Rapport PPII - Groupe 29
Application de détection de fuites de données : OopsHunter

Membres du groupe : ALLOMBERT-BLAISE Oscar - GODAIL-FABRIZIO Giuliana - KAMAL JIT Navjit - RAMBEAUX Erwann

Principe de l'application :

Cette application Web permet de détecter des fuites de données sensibles contenues dans des documents de différents types (pdf, xlsx, txt, …). L'utilisateur télécharge ses fichiers sur le site, et il peut ensuite obtenir des rapports de fuite de données directement en un clic. Ces rapports comprennent les types de données qui ont fuitées (carte bancaire, téléphone, …), ainsi que les données en elles-mêmes. Il a également accès à un historique des analyses effectuées, et les documents téléchargés peuvent être visualisés et filtrés.

L'application comporte aussi une partie administration, afin de gérer les informations des utilisateurs. On peut modifier leur adresse, leur e-mail, leur département, etc. Un système d'authentification via des comptes permet aussi de savoir qui a téléchargé quel document, afin de déterminer un score de fuite de données sensibles à chaque utilisateur.

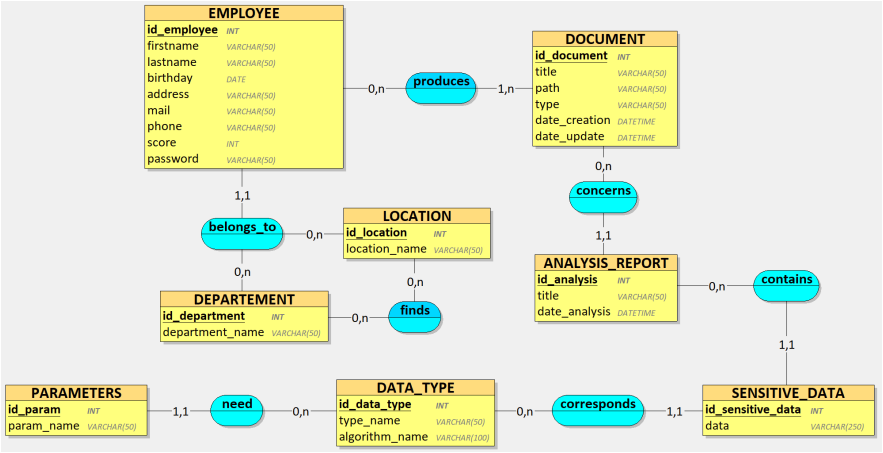
Nom de l'application :

Nous avons choisi d'appeler cette application OopsHunter. Le terme "Oops" fait référence aux erreurs commises par les employés en divulguant des données sensibles. Et le terme "Hunter" provient du principe de l'application : "Chasser" les fuites de données (les "Oops") présentes dans les documents.

Choix techniques pour l'implémentation :

Base de données :

Avant de commencer à implémenter l'application, nous avons logiquement commencé par établir le schéma de notre modèle de données. Nous avons utilisé Looping pour réaliser le schéma entité-association de la base de données, avant de la normaliser en 3NF et de l'implémenter sur sqlite.



L'application n'étant pas hébergée sur un serveur, les documents seront stockés localement dans le répertoire de l'application. Mais il est important de stocker les informations concernant ces documents dans la base de données. C'est pour ça que nous avons mis en place la table DOCUMENTS. Elle permet de stocker la date de création, son titre, son chemin relatif dans le répertoire de l'application, etc.

La table ANALYSIS_REPORT permet de stocker les rapports d'analyses générés après la fouille de données. Elle référence le document qu'elle concerne, et les données trouvées lors de l'analyse sont stockées dans la table SENSITIVE_DATA. Enfin, la table DATA_TYPE permet de stocker les différents types de données, ainsi que la manière dont ce type de données doit être recherché (nom de fonction + paramètres, stockés dans la table PARAMETERS). Cela permet de pouvoir ajouter de nouveaux types de données sans modifier le code (en utilisant des regex, ou des mots-clés fixes).

La table EMPLOYEE sert à stocker les informations concernant les utilisateurs de l'application (les employés de la "Firme"). Cela permet à l'algorithme de détection de fuites de données sensibles de connaître les données des employés et donc de pouvoir les détecter. Les employés sont reliés à des départements (table DEPARTEMENT), qui sont eux-mêmes reliés à des régions (table LOCATION), avec la table finds. Les employés peuvent être auteur ou co-auteur de documents (table produces).

