Ecole Publique d'Ingénieurs en 3 ans

Rapport de Stage

CONCEPTION D'UNE APPLICATION WEB POUR L'EXPLOITATION DE BAGUETTE : BEHAVIORAL ANALYSIS GRAPH USING EXECUTION TRACES TOWARDS EXPLANABILITY

Date de soutenance

25/08/2023

Aymane El Otmani,
Année Universitaire 2022/2023

INFORMATIQUE / E-PAIEMENT ET CYBERSECURITE

Tuteur entreprise : **Valérie Viet Triem Tong**
Tuteur ENSICAEN : **Morgan Barbier**

ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

Inria

CentraleSupélec

www.ensicaen.fr

TABLE DES MATIERES

Remerciements

L'opportunité de stage au sein de l'équipe de Centrale Supélec de Rennes en collaboration avec Inria a été une expérience d'apprentissage exceptionnelle. Je suis reconnaissant d'avoir eu l'occasion de travailler avec des personnes aussi formidables. J'aimerais exprimer ma gratitude et mes remerciements spéciaux à mon tuteur, Morgan Barbier, qui malgré ses nombreuses responsabilités, a pris le temps de rester en contact avec moi tout au long du stage, en me demandant comment cela se passait. Sa disponibilité et son soutien m'ont grandement aidé à avancer dans mon projet.

J'adresse également mes remerciements à Valérie Viet Triem Tong, responsable administrative, ma tutrice entreprise et professeure à Centrale Supélec. Elle était toujours en contact avec moi, veillant à me présenter aux personnes pouvant m'aider tout au long du stage avec leur expertise. Sa générosité et sa flexibilité m'ont permis de bénéficier d'un environnement propice à l'apprentissage et au développement de mes compétences.

Un grand merci à Vincent Raulin, Doctorant et créateur de l'outil "BAGUETTE" Behavioral Analysis Graph Using Execution Traces Towards Explainability. En tant que chef du projet, il a généreusement consacré de son temps pour m'aider à intégrer mon projet à son outil. Sa disponibilité et sa collaboration ont été précieuses pour la réussite de mon stage.

Je tiens également à exprimer ma reconnaissance envers Pierre-François Gimenez pour sa collaboration précieuse durant l'absence de ma tutrice. Sa disponibilité et son engagement ont été d'une grande aide pour maintenir la fluidité de mon travail.

Enfin, je souhaite exprimer ma gratitude envers Pascale Dalleau, ma marraine, qui m'a apporté un soutien moral aussi bien que financier pour mon installation à Rennes et pendant toute la durée de mon séjour dans cette ville.

Je tiens également à exprimer ma profonde gratitude envers l'équipe CIDRE, ainsi qu'envers les membres de Centrale Supélec et d'Inria, avec qui j'ai eu le privilège de collaborer. En particulier, je tiens à remercier chaleureusement ceux qui ont joué un rôle essentiel dans mon apprentissage et dans la progression de mon projet. Vos précieux conseils, votre expertise inégalée et votre soutien indéfectible ont constitué les piliers de mon expérience de stage, apportant une valeur inestimable à ma formation.

1. Introduction

En tant qu'étudiant en école d'ingénieur, spécialisé en e-paiement et cybersécurité, j'ai toujours été animé par une passion profonde pour le domaine de la cybersécurité. Conscient de son importance cruciale dans notre monde moderne, cette passion m'a constamment poussé à rechercher des opportunités pour approfondir mes connaissances et compétences dans ce domaine en constante évolution.

C'est avec une grande reconnaissance et un profond respect que j'ai accueilli l'opportunité offerte par l'INRIA et CentraleSupélec, deux institutions de renom. Elles m'ont confié une mission sérieuse et exigeante : développer un site web pour l'outil "BAGUETTE". L'objectif était clair et précis : concevoir une interface utilisateur intuitive et efficace, permettant d'exploiter aisément les fonctionnalités avancées de cet outil, tout en mettant en avant sa page d'accueil pour assurer une expérience utilisateur optimale.

