (noms d’utilisateur, mots de passe).
* Gestion des opérations CRUD (création, lecture, mise à jour, suppression).

Sécurité :

* Hachage des mots de passe: Utilisation de **hashlib** pour stocker les mots de passe de manière sécurisée.
* Gestion des droits d’utilisateur: Permet aux utilisateurs d’afficher et de supprimer leurs données.

Fonctionnalités :

* Affichage de logos : Affichage de logos de plateformes de jeux vidéo.
* Collecte de données utilisateur : Collecte des noms d’utilisateur et adresses e-mail.
* Consentement de l’utilisateur : Demande de consentement pour l’utilisation des données.
* Personnalisation de l’interface : Utilisation de CSS pour personnaliser l’apparence.
* Ajout d’une image de fond d’écran : Utilisation de base64 pour intégrer une image de fond.
* Ajout d’interactivité : Ajout d’effet de survol et de clic sur les éléments.

Environnement d’exécution:

* Navigateur web: L’application est exécutée dans un navigateur web.
* Serveur Streamlit: Le serveur Streamlit gère la communication entre le navigateur et le code Python.

Architecture :

* Architecture monothread : Streamlit utilise un modèle pour gérer les requêtes des utilisateurs.
* Architecture événementielle : Streamlit utilise une architecture événementielle pour gérer les interactions de l’utilisateur.

Tests : Test unitaires pour vérifier la fonctionnalité des composants individuels.

* Test d’intégration pour vérifier l'interaction entre les composants.
* Test d’utilisabilité pour évaluer l'expérience utilisateur.

**Réalisations**

1. **Exemple 1**
2. **Contexte**

Exemple :

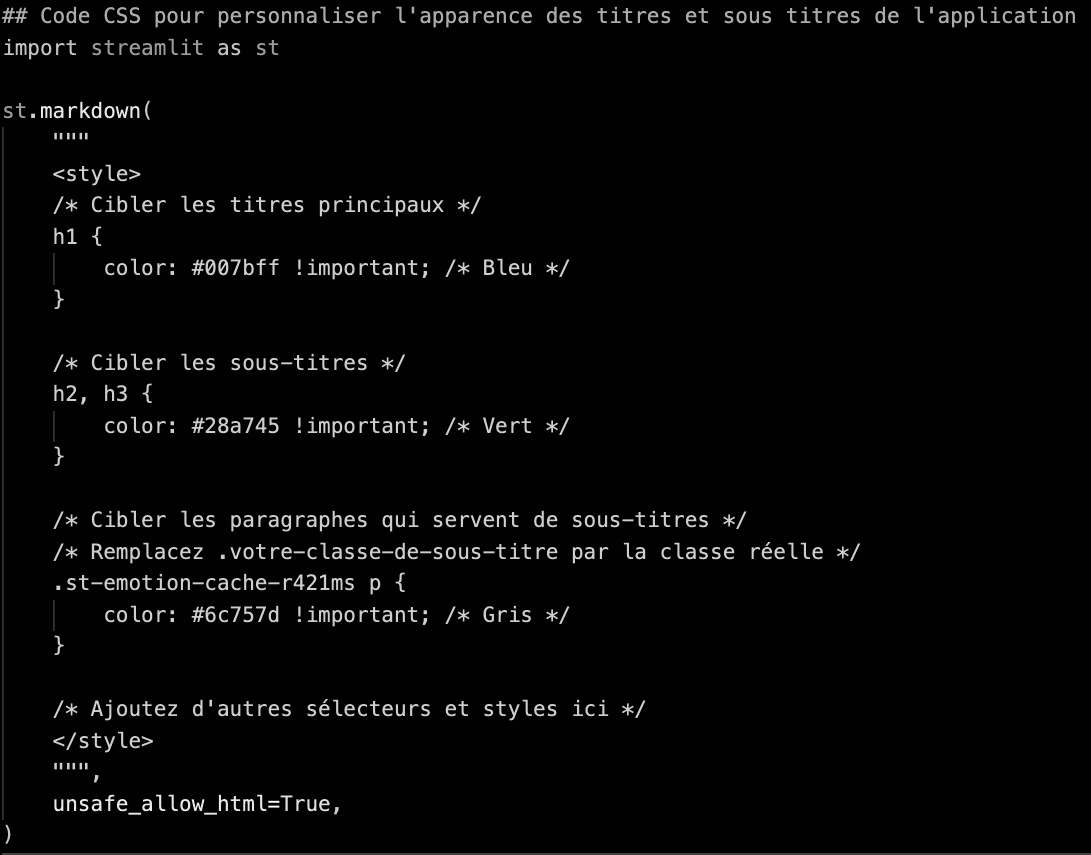
Le code fourni illustre l'utilisation de CSS intégré directement dans l’application Streamlit pour modifier les styles des titres et des sous-titres.

* Plus précisément, il définit les couleurs suivantes :
* #007bff (bleu) pour les titres principaux (h1).
* #28a745 (vert) pour les sous titres (h2 et h3)
* #6c757d (gris) pour les paragraphes servant de sous-titres, identifiés par la classe .st-emotion-cache-r421ms p.
* L’utilisation du mot clé !important assure que ces styles personnalisés prennent le dessus sur les styles par défaut de Streamlit.

Contexte :

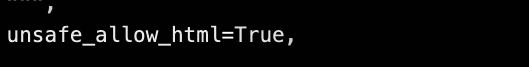
* L’objectif était de créer une interface utilisateur plus attrayante et cohérente avec une charte graphique spécifique.
* En personnalisant les couleurs des titres et des sous-titres, l’application devient plus agréable à utiliser et facilite la lecture des informations.
* Cette personnalisation permet également de renforcer l’identité visuelle de l’application et de la rendre plus professionnelle.
* Dans le contexte de l’application, il est possible que le bleu soit la couleur principale de l’entreprise, et le vert une couleur secondaire, ce qui justifie leurs utilisations.
* Le gris, quant à lui, permet d’adoucir la lecture des informations, qui sont moins importantes que les titres et sous-titres.
* De plus, l’utilisation de code CSS intégré directement dans Streamlit démontre une approche flexible et efficace pour personnaliser l’apparence de l’application sans recourir à des fichiers CSS externes.

1. **Aperçus du code**



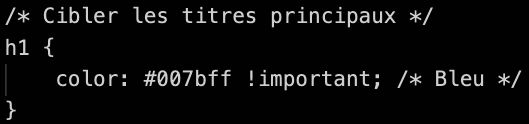
1. **Arguments et commentaires**





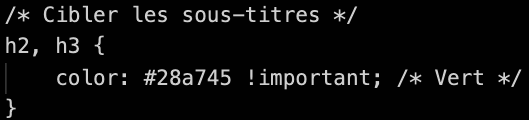
Argument : Utilisation de st.markdown() pour insérer du code HTML et CSS directement dans l’application Streamlit.