Algorithmes de détection de fuite de données sensibles:

Avant d'implémenter ces algorithmes, nous avons réalisé un état de l'art des solutions existantes sur le marché pour voir quels sont les types des données recherchées et comment cela est fait. Nous avons aussi comparé l'efficacité des différents algorithmes de recherche de chaînes de caractères, et de fonctions comprises dans des librairies comme phonenumbers (voir diapo de la première présentation).

Une fois ce travail effectué, nous avons mis en place les algorithmes nécessaires à l'analyse des documents. Pour trouver des chaînes de caractères qu'on connaît à l'avance (noms / prénoms des employés, des mots-clés choisis par l'entreprise, etc.), nous utilisons simplement la fonction "in" présente nativement dans Python.

Pour trouver des données uniquement par leur format sans les connaître (emails, cartes bancaires, IBAN, numéros de sécurité sociale, etc.), nous utilisons des expressions régulières avec le module re Python.

Pour les numéros de téléphones, nous utilisons la librairie phonenumbers, qui permet de les rechercher de manière efficace, tout en vérifiant leur validité.

Enfin, pour les numéros de cartes bancaires, nous utilisons l'algorithme de Luhn pour vérifier leur validité.

Pour les fichiers PDF, nous utilisons la librairie (et le logiciel) tesseract, permettant de faire de l'OCR, afin de les rendre lisibles par notre programme si ceux-ci ne le sont pas.

Réalisation des pages Web / de la logique backend :

Comme le sujet l'exige, le site Web est conçu grâce au framework Flask, qui permet de réaliser des applications Web avec Python. Nous avons utilisé HTML, ainsi que le framework CSS Bootstrap, afin de réaliser le style des pages plus facilement.

Pour le backend, nous avons utilisé des controllers, comme vu en cours, en utilisant des Blueprints. Cela permet de séparer proprement les différentes fonctionnalités de l'application dans le code, et de faire facilement des modifications. Nous avons aussi séparé les requêtes faites à la BDD dans un dossier queries, et cela utilise le module sqlite3.

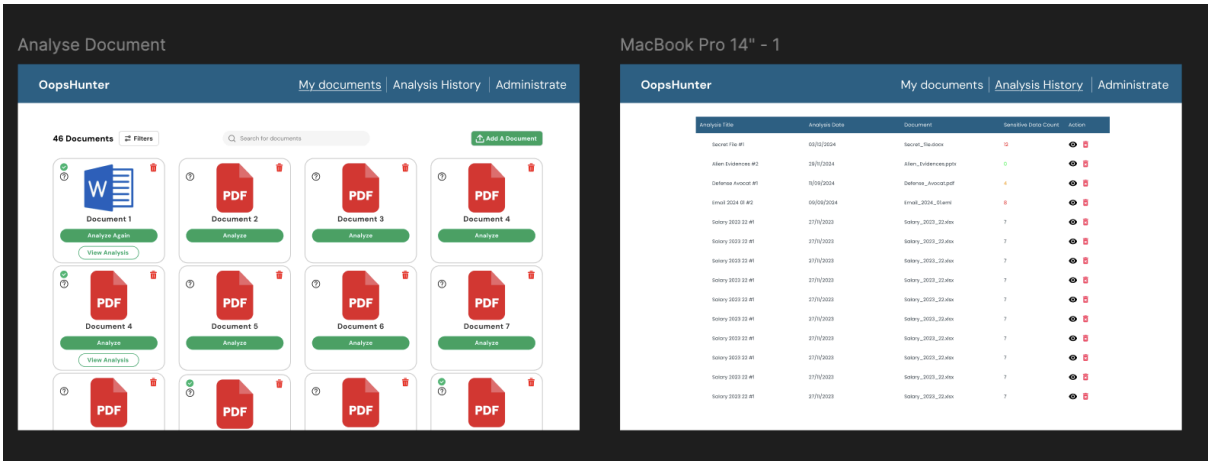
Pour l'authentification, nous utilisons un décorateur login_required qui permet de rediriger sur tous les chemins du site vers la page de connexion si jamais l'utilisateur n'est pas connecté (absence d'id dans la session).

Gestion de Projet :

Pour gérer la planification du projet, les différentes tâches et leur attribution, nous avons utilisé Asana. C'est un outil en ligne simple d'utilisation, qui permet de facilement visualiser les tâches restantes et leurs deadlines.