Cette expérience m'a offert une occasion inestimable d'approfondir mes compétences en développement web, couvrant à la fois le frontend et le backend, ainsi que la gestion des bases de données. Chaque étape du projet a renforcé ma compréhension technique, m'incitant à réaliser l'importance de la collaboration, de la communication au sein d'une équipe, et de la capacité à comprendre et répondre aux besoins des utilisateurs.

En somme, ce stage a été bien plus qu'une simple expérience professionnelle ; il a été un voyage d'apprentissage et de découverte, renforçant ma détermination à poursuivre une carrière dans le domaine de la cybersécurité avec compétence, confiance et dévouement.

2. Présentation des établissements

2.1. INRIA

L'INRIA, Institut de renommée internationale, est un fer de lance dans la recherche scientifique et technologique dans le domaine du numérique. Avec plus de 3 900 chercheurs et ingénieurs, répartis en 220 équipes-projets, l'INRIA explore des pistes innovantes en collaboration avec des partenaires industriels et académiques. La sécurité numérique constitue un pilier central de la vision stratégique de l'INRIA. Sa collaboration étroite avec des acteurs clés tels que l'ANSSI (Agence nationale de la sécurité des systèmes d'information) témoigne de son engagement envers la construction d'une société numérique de confiance. Dans un contexte où la cryptographie post-quantique, l'intelligence artificielle et la protection des données deviennent des enjeux critiques, l'INRIA joue un rôle majeur dans le renforcement de la recherche en cybersécurité. [3][4]

2.2. Centrale Supélec

Centrale Supélec, quant à lui, est un établissement de premier plan, reconnu pour son environnement propice à l'innovation et à la recherche avancée. En synergie avec l'INRIA, Centrale Supélec partage un intérêt marqué pour la cybersécurité dans l'ère numérique. Ensemble, ces institutions abordent avec sérieux et expertise les défis complexes de la sécurité informatique, contribuant ainsi à préserver la sécurité des systèmes numériques et à anticiper les évolutions technologiques futures.

3. Présentation du sujet

3.1. Contexte : Risque des cyberattaques

De nos jours, le monde numérique est confronté à une augmentation exponentielle des cyberattaques. Chaque année, de nouveaux logiciels malveillants voient le jour, rendant la cybersécurité plus cruciale que jamais. Des malwares tels que les virus, les ransomwares, les chevaux de Troie et les logiciels espions continuent de menacer la sécurité des données et des infrastructures à l'échelle mondiale. Dans ce contexte, l'analyse des logiciels malveillants est devenue une nécessité pour comprendre, détecter et contrer ces menaces. [5]

3.2. Nécessité d'un outil d'analyse pour les experts en cybersécurité :

L'analyse des logiciels malveillants consiste à étudier un échantillon de code suspect pour le comprendre et produire une représentation ou une explication de ce code. Cette représentation est cruciale pour les experts humains ou les outils automatisés afin de détecter et de contrer efficacement les menaces. Cependant, de nombreuses analyses produisent des rapports volumineux qui peuvent être difficiles à interpréter rapidement. [5]

3.3. Rapports Cuckoo

L'une des méthodes d'analyse dynamique les plus populaires est le rapport Cuckoo. Cette méthode est reconnue pour la qualité des données qu'elle génère. Elle offre une capacité de détection élevée et une robustesse face aux techniques d'évasion. Le terme 'analyse dynamique' se réfère à la méthode d'analyse de logiciels malveillants qui s'exécute réellement sur un environnement de test pour observer le comportement du logiciel. Cette méthode est largement utilisée pour évaluer la sécurité des logiciels et détecter les menaces potentielles. Cependant, malgré son efficacité, le rapport Cuckoo peut être volumineux et complexe à analyser. [5]