Commentaire: unsafe\_allow\_html=True est requis pour permettre à Streamlit d’interpréter le code HTML et CSS. Cela offre une grande flexibilité pour personnaliser l’apparence de l’application.



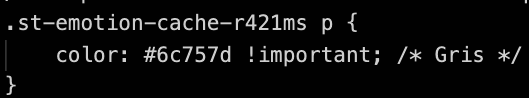
Argument : Modification de la couleur des titres principaux (h1) en bleu (#007bff).

Commentaire : !important assure que ce style spécifique prend le dessus sur les styles par défaut de Streamlit, garantissant que la couleur bleue est appliquée.



Argument : Modification de la couleur des sous-titres (h2 et h3) en vert (#28a745).

Commentaire: Comme pour les titres principaux, !important est utilisé pour assurer la priorité de ce style.



Argument : Modification de la couleur des paragraphes (<p>) qui servent de sous-titres, identifiés par la classe CSS .st-emotion-cache-r421ms, en gris (#6c757d).

Commentaire : Ce sélecteur cible spécifiquement les paragraphes dans un contexte où ils sont utilisés comme sous-titres, en utilisant une classe CSS générée par Streamlit. !important est utilisé pour assurer la priorité de ce style.

1. **Exemple 2**
2. **Contexte**

Exemple:

* Le code utilise streamlit.components.v1.html() pour intégrer du code JavaScript directement dans l’application Streamlit.
* Ce JavaScript ajoute des interactions dynamiques à un élément spécifique de l’interface, identifié par la classe CSS .stMainBlockContainer.block-container.st-emotion-cache-mtjnbi.eht7o1d4

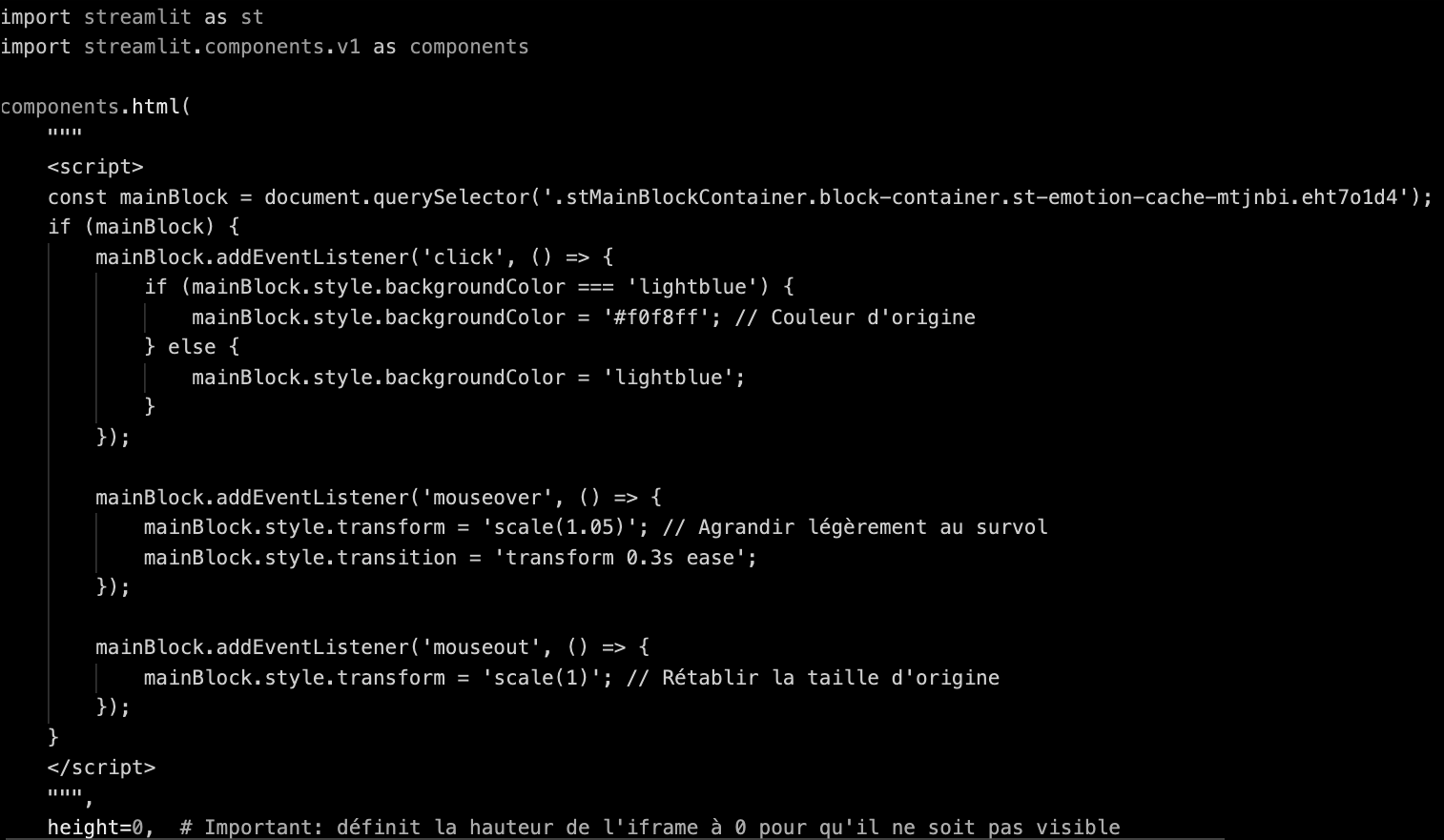
Les interactions incluent :

* Un changement de couleur de fond au clic (bascule en lightblue et #ff0f8ff).
* Un effet de grossissement au survol (mouseover).
* Un retour à la taille normale lorsque la souris quitte l’élément (mouseout).
* L’argument height=0 est utilisé pour s’assurer que l’iframe contenant le code JavaScript n'occupe pas d’espace visible dans l’interface utilisateur.

Contexte :

* L’objectif était de rendre l’interface utilisateur plus interactive et réactive améliorant ainsi l’expérience utilisateur.
* En ajoutant des effets visuels et des interactions dynamiques, l’application devient plus attrayante et engageante.
* Cette approche permet de personnaliser le comportement des éléments de l’interface utilisateur au-delà des fonctionnalités de base de Streamlit.
* Dans le contexte de l’application, cet élément pourrait être un conteneur principal affichant des informations. L’ajout d’interactions visuelles signale à l’utilisateur que cet élément interactif et incite à l’exploration.
* L’utilisation de JavaScript intégré directement dans Streamlit démontre une approche flexible pour ajouter des fonctionnalités interactives avancées.