Avant de commencer le projet, nous avons réalisé une fiche projet, permettant de définir les attendus, les technologies qu'on va pouvoir utiliser, etc.

Avant de réaliser le site, nous avons également conçu une maquette sur Figma, afin de visualiser le site et ses fonctionnalités, pour avoir une première trame et être sûr de rien oublier.



Durant toute la réalisation du projet, nous avons rédigé des comptes rendus après chaque réunion, afin de synthétiser les avancées, les discussions et les tâches à réaliser à chaque étape de l'implémentation.

Répartition des tâches :

Membre du groupe	**Tâches réalisées**
Erwann	- Réalisation de certaines pages de la maquette - Développement du back-end permettant de réaliser les analyses - Travail sur la gestion des types de données - Création des documents de tests
Giuliana	- Mise en place de la base du projet sur Git - Contribution à la conception du modèle conceptuel de données - Écriture du script de création de la base de données - Création d'un jeu de données pour les tests - Connection entre la base de données et le frontend - Réalisation partielle du style de l'application - Réalisation de la barre de navigation - Développement d'une page d'erreur - Développement du frontend pour afficher les analyses - Implémentation de la page pour afficher des documents - Ajout de la possibilité de filtrer les documents et inclusion des filtres dans l'URL - Implémentation des fonctionnalités de gestion des documents : ajout, suppression et téléchargement
Navjit	- Première ébauche de la page authentification - Afficher les employés, leurs informations personnelles et leur score, fonctions python pour

	ajouter, modifier des employées et supprimer des employés. - Afficher les data type, leur nom, nom de l'algorithme et première ébauche pour afficher des paramètres. Ajouter, modifier des data types et supprimer des data types. - Formulaire pour ajouter/modifier un employé et ses informations depuis le compte d'administration. - Formulaire pour ajouter/modifier un data type et des informations depuis le compte d'administration. - Première ébauche du style de base du site - Participation à la conception du modèle conceptuel de données
Oscar	- Réalisation des comptes-rendus de réunion / participation à la réalisation de la fiche projet - Réalisation de certaines pages sur la maquette - Travail sur la base de données - Lien entre le backend et le frontend pour les analyses - Système d'authentification - Page de rapport d'analyse - Multiples correctifs

Analyse Post-Mortem :

Atteinte des objectifs :

Objectif	**Niveau d'atteinte**
Modèle de données qui correspond à la description du projet et normalisé en 3NF	**Complètement atteint** La base de données est normalisée en 3NF, et permet de représenter pertinemment les attentes du sujet.
Interface utilisateur pertinente, ergonomique et fluide	**Complètement atteint** L'interface est fluide, compréhensible, et elle correspond aux attentes du sujet.
Structure du code propre, et respect des bonnes pratiques de programmation	**Complètement atteint** Commentaires, séparation des fonctionnalités en controllers, isolation des requêtes BDD

Algorithmes de fouille de données correctement implémentés, performants et qualitatifs	**Complètement atteint** Les algorithmes répondent aux attentes du sujet, et permettent une expérience utilisateur rapide.
Gestion du personnel et de leurs droits qui correspond à toutes les attentes du sujet	**Partiellement atteint** La gestion des informations personnelles des employés est implémentée. Mais il manque un système de droits selon le département des employés, qui leur donnerait des autorisations de divulguer certaines données.
Réalisation de tests unitaires qui englobent tout le périmètre de l'application et qui assurent sa robustesse	**Pas atteint** Absence de tests unitaires
Gestion de projet durant toute la durée du projet, documentée.	**Complètement atteint** Réalisation d'une fiche de projet, de comptes-rendus de réunion, d'une maquette, et utilisation d'Asana pour planifier et répartir les tâches.

Commentaires personnels :

Giuliana :

"Ce projet m'a permis de revoir des notions étudiées lors de ma première année en BUT Informatique.
J'ai particulièrement apprécié travailler au sein de cette équipe, car nous étions bien organisés dans l'ensemble et la communication était fluide. L'utilisation d'Asana et la création de branches sur Git ont facilité la gestion de projet et le développement collaboratif tout au long du cycle de notre travail.