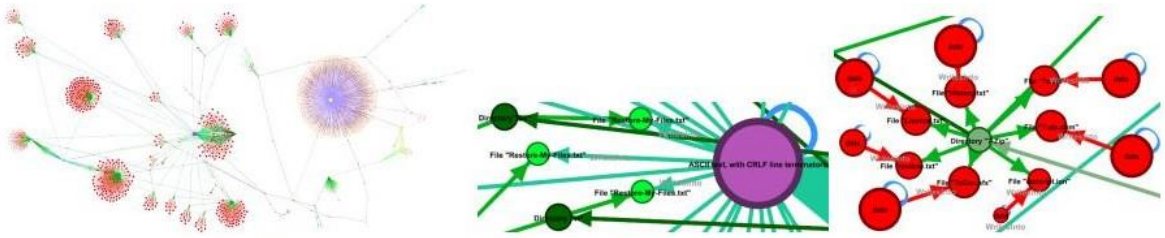
3.4. BAGUETTE – Une solution innovante

Face à ces défis, un nouvel outil nommé BAGUETTE a été développé. BAGUETTE est un post-traitement du rapport d'analyse dynamique produit par Cuckoo sandbox reposant sur un graphe hétérogène. Il est proposé comme un cadre basé sur des graphes hétérogènes qui tente de capturer autant que possible l'effet du logiciel malveillant sur le système tout en restant lisible à la fois par les approches basées sur ML et par les analystes humains.

Un graphe BAGUETTE montre les actions que les échantillons de logiciels malveillants ont prises sur les ressources du système, telles que les fichiers et les trafics réseau. Il permet aux analystes humains de signaler et de comprendre les charges utiles malveillantes des échantillons de logiciels malveillants.

BAGUETTE est défini comme un graphe hétérogène contenant 14 types de nœuds, chacun ayant ses propres attributs. Les arêtes et les flèches sont également typées. Le nœud racine d'un graphe BAGUETTE est un nœud Hôte. Il représente une machine physique qui démarre des processus, se connecte à d'autres machines et possède un système de fichiers. L'annexe 10 représente la structure d'un graphe Baguette et les types de nœuds et d'arrêtes. La figure 1 montre des images des graphes baguettes visualisé par GEPHI.

L'objectif de BAGUETTE est non seulement d'aider à la recherche automatique de comportements spécifiques dans une base de données de logiciels malveillants et d'assister efficacement l'expert dans l'analyse des échantillons mais aussi de créer un antivirus dynamique puissant et efficace. [5]



(a) Overview of the graph (b) Ransom dropping behavior (c) File system encryption behavior

Figure 1 Vue sur les graphes générés par BAGUETTE

3.5. Travail demandé

L'outil "BAGUETTE" propose des fonctionnalités avancées dans le domaine de la cybersécurité. Parmi ses capacités distinctives, deux fonctionnalités se démarquent :

1. **Bake** : Cette fonctionnalité permet de construire un graphe BAGUETTE à patrir d'un rapport Cuckoo.

2. **Toast** : Toast est une fonctionnalité avancée qui permet à l'utilisateur de saisir un métagraphe représentant un comportement spécifique. L'outil recherche ensuite ce comportement au sein des malwares présents dans la base de données, renvoyant le nombre de correspondances trouvées.

Cependant, malgré ses capacités impressionnantes, l'utilisation de BAGUETTE peut s'avérer complexe pour certains utilisateurs. Dans cette optique, ma mission a été clairement définie : développer un site web facilitant l'utilisation de la fonctionnalité "Toast".

Objectifs du site web :

- **Interface utilisateur intuitive** : Permettre aux utilisateurs de tracer un graphe facilement, offrir des outils d'aide pour créer, mettre à jour ou supprimer des nœuds (vertex) et des arêtes (edges/arrows). Il est essentiel de garantir que chaque type d'arête respecte les contraintes de nœuds source et cible.

- **Gestion des graphes** : Offrir la possibilité d'exporter le graphe au format JSON pour des utilisations ultérieures et d'importer des graphes préexistants. Cette fonctionnalité doit être sécurisée pour éviter l'introduction de code malveillant qui pourrait compromettre le système ou la base de données.