1. **Aperçus du code**



1. **Arguments et commentaires**

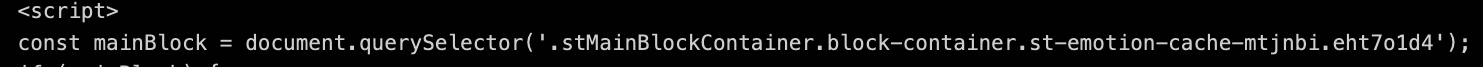


Argument :

* Importation de la bibliothèque Streamlit, essentielle pour créer et afficher des éléments interactifs dans l’application web.
* Importation du module components de Streamlit, nécessaire pour intégrer du code HTML et JavaScript personnalisé.

Commentaire :

* import streamlit as st est la ligne standard pour utiliser Streamlit.
* import streamlit.components.v1 as components est spécifique à l’intégration de composants HTML et JavaScript.



Argument :

* Utilisation de document.querySelector() pour sélectionner l’élément HTML à manipuler, en utilisant sa classe CSS

Commentaire :

* Cette ligne cible un élément spécifique de l’interface utilisateur de Streamlit, ce qui permet de personnaliser son comportement.
* La classe CSS utilisée est générée dynamiquement par Streamlit, ce qui peut nécessiter une inspection de l’élément pour l’identifier.







Argument :

* Ajout de gestionnaires d’événements pour le clic, le survol de la souris et le moment où la souris quitte l’élément.

Commentaire :

* Ces gestionnaires d’événements définissent les actions à effectuer lorsque l’utilisateur interagit avec l’élément.
* Cela permet de créer des interactions dynamiques et réactives.

1. **Exemple 3**
2. **Contexte**

Exemple **:**

* Le code met en place une fonctionnalité de gestion des utilisateurs directement dans une application Streamlit.
* Il utilise une base de données SQLite (via les fonctions init\_db, add\_user et get\_users, supposées définies dans un module db) pour stocker et récupérer les informations des utilisateurs.
* Une barre de navigation latérale est créée avec Streamlit pour offrir une interface utilisateur dédiée à la gestion des utilisateurs.
* Un formulaire permet d’ajouter de nouveaux utilisateurs avec un nom d’utilisateur et un mot de passe.
* La liste des utilisateurs enregistrés est affichée sous forme de tableau à l’aide de st.dataframe.

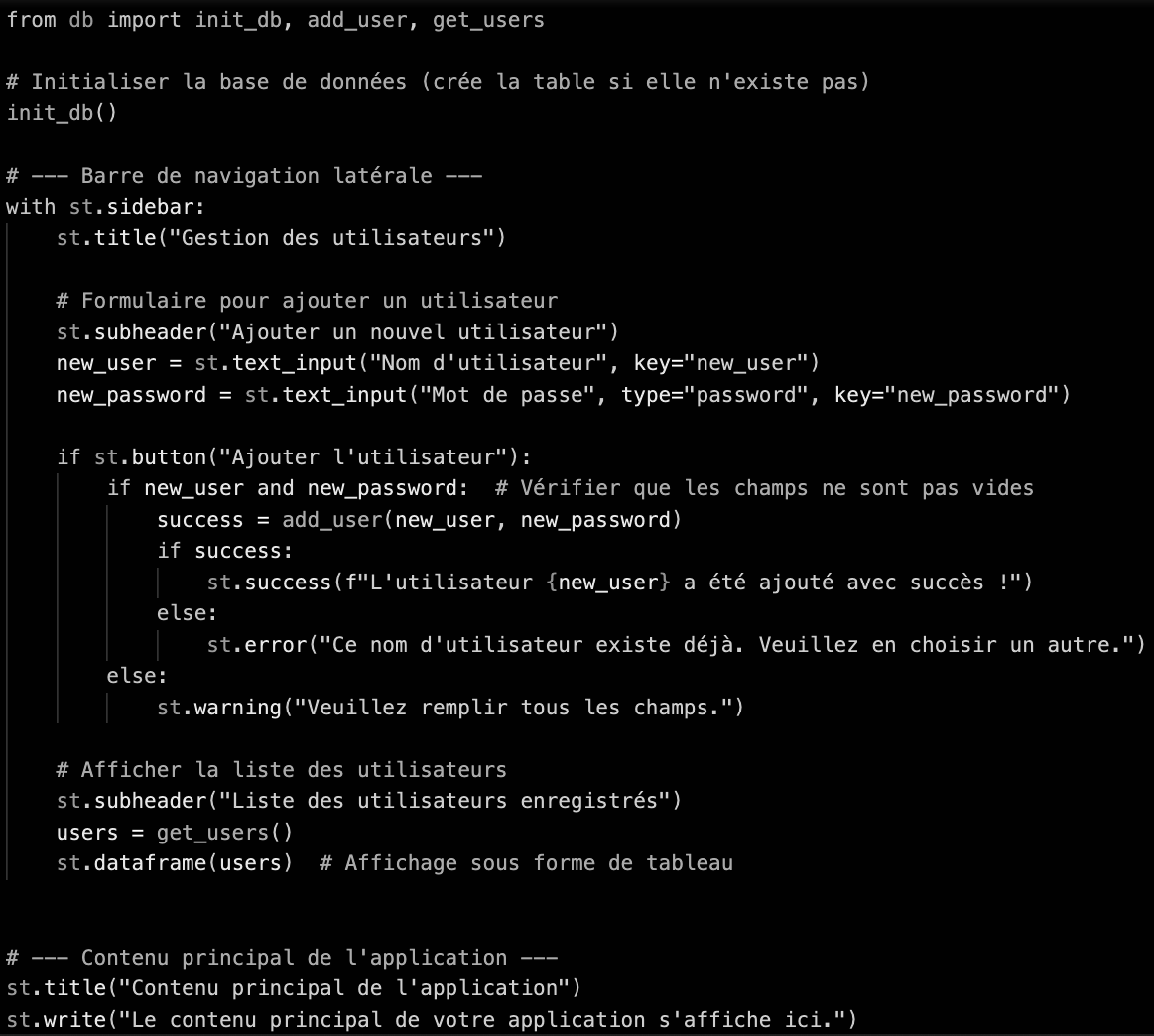
Contexte **:**

* L’objectif était de créer une application Streamlit avec une fonctionnalité de gestion des utilisateurs intégrée.

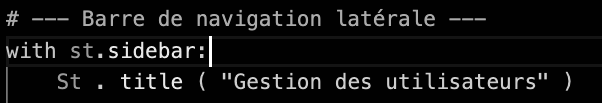
Cela peut être utile dans divers contextes, tels que :

* Une application nécessitant une authentification pour accéder à certaines fonctionnalités.
* Un tableau de bord ou seuls les utilisateurs enregistrés peuvent consulter ou modifier les données.
* Un outil de collaboration ou les utilisateurs ont des rôles et des permissions spécifiques.
* L’utilisation de SQLite permet de simplifier la gestion de la base de données, en particulier pour les applications de petite ou moyenne taille.
* L’interface Streamlit offre une expérience utilisateur conviviale pour l’ajout et la visualisation des utilisateurs.