Concernant les points d'amélioration possibles, je pense qu'il serait pertinent de gérer les accès (aux pages / aux actions de suppression, de modification...) en fonction du département auquel appartient chaque employé. De plus, intégrer un système de filtres pour l'affichage des analyses ou des employés pourrait enrichir l'expérience utilisateur. Enfin, je pense qu'implémenter un dashboard synthétisant les résultats des analyses pour chaque département ou employé pourrait être intéressant."

Navjit :

“Ce projet m'a permis de comprendre comment fonctionne de façon générale une application web. J’ai pu appréhender l’importance de l’utilisation de Flask, le système de routage, l'implémentation d'une page web et l'ajout du style. Ce premier projet a été utile pour voir comment se passe concrètement la gestion de projet lors d'un projet (discussion sur les différents aspects du sujet, des différentes attentes et approches,etc). C'était intéressant de lister les tâches, se répartir le travail et la collaboration pour résoudre les problèmes ont été particulièrement enrichissantes et formatrices. Je pense qu'il y a plusieurs pistes d'amélioration : tout d'abord nous pouvons modifier l'application web de sorte que seul les personnes appartenant au département des ressources humaines aient accès au compte administrateur ce qui sous-entends également la modification de la barre de navigation du site pour que la rubrique administration n'y appraisse que lorsque c'est nécessaire. Il serait également pertinent d’introduire un système de gestion d'accès de données pour qu'une personne ne puisse voir que les documents qui concernent son département. Il est également intéressant d'ajouter des fonctions de filtres pour les employés pourrait améliorer l’expérience utilisateur. Enfin, l'ajout d’un tableau de bord dynamique peut renforcer la surveillance et la sécurité des données car permettrait de suivre en temps réel. Ce dashboard devrait afficher le taux de fuite de données dans l'entreprise et les personnes avec les scores les plus élevés afin de garder un œil sur leur activité. Ce genre de système permettra à l'entreprise de surveiller la sécurité de ses données et d'identifier les éventuels "coupables" ou de savoir si un (des) compte(s) a (ont) été victime(s) d'attaque ainsi une tierce personne serait en réalité coupable des fuites de données. Donc c'est un bon outil pour identifier rapidement des comportements.”

Oscar :

“J’ai apprécié travailler avec ce groupe, la communication était bonne et les tâches ont bien été réparties. L’utilisation d’Asana pour la gestion de projet nous a permis de planifier correctement le travail. Je connaissais déjà Flask mais j’ai pu m’améliorer en appliquant mes connaissances de BUT Informatique pour ce projet.

Pour ce qui est des axes d’améliorations, je pense qu’il faudrait mettre en place des tests unitaires, afin de vérifier la robustesse de l’application. De plus, la gestion des droits des employés aurait pu être améliorée, en donnant des privilèges à certains départements comme les ressources humaines. L’historique des analyses pourrait aussi être amélioré, avec des filtres par exemple ou une fonction de recherche.”

Erwann :

“Ce premier projet m’a permis de revoir les bases du développement web que j’avais déjà abordées lors de mon BUT. Le projet a été réalisé dans une très bonne ambiance, où chacun a pu travailler sur ce qui lui plaisait le plus, tout en ayant l’occasion de toucher à d’autres aspects. La gestion du projet s’est avérée efficace,

malgré un rush final. Dans l'ensemble, tout s'est assez bien déroulé. La communication au sein de l'équipe a été efficace et précise ce qui a permis un bon développement du projet. La création de maquettes nous a permis de nous mettre rapidement d'accord sur le style graphique et d'avoir un objectif clair. Cependant, l'application pourrait être améliorée, notamment en ce qui concerne une gestion plus poussée des comptes utilisateurs/administrateurs, ainsi que l'ajout d'options permettant de choisir les types de données à analyser ou non. Le visuel pourrait également être revu, pour l'ajout d'un type de données. Je suis tout de même satisfait du rendu final, et surtout de l'implication de chacun dans ce projet."

Sources utilisées pour la réalisation du projet :

https://getbootstrap.com/ : Documentation qui a permis d'apprendre à utiliser Bootstrap
https://flask.palletsprojects.com/en/stable/ : Documentation de Flask
https://tesseract-ocr.github.io/ : Documentation de Tesseract
https://docs.python.org/3/ : Documentation de Python
https://www.conventionalcommits.org/en/v1.0.0/ : Pour apprendre à faire des messages de commit
Sources pour l'état de l'art dans les liens hypertextes de celui-ci.