- **Gestion des comptes** : Les utilisateurs peuvent soit s'inscrire et se connecter, soit continuer en tant qu'invité. Les utilisateurs inscrits bénéficieront de la possibilité de sauvegarder leurs graphes dans la base de données du site, leur permettant de reprendre leur travail à tout moment.

- **Recherche de comportements malveillants** : Une fois le métagraphe créé, les utilisateurs peuvent rechercher des malwares dans la base de données qui présentent ce comportement spécifique et obtenir une liste des correspondances.

- **Page d'accueil et contact** : Le site disposera d'une page d'accueil présentant l'outil BAGUETTE et ses capacités. De plus, une rubrique de contact sera mise à disposition pour toute demande ou feedback.

4. Etude de l'existant

Lors de mon intégration à l'équipe, ma première mission était de construire le site web à partir de zéro. Sans infrastructure préexistante, j'ai entamé le développement en prenant le temps de me familiariser avec les outils et logiciels liés au projet.

Parmi ces ressources, "BAGUETTE", codé principalement en Python, s'est avéré être un élément central. Afin de m'assurer que le site web serait en adéquation avec les spécificités de ce logiciel, j'ai consacré une semaine à l'utiliser en mode interactif, ce qui m'a permis de comprendre ses fonctionnalités et ses capacités.

La figure 2 démontre mon processus de familiarisation avec Baguette et la création d'un métagraphe, un élément essentiel de notre projet qui permet de représenter des données complexes et leurs relations de manière structurée : dans le code on crée un metagraphe avec deux nœuds. Le premier nœud est un File et le second est un Diff ensuite on crée une arête qui les lie et enfin une condition sur le nœud diff.

```
>>> MG = MetaGraph()     # Empty metagraph

>>> MG.file = MetaVertex[filesystem.File]    # First vertex

>>> MG.diff = MetaVertex[data.Diff]          # Second vertex

>>> MG.writes = MetaEdge(MG.file, MG.diff)[data.IsDiffOf]    # Edge between
them

>>> MG.diff.condition = Evaluator("x.written_entropy >= 6") # Condition on
one vertex.
```

Figure 2 Exemple d'utilisation du mode interactive de BAGUETTE

5. Travail personnel effectué

5.1. Architecture générale de la page de l'outil du traçage

Dans le cadre de mon projet, j'ai choisi d'utiliser Flask comme framework de backend. Cette décision a été principalement motivée par le fait que l'outil Baguette est entièrement codé en Python. Flask, étant un micro-framework Python, m'a permis de rester proche de l'environnement natif de Baguette, garantissant ainsi une intégration fluide et une cohérence dans le développement. De plus, Flask est reconnu pour sa puissance et sa flexibilité mais aussi par sa capacité à gérer les routes d'API. Il offre une grande liberté dans la structuration du code tout en fournissant toutes les fonctionnalités essentielles pour le développement d'applications web robustes. Cette combinaison de proximité avec Python et de capacités étendues a fait de Flask le choix idéal pour mon backend. [2]

Pour la partie front-end, j'ai principalement utilisé HTML, CSS et JavaScript. De plus, pour améliorer l'esthétique et la réactivité de l'interface utilisateur, j'ai intégré le framework TailwindCSS, qui m'a permis de créer des designs modernes et adaptatifs.

Pour la documentation et la conception de l'API, j'ai utilisé Swagger. Cet outil m'a aidé à définir, concevoir, documenter et gérer les API REST. Grâce à Swagger, j'ai pu garantir une documentation claire et à jour, facilitant ainsi la collaboration avec d'autres développeurs et assurant une meilleure compréhension de l'API pour les utilisateurs finaux.