1. **Aperçus du code**

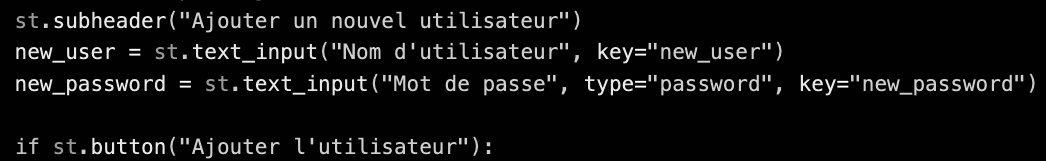


1. **Arguments et commentaires**



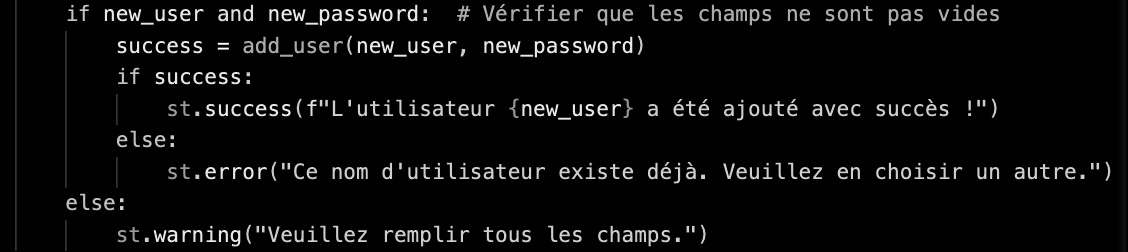
Argument : Utilisation de st.sidebar pour créer une barre de navigation latérale dans l’application Streamlit.

Commentaire: La barre latérale est utilisée pour organiser les fonctionnalités liées à la gestion des utilisateurs, les séparant du contenu principal de l’application.



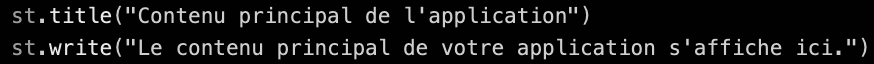
Argument: Création d’un formulaire avec des champs de texte pour le nom d’utilisateur et le mot de passe, et un bouton pour soumettre le formulaire.

Commentaire : Ce formulaire permet aux utilisateurs d’ajouter de nouveaux utilisateurs à la base de données.



Argument : Validation des champs du formulaire et appel de la fonction add\_user pour ajouter un nouvel utilisateur.

Commentaire: Cette logique gère les cas où les champs sont vides, ou l’utilisateur existe déjà, et affiche des messages de succès ou d’erreur appropriés.



Argument: Affichage d’un titre et d’un texte de démonstration pour le contenu principal de l’application.

Commentaire: Cette section est un espace réservé pour le contenu principal de l’application, qui peut être personnalisé en fonction des besoins.

**Présentation d’un jeu d’essai**

1. **Données en entrée**

* Le jeu de données “vgsales\_cleaned.csv” contient des informations sur les ventes de jeux vidéo.
* Il a été obtenu après un nettoyage du jeu de données original “vgsales.csv”.
* Il comprend des informations telles que le titre du jeu, la plateforme, le genre, l’éditeur et les ventes mondiales.

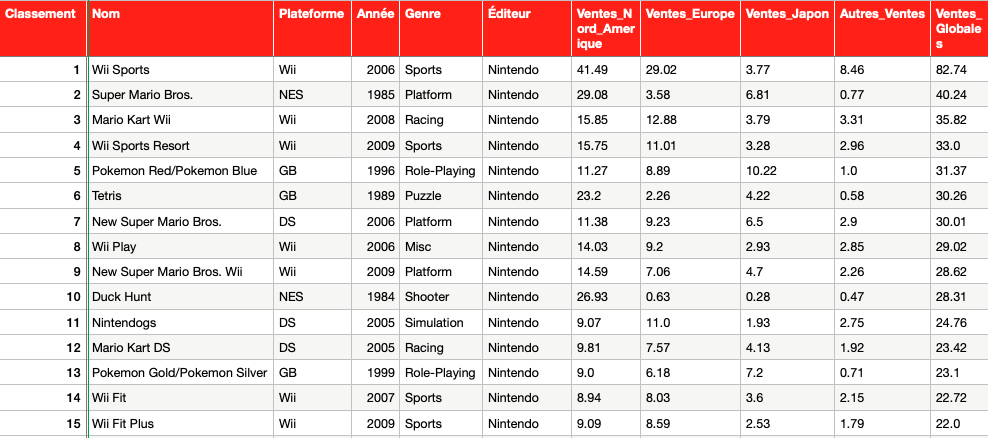
Structure des données :

Colonnes :

* Classement
* Nom
* Plateforme
* Année
* Genre
* Editeur
* Ventes Nord Amérique
* Ventes Europe
* Ventes Japon
* Autres Ventes
* Ventes Globales

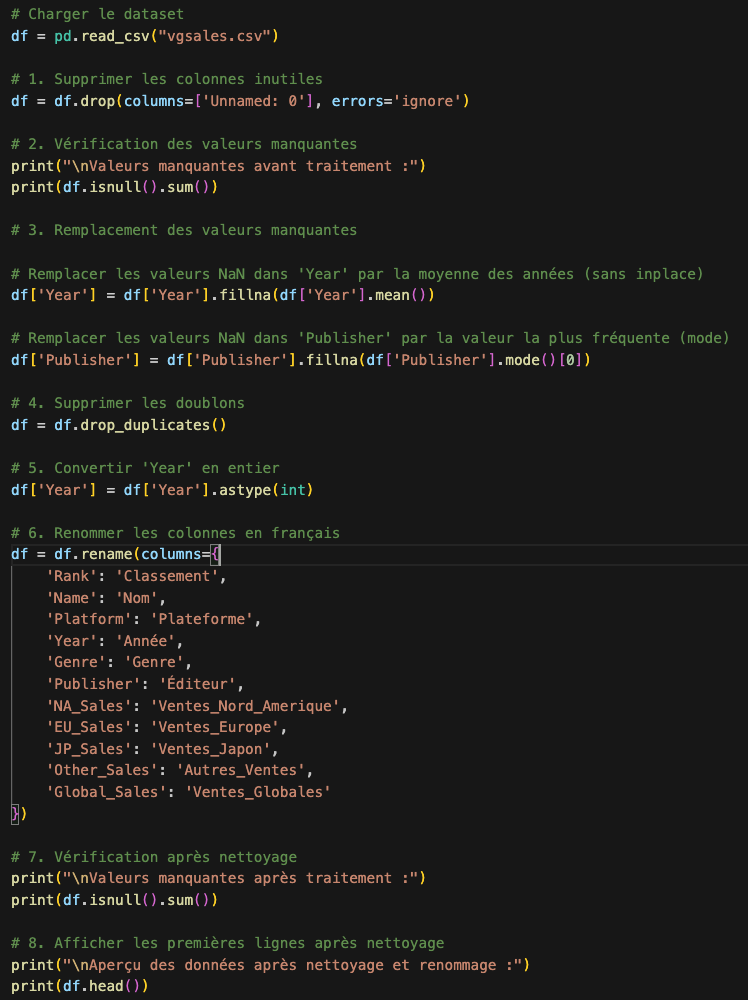
Description des données brutes :