En ce qui concerne la gestion de la base de données, j'ai opté pour SQLite3 en raison de sa légèreté et de sa facilité d'intégration avec Flask. Pour interagir avec cette base de données, j'ai utilisé SQLAlchemy, un ORM (Object Relational Mapper) qui facilite la manipulation des données. De plus, pour la sérialisation et la désérialisation des objets, j'ai fait appel à Marshmallow, garantissant ainsi une transition fluide entre les objets Python et les réponses JSON.

Les composants spécifiques du diagramme de la figure 3 et leur interaction seront détaillés et expliqués au fur et à mesure des prochains paragraphes.

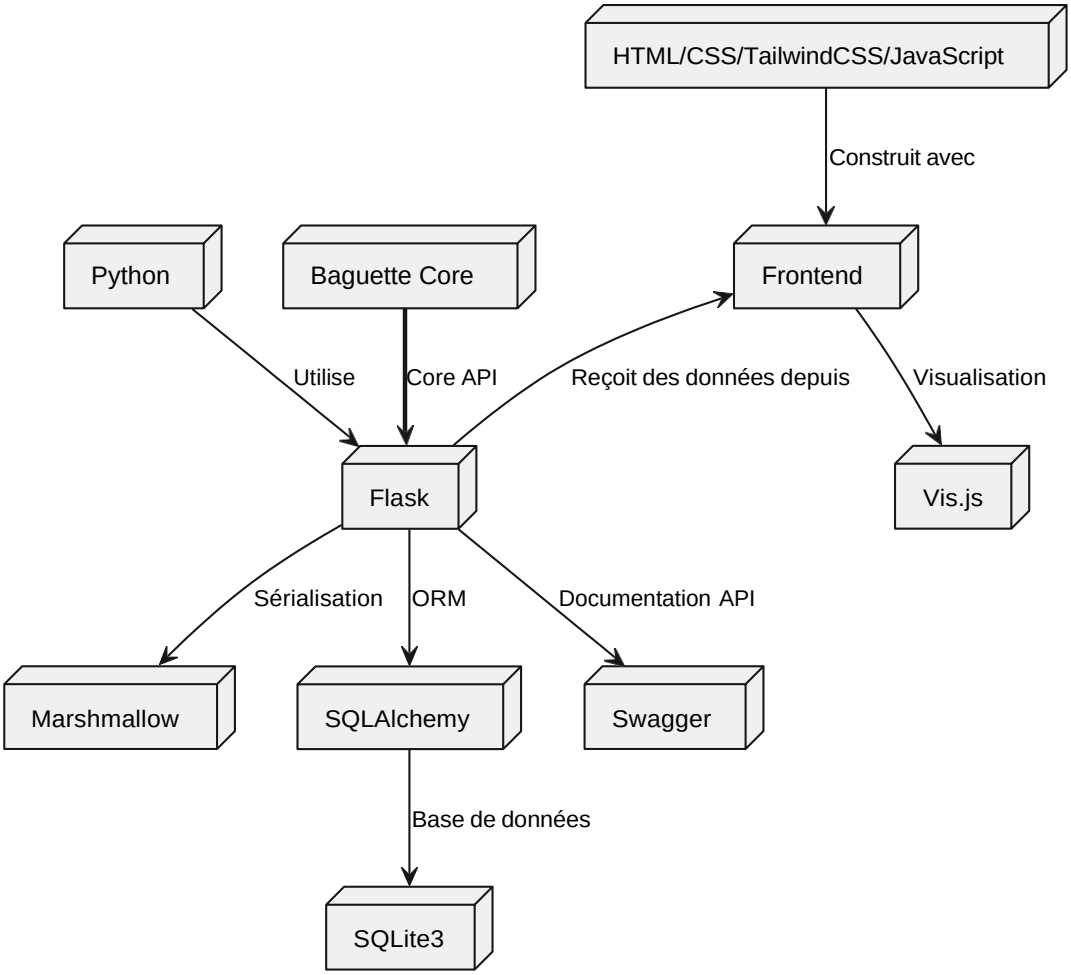


Figure 3 Diagramme de composants représentant les technologies utilisées

La Figure 4 illustre l'architecture du dossier associé à la page de traçage des graphes. J'ai organisé les fichiers JavaScript dans un sous dossier, chaque fichier étant dédié à une fonctionnalité spécifique. Les fichiers HTML sont rangés dans le sous-dossier "templates". Les fichiers HTML commençant par un "_" représentent des segments de la page "drawing baguette" et ont été créés pour éviter un fichier unique trop volumineux. Ces segments sont finalement inclus dans la page"home.html".

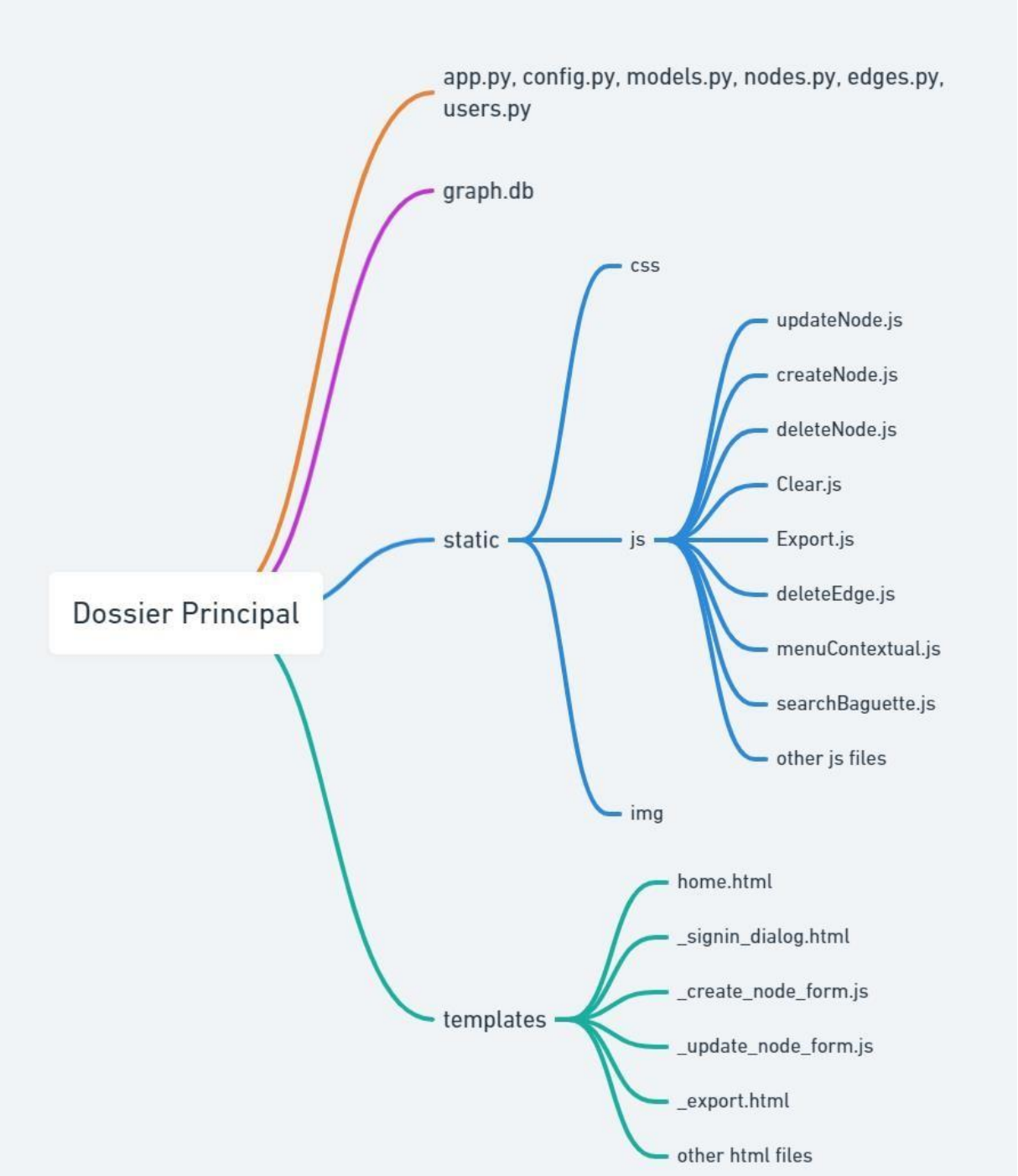


Figure 4 Arborescence représentant la structure du dossier du traçage du graphe

5.2. Base de données

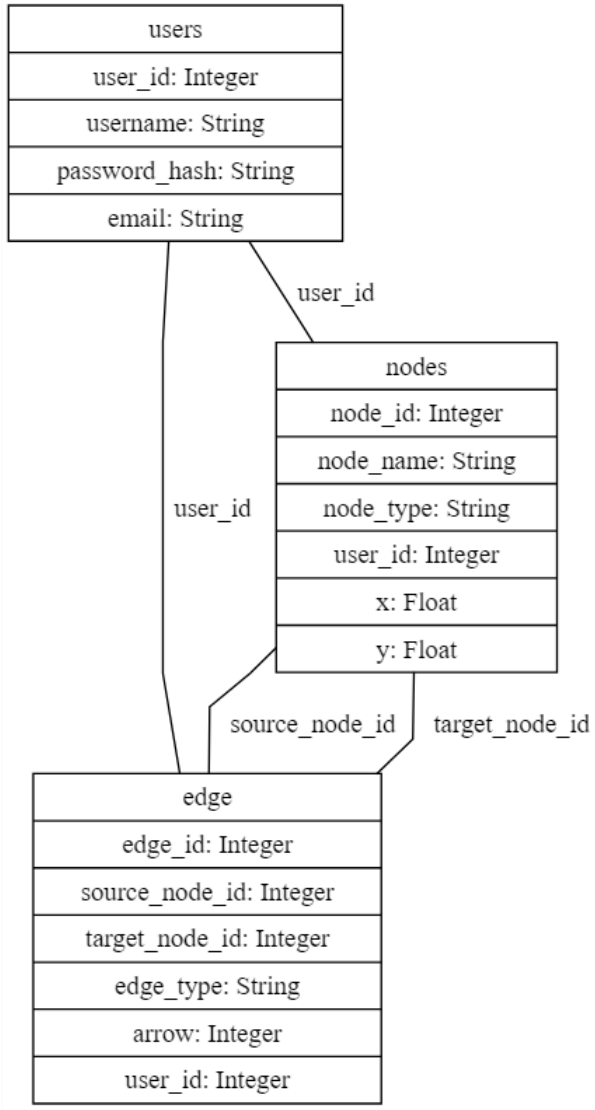


Figure 5 diagramme de base de données

La base de données Graph se compose de trois tables. La table **User** stocke les informations relatives aux utilisateurs enregistrés, avec des détails tels qu'un identifiant unique, un nom d'utilisateur, un mot de passe chiffré et une adresse e-mail. La table **Node** décrit un nœud du système, intégrant un identifiant unique, un nom, un type et des coordonnées x et y. Enfin, la table **Edge** représente les liaisons entre les nœuds, contenant un identifiant unique, des identifiants pour les nœuds source et cible, un type d'arête, une indication si c'est une arête ou une flèche, et un identifiant d'utilisateur associé.

Les outils utilisés pour manipuler la base de données :

Swagger[6] : En intégrant la spécification OpenAPI à l'aide du module Connexion, j'ai pu bénéficier de la robustesse de Swagger pour mon projet.