* Le fichier “vgsales\_cleaned.csv” a déjà subi un processus de nettoyage.
* Exemple de données brutes (extrait) :



1. **Données attendues**

* Les données devraient être nettoyées pour être prêtes à l’analyse.
* Les valeurs manquantes dans “year” devaient être remplacées par la moyenne des années.
* Les doublons devaient être supprimés.
* La colonnes “Year” devait être convertie en entiers.
* Les noms de colonnes devaient être traduits en français.
* Après le traitement, aucune valeur “Nan” ne devait être présente dans la colonne “Year”.
* La colonnes “Year” devait contenir des entiers.
* Les premières lignes du DataFrame après le traitement montrent les données nettoyées avec les noms de colonnes en français.



1. **Données obtenues**

Après avoir chargé le fichier “vgsales\_cleaned.csv” dans votre application Streamlit (ou dans votre environnement d’analyse de données), vous obtiendrez un dataframe (si vous utilisez Pandas) ou une structure de données similaire contenant les informations suivantes :

Structure des données :

* Le jeu de données est structuré en un tableau avec des lignes et des colonnes
* Chaque ligne représente un jeu vidéo
* Chaque colonnes représente une caractéristique de jeu vidéo

Colonnes :

* Classement : Le classement du jeu en termes de ventes mondiales.
* Nom : Le titre du jeu vidéo.
* Plateforme : La plateforme sur laquelle le jeu est sorti ( par exemple : Wii, Nes, PS4).
* Année : L’année de sortie du jeu.
* Genre : Le genre du jeu ( par exemple : Sports, Racing, etc..).
* Éditeur : L’éditeur du jeu.
* Ventes Nord Amérique : Les ventes en Amérique du Nord (en millions).
* Ventes Japon : Les ventes au Japon (en millions).
* Autres Ventes : Les ventes dans d’autres régions (en millions).
* Ventes Globales : Les ventes mondiales totales (en millions).

Utilisation des données :

Ce jeu de données permet d'effectuer diverses analyses, telles que :

* Analyse des ventes par plateforme, genre ou éditeur.
* Visualisation des tendances de ventes au fil du temps.
* Identification des jeux les plus populaires.
* Comparaison des ventes entre les régions.

**Veille sur les vulnérabilités de sécurité**

1. **Besoin d’information**

Identification des risques :

* Le code utilise des bases de données SQLite pour stocker des informations d’utilisateurs, y compris des mots de passe.
* Il y a un besoin crucial de s’assurer que ces données sont stockées et traitées de manière sécurisée pour prévenir les accès non autorisés et les fuites de données.
* L’utilisation de code JavaScript personnalisé intégré dans l’application via components.html() peut également introduire des vulnérabilités si elle n’est pas gérée correctement.

Protection des données sensible :

* Les mots de passe des utilisateurs sont des informations extrêmement sensibles qui
* nécessitent une protection rigoureuse.
* Les informations personnelles collectées (noms d’utilisateur, adresses e-mail) doivent également être protégées conformément aux réglementations sur la confidentialité des données.

Mise à jour des connaissances :

* Il est essentiel de se tenir informé des dernières vulnérabilités de sécurité liées à Streamlit, SQLite, JavaScript et aux bibliothèques Python utilisées.
* Cela inclut les vulnérabilités connues, les meilleures pratiques de sécurité et les mises à jour de sécurité.



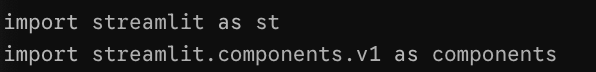
1. **Mots clés de recherche**

Généraux :

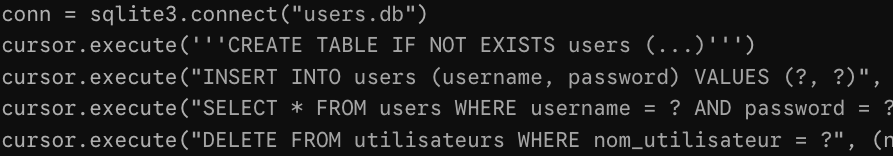
* Vulnérabilités de sécurité Streamlit.
* Sécurité des applications web Python.
* Protection des données SQLite.
* Sécurité JavaScript dans Streamlit.
* Meilleures pratiques de sécurité pour les applications web.

Spécifiques :

* SQL injection SQLite Python.
* Cross-site scripting (XSS) Streamlit components.
* Hachage de mots de passe Python hashlib.
* Gestion des sessions Streamlit.
* OWASP top 10 Streamlit.













1. **Liste des sites retournés**

Documentation officielle :

* Streamlit documentation : Pour les mises à jour de sécurité et les meilleures pratiques.
* Python documentation : Pour la sécurité des bibliothèques (SQLite, hashlib).
* SQLite documentation : Pour la sécurité de la base de données.

Organisations de sécurité :

* OWASP (Open Web Application Security Project) : Pour les vulnérabilités web courantes.
* NIST (National Institute of Standards and Technology) : Pour les bases de données de vulnérabilités (CVE)
* CERT (Computer Emergency Response Team) : Pour les alertes de sécurité.

Communautés et forums :

* Stack Overflow : Pour les discussions sur les problèmes de sécurité.
* GitHub : Pour les rapports de problèmes et les mises à jour de sécurité des bibliothèques.

1. **Critères de sélection des sites**

Fiabilité :

* Privilégier les sites officiels, les organisations de sécurité reconnues et les sources réputées.

Pertinence :

* Se concentrer sur les sites qui fournissent des informations spécifiques aux technologies utilisées ( Streamlit, Python, SQLite, JavaScript).

Actualité :

* rechercher les informations les plus récentes sur les vulnérabilités et les mises à jour de sécurité.

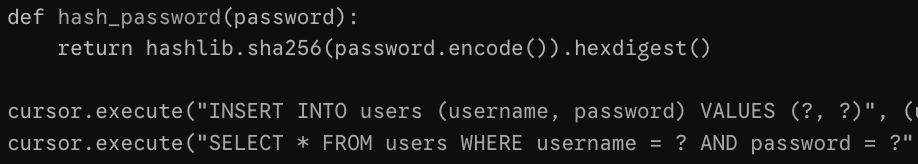
Exhaustivité :

* Consulter plusieurs sources pour obtenir une vue d’ensemble complète des risques et des solutions.