Cette spécification, reconnue pour sa capacité à décrire les API REST, offre des avantages non négligeables :

- Elle valide systématiquement les données entrantes et sortantes de l'API.
- Elle configure les points de terminaison URL de l'API et détermine les paramètres attendus. Grâce à OpenAPI et Swagger, mon application Flask dispose désormais d'une interface utilisateur (UI) intuitive permettant d'explorer l'API en toute simplicité. Annexe 11

SQLite3[1][6] : J'ai opté pour SQLite comme moteur de base de données pour stocker les données des graphes. SQLite se distingue par sa capacité à fonctionner sans serveur SQL dédié. Sa particularité réside dans l'utilisation d'un unique fichier pour gérer l'ensemble des fonctionnalités de la base de données. Avec le module sqlite3 intégré à Python, l'interaction avec SQLite devient fluide, sans nécessité de recourir à des paquets externes.

SQLAlchemy [6]: SQLAlchemy est au cœur de la gestion de ma base de données. Son ORM permet une interaction élégante avec la table de base de données des graphes. Toutefois, une problématique se pose lors de la communication entre l'API REST et les données JSON. En effet, SQLAlchemy renvoie les données sous forme d'instances de classes Python, ce qui peut poser des problèmes de sérialisation.

Marshmallow [6]: Pour pallier cette difficulté, j'utilise Marshmallow. Ce module m'assiste dans la création d'une classe NodeSchema, EdgeSchema, UserSchema , analogue à celle élaborée avec SQLAlchemy. Outre sa capacité à convertir les attributs d'une classe en formats compatibles avec JSON, Marshmallow joue un rôle crucial dans la validation des données. Il garantit l'intégrité des données en s'assurant de la présence des attributs et de la conformité de leur type. De plus, Marshmallow offre une protection supplémentaire contre les tentatives d'injection de code malveillant, préservant ainsi l'intégrité et la sécurité de la base de données. Par exemple, si une donnée devrait contenir un nombre entier, Marshmallow s'assurera que ce numéro est effectivement un entier, évitant ainsi les erreurs de type de données qui pourraient compromettre l'intégrité des données.

Afin d'illustrer concrètement l'utilisation combinée de SQLAlchemy et Marshmallow dans la gestion des données, la figure 6 présente la déclaration d'un des schémas de Marshmallow, **NodeSchema**, ainsi que la classe **Node** qui représente le modèle pour un nœud en utilisant SQLAlchemy. Cette combinaison permet non seulement une interaction fluide avec la base de données, mais aussi une sérialisation et une validation efficaces des données.

```python
class Node(db.Model):
    __tablename__ = "nodes"

    node_id = db.Column(db.Integer, primary_key=True)
    node_name = db.Column(db.String(32), nullable=False)
    node_type = db.Column(db.String(32))
    user_id = db.Column(db.Integer, db.ForeignKey('users.user_i
    x = db.Column(db.Float)
    y = db.Column(db.Float)

    outgoing_edges = db.relationship(
        "Edge",
        backref="source_node",
        foreign_keys="[Edge.source_node_id]",
    )

    incoming_edges = db.relationship(
        "Edge",
        backref="target_node",
        foreign_keys="[Edge.target_node_id]",
    )
```

```python
class NodeSchema(ma.SQLAlchemyAutoSchema):
    class Meta:
        model = Node
        load_instance = True
        sqla_session = db.session
        include_relationships = True
    outgoing_edges = fields.Nested(EdgeSchema, many=True)
    incoming_edges = fields.Nested(EdgeSchema, many=True)
    user_id = fd.Integer()
```

Figure 6 Modèle et Schéma pour la Gestion des Nœuds

Le code au-dessus détaille la structure et la gestion d'un "nœud" dans une base de données. La classe **Node** est un modèle SQLAlchemy représentant un nœud, avec une table associée

nommée "nodes". Cette table contient plusieurs colonnes, dont une clé primaire **node_id**, un nom de nœud **node_name**, un type de nœud **node_type**, une clé étrangère **user_id** faisant référence à un utilisateur, et des coordonnées flottantes **x** et **y**. De plus, la classe établit des relations avec une table d'arêtes, via **outgoing_edges** et **incoming_edges**, en utilisant