1. **Vulnérabilités trouvées**

Potentielles vulnérabilités :

* Stockage de mots de passe en clair : Bien que le code utilise hashlib, il est crucial de s’assurer que le hachage est effectué correctement et que les mots de passe ne sont jamais stockés en clair.



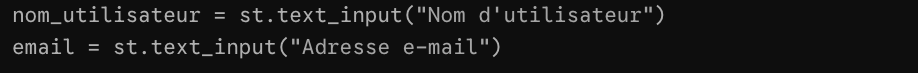
* Injection SQL : L’utilisation de requêtes SQL directes peut être vulnérable aux attaques par injection SQL si les entrées utilisateur ne sont pas correctement validées et échappées.



* Cross-site scripting(XSS) : L’intégration de code JavaScript personnalisé via components.html() peut introduire des vulnérabilités XSS si les données utilisateur sont affichées sans être correctement échappées.



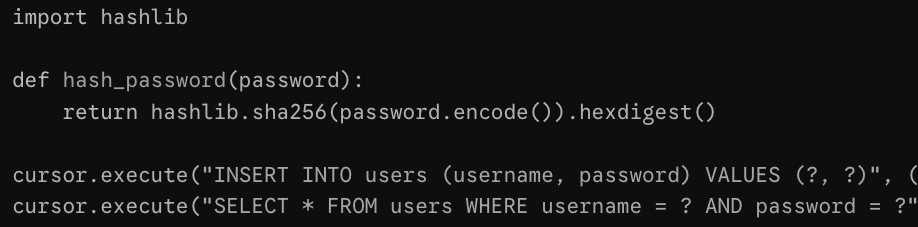
* Gestion des sessions : Streamlit ne gère pas les sessions par défaut, ce qui peut poser des problèmes de sécurité si des informations sensibles sont stockées côté client.
* Protection des données : Il est essentiel de s’assurer que les données personnelles collectées sont stockées et traitées conformément aux réglementations sur la confidentialité des données (RGPD, etc.).



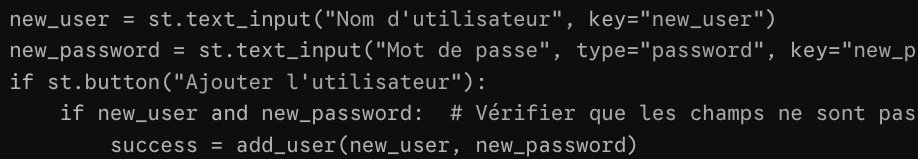
1. **Failles potentielles corrigées**

Mesures de sécurité implémentées :

* Hachage des mots de passe : le code utilise hashlib pour hacher les mots de passe avant de les stocker dans la base de données.



* Validation des entrées utilisateur : Il est essentiel d’ajouter une validation rigoureuse des entrées utilisateur pour prévenir les attaques par injection SQL et XSS.



* Utilisation de paramètres préparés : Pour prévenir les attaques par injection SQL, il est recommandé d'utiliser des paramètres préparés lors de l’exécution de requêtes SQL.



* Politique de confidentialité : Il est important de mettre en place une politique de confidentialité claire et transparente pour informer les utilisateurs sur la collecte et l’utilisation de leurs données.
* Mise à jour des bibliothèques : Il est essentiel de maintenir les bibliothèques Streamlit, Python et SQLite à jour pour bénéficier des dernières corrections de sécurité.



**Pratiques DevOps**

Le DevOps est un ensemble de méthodes visant à faciliter l’intégration d’un produit comme une application par exemple. Grâce au DevOps, on minimise le temps de transition entre le développement informatique (concevoir et coder l’application) et son déploiement sur une machine. On facilite également la vie des développeurs en évitant les “régressions” (c’est-à-dire l’apparition de nouveaux bugs dûes à une modification du code de l’application) grâce à l’automatisation de tests. Globalement, l’objectif des pratiques DevOps est d’automatiser un maximum la chaîne de déploiement de l’application.

1. **Tests unitaires**

Un test unitaire est une méthode de vérification automatisée qui s'applique à la plus petite partie testable d'une application (par exemple, une fonction Python) afin de s'assurer qu'elle se comporte comme prévu. Concrètement, il s'agit de :

* **Isoler une unité de code :** Chaque test vérifie une fonctionnalité spécifique sans dépendre d'autres composants.
* **Automatiser la validation :** Les tests unitaires s'exécutent rapidement et permettent de détecter immédiatement les erreurs ou régressions lors de modifications du code.
* **Faciliter la maintenance :** En garantissant que chaque composant fonctionne correctement individuellement, ils aident à préserver la stabilité globale de l'application au fil des évolutions.

Dans l’idéal, il devrait exister un fichier de test unitaire par fonctionnalité de l’application. Dans le cas de notre application Streamlit, par exemple, il serait bien d’avoir des tests unitaires pour :

* Vérifier le bon import du jeu de données grâce à Pandas afin de détecter une potentielle suppression du CSV utilisé / une potentielle anomalie dans la table SQL utilisée. —> **CHOISIR ICI.**
* Vérifier que la fonction de traitement des données : Data Cleaning, formatage, … fonctionne correctement et qu’elle rend bien les bons résultats malgré des données d’entrée d’une qualité extrêmement médiocre.
* Vérifier la bonne application des filtres sur le DataFrame Pandas…

Nous avons dans notre cas, implémenté dans un but éducatif des tests unitaires à l’aide de deux librairies largement utilisées en Python : Pytest et Unittest. On a choisi de nous concentrer sur quelques tests unitaires : l’authentification et le bon import du jeu de données qui sont les deux premières fonctionnalités utilisées par l’utilisateur.

Par exemple, dans le cas du test unitaire présenté ci-dessus, on tente de requêter la base de données initialisée au préalable et on s’assure à l’aide de la ligne 12 que la réponse à cette exécution n’est pas None (auquel cas, cela impliquerait qu’il y a une erreur dans l’initialisation de la base de données : la table Users n’a pas été créée par exemple).

On teste ensuite dans la méthode suivante l’inscription d’un utilisateur dans la base de données : on ajoute un nouvel utilisateur avec son e-mail et mot de passe et on s’assure que la table Users contient bien un enregistrement.

1. **Tests end-to-end**

Un test end-to-end est un test automatisé qui simule l'expérience complète d'un utilisateur réel en interagissant avec l’application à l’aide de scénarios préparés à l’avance. Grâce à ce test, on serait alors capable d’automatiser des tests de clics sur des boutons, d’ajustement des filtres, de saisie de données…

Contrairement aux tests unitaires qui se concentrent sur des fonctionnalités isolées (petites fonctions qui ne remplissent qu’un seul usage), les tests end-to-end vérifient que l’ensemble de l’application, de l’interface utilisateur Streamlit jusqu’au back-end (fonctions de filtrage et de traitement des données) fonctionne sans défaut.

Dans le cas de l’application qui a été développée, je n’ai pas eu le temps de développer de tels scénarios de tests. Mais, il aurait été intéressant en utilisant la librairie Selenium de tester mon application sur différents navigateurs par exemple.

1. **Pipeline GitHub CI/CD**

GitHub CI/CD est un ensemble d’outils destinés à automatiser le processus de déploiement d’un développement. Il repose sur deux concepts majeurs :

* **Intégration Continue (CI)** : Cette pratique consiste à intégrer régulièrement les modifications apportées par les développeurs dans un dépôt commun. À chaque modification, les tests automatisés (unitaires par exemple) sont exécutés afin de vérifier que le nouveau code ne génère pas de régressions ou d’erreurs. Ainsi, tout problème est détecté rapidement dès son apparition.
* **Livraison/Déploiement Continu (CD)** : Une fois que le code a passé les tests d’intégration, il peut être automatiquement préparé pour être déployé dans un environnement de production ou de pré-production. Le déploiement continu va plus loin en automatisant également la mise en production dès que les tests sont validés.

GitHub propose notamment **GitHub Actions**, une solution intégrée que nous avons utilisée dans le cadre de ce projet, qui permet de créer et de gérer des flux (workflows) automatisés. Ces workflows sont définis dans des fichiers de configuration au format YAML et se déclenchent en fonction d’événements spécifiques dans le dépôt (comme un push, une pull request…).

Dans le flux construit, il a été décidé qu’à chaque push d’un nouveau commit, on allait :

* Vérifier si les tests unitaires fonctionnent correctement.
* Vérifier la qualité du code écrit à l’aide de PyLint.
* Vérifier si l’image Docker de l’application et de ses dépendances parvenait à être “build” convenablement.

1. **Conteneurisation Docker**

Docker est un outil de “conteneurisation” : il permet d'emballer une application ainsi que toutes ses dépendances (version de Python, librairies utiles à la bonne version ainsi que d’autres programmes utiles, éventuellement données) dans un environnement isolé appelé *conteneur*. Cela signifie que l'application peut s'exécuter de manière identique, quelle que soit la machine hôte, car tout ce dont elle a besoin est inclus dans le conteneur.

C’est un des outils les plus importants dans le cadre des pratiques DevOps car il permet d’éviter les problèmes de déploiement et d’intégration. On a choisi de l’utiliser pour notre application afin qu’elle soit utilisable sur n’importe quel ordinateur, peu importe où elle est hébergée, facilement et sans bug d’intégration. Pour conteneuriser notre application, il a fallu :

1. **Build son image :** Une image Docker est comme une photo de l'environnement d'exécution de l'application. Pour la créer, j’ai rédigé un fichier appelé *Dockerfile* qui décrit toutes les étapes nécessaires pour installer les dépendances, copier le code source, et configurer l'environnement. Par exemple, dans un projet Streamlit, le Dockerfile va spécifier l'installation de Python, des bibliothèques requises, et la commande pour lancer Streamlit. La commande docker build -t mon-appli . permet ensuite de construire cette image à partir du Dockerfile.
2. **Exécution d'un conteneur :** Une fois l'image construite, on peut lancer un conteneur qui est une instance en cours d'exécution de cette image. La commande docker run -d -p 8501:8501 mon-appli démarre le conteneur en arrière-plan et fait correspondre le port 8501 du conteneur sur le port 8501 de la machine hôte, rendant ainsi l'application accessible.

Comme dit dans la partie précédente, j’ai fait en sorte que mon workflow Github Actions teste le build de l’image et l’exécution du conteneur à chaque push pour éviter des bugs inattendus de déploiement.

**Description d’une situation de travail ayant nécessité une recherche**

1. **Besoin d’information**

Contexte :

* Le développement d’une application Streamlit complexe, intégrant la gestion des utilisateurs et l’authentification, a nécessité une approche rapide et efficace.
* L’utilisation d’IA (modèle de langage, outils de génération de code) a permis d'accélérer le processus de conception et de développement.
* Cependant, l’utilisation d’IA a soulevé des questions cruciales concernant la sécurité de l’application, en particulier en ce qui concerne les vulnérabilités potentielles introduites par le code générée par l’IA.

Problématique :

* Comment s’assurer que l’application Streamlit, conçue avec l’aide d’IA, respecte les meilleures pratiques de sécurité ?
* Quelles sont les vulnérabilités potentielles spécifiques aux applications générées ou assistées par l’IA ?
* Comment valider et sécuriser le code généré par l’IA pour prévenir les attaques telles que l’injection SQL et les attaques XSS ?
* Comment garantir la protection des données sensibles des utilisateurs (mots de passe, informations personnelles) dans un contexte de développement assisté par l’IA ?

1. **Mots clés de recherche**

“Sécurité des applications générées avec l’aide de l’IA” :

* Ce mot-clé vise à explorer les risques spécifiques liés à l’utilisation de l’IA pour générer du code.
* Il inclut la recherche de vulnérabilités potentielles introduites par les modèles d’IA, les outils de génération de code et les API.
* L’objectif est de comprendre comment valider et sécuriser le code généré par l’IA pour prévenir les attaques.

“Vulnérabilités de sécurité Streamlit et IA” :

* Ce mot-clé se concentre sur les vulnérabilités spécifiques qui peuvent survenir lors de l’utilisation conjointe de Streamlit et de l’IA.
* Il inclut la recherche de problèmes liés à l'intégration de code générée par l’IA dans une application Streamlit.
* L’objectif est d’identifier les risques potentiels et de trouver des solutions pour les atténuer.

“Validation de code IA pour la sécurité web” :

* Ce mot-clé vise à explorer les méthodes et les outils permettant de valider et de sécuriser le code généré par l’IA pour les applications web.
* Il inclut la recherche de techniques d’analyse statique et dynamique, de test de sécurité et de revues de code.
* L’objectif est de s’assurer que le code généré par l’IA respecte les meilleures pratiques de sécurité web.

“Protection des données dans les applications” :

* Ce mot-clé se concentre sur les enjeux de protection des données sensibles (mots de passe, informations personnelles) dans les applications développées avec l’aide de l’IA.
* Il inclut la recherche de techniques de chiffrement, de hachage et de gestion des accès.
* L’objectif est de garantir la confidentialité et l’intégrité des données utilisateur.

“Injection SQL dans les applications” :

* Ce mot-clé se concentre sur les risques d’injection SQL dans les requêtes générées par l’aide de l’IA.
* Il inclut la recherche de techniques de validation des entrées de paramètres des requêtes.
* L’objectif est de prévenir les attaques qui pourraient compromettre la base de données.

“Hachage de mots de passe” :

* Ce mot-clé vise à explorer les méthodes de hachage de mots de passe sécurisées dans un contexte de développement assisté par l’IA.
* Il inclut la recherche d’algorithmes de hachage robustes et de techniques de salage.
* L’objectif est de protéger les mots de passe des utilisateurs contre les attaques par force brute et les attaques par table arc-en-ciel.

1. **Liste des sites retournés**

Documentation officielle de Streamlit :

* Lien : <https://docs.streamlit.io/>
* Description : Documentation complète de Streamlit, incluant des guides, des tutoriels et des informations sur les mises à jour de sécurité.

Documentation officielle de Python :

* Lien : <https://docs.python.org/>
* Description : Documentation de référence pour le langage Python et ses bibliothèques standard, incluant des informations sur la sécurité des bibliothèques utilisées (SQLite, hashlib, etc.).

Documentation de SQLite :

* Lien : <https://www.sqlite.org/docs.html>
* Description : Toutes les documentations concernant SQLite.

Stack Overflow :

* Lien : <https://stackoverflow.com/>
* Description : Forum de questions-réponses pour les développeurs, ou vous pouvez trouver des discussions sur les problèmes de sécurité.

GitHub :

* Lien : <https://github.com/>
* Description : Plateforme de développement collaborative ou les développeurs partagent du code et discutent de problèmes de sécurité, incluant des projets open source et des rapports de vulnérabilités.

1. **Critères de sélection des sites**

Fiabilité :

* Objectif : s'assurer que les informations proviennent de sources dignes de confiance et reconnues dans le domaine de la sécurité et de l’IA.

Application :

* Privilégier les sites officielles des organisations de sécurité (OWASP, NIST,CERT) et les publications scientifiques évaluées par des pairs.
* Éviter les sources non vérifiées; les blogs personnels sans expertise reconnue et les forums ou les informations peuvent être erronées ou obsolètes.
* Vérifier la réputation des auteurs et des organisations qui publient les informations.

Pertinence :

* Objectif : S’assurer que les informations sont directement applicables à mon application Streamlit et à l’utilisation de l’IA.

Application :

* Se concentrer sur les sites qui traitent spécifiquement de la sécurité des applications web développées avec Streamlit et de vulnérabilités liées à l’IA.
* Rechercher des informations sur les techniques de validation de code IA, la protection des données dans les applications IA et les meilleures pratiques de sécurité pour le développement IA.
* Éviter les sites qui traitent de la sécurité de l’IA de manière trop générale ou qui ne sont pas pertinents pour le contexte de mon application.

1. **Solution et mise en œuvre**

Validation et sécurisation du code généré par l’IA :

* Examen manuel approfondi du code généré par l’IA pour identifier les vulnérabilités potentielles.
* Utilisation d’outils d’analyse de code statique et dynamique pour détecter les failles de sécurité.
* Mise en œuvre de tests de sécurité rigoureux pour valider la robustesse de l’application.

Mesures de sécurité spécifiques :

* Hachage des mots de passe avec un sel aléatoire et algorithme robuste.
* Utilisation de paramètres préparés pour les requêtes SQL afin de prévenir l’injection SQL.
* Échappement rigoureux des données utilisateur pour prévenir les attaques XSS.
* Mise en œuvre d’une gestion des sessions sécurisée côté serveur.
* Politique de confidentialité claire et transparente pour informer les utilisateurs sur la collecte et l'utilisation de leurs données.

Surveillance continue et mise à jour :

* Mise en place d’un système de surveillance pour détecter les tentatives d’intrusion et les anomalies.
* Mise à jour régulière des bibliothèques et des outils d’IA pour bénéficier des dernières corrections de sécurité.
* Veille constante sur les nouvelles vulnérabilités et les meilleures pratiques de sécurité.

Résultats :

* Mon application Streamlit a été sécurisée en tenant compte des risques spécifiques liés à l’utilisation d’IA.
* Les données sensibles des utilisateurs sont protégées contre les accès non autorisés.
* L'application respecte les réglementations sur la protection des données.
* Un processus de validation et de sécurisation du code généré par l’IA a été mis en place pour les futurs développements.

**Conclusion**

Cette formation en bootcamp du 13 janvier 2025 au 28 mars 2025, en analyse de données a été une expérience intense et enrichissante, me permettant d'acquérir des compétences essentielles dans ce domaine.

Le projet de développement d’une application Streamlit dédiée à l’analyse de données de jeux vidéo a constitué un défi stimulant, me permettant de mettre en pratique les connaissances acquises tout au long de la formation.

L’utilisation d’outils d’IA pour accélérer le développement a été particulièrement utile, indispensable et instructive, me permettant de découvrir les avantages et les limites de cette approche dans le contexte spécifique de l’analyse de données. J’ai pu constater comment l’IA peut faciliter la création d’interfaces interactives et la visualisation de données complexes.

L’intégration d’une fonctionnalité de gestion des utilisateurs et d’authentification a permis de créer un environnement sécurisé pour les utilisateurs de l’application. J’ai utilisé les fonctionnalités de Streamlit pour créer une interface conviviale pour l’ajout et la gestion des utilisateurs.

Ce projet m’a permis de développer des compétences précieuses en matière de développement d’applications Streamlit pour l’analyse de données de jeux vidéo. J’ai appris à concevoir et à mettre en œuvre des interfaces interactives, à traiter et à visualiser des données complexes, et à travailler de manière autonome dans un environnement intensif.

J’ai également pris conscience de l’importance de l'expérience utilisateur dans la conception d’applications d’analyse de données. Je suis convaincu que les compétences acquises au cours de cette formation me seront utiles dans ma future carrière en tant qu’analyste de données.

En conclusion, cette formation en bootcamp a été une expérience transformatrice, me permettant d’acquérir rapidement des compétences techniques et professionnelles essentielles. J’ai pu constater le potentiel de Streamlit pour la création d’applications d’analyse de données de jeux vidéo, et je suis enthousiaste à l’idée d’appliquer ces connaissances dans mes futurs projets.

**Annexes**

Détails du programme de formation Data Analyste :

* Manipulation de données avec Pandas.
* Visualisation de données avec Matplotlib, Seaborn et plotly.
* Analyse statistiques et modélisation prédictive.
* Développement d’application web interactives avec Streamlit.

Compétences acquises :

* Programmation Python pour l’analyse de données.
* Création de visualisations de données interactives.
* Développement d'applications web avec Streamlit.
* Utilisation de bases de données SQLite.
* Gestion de la sécurité des données (hachage de mots de passe, prévention des injections SQL)

Outils et technologies utilisés :

* Streamlit
* Pandas
* Matplotlib
* Seaborn
* Plotly
* SQLite3
* Hashlib

Données et visualisations supplémentaires :

Jeu de données utilisé : **“vgsales\_cleaned.csv”**

Visualisations supplémentaires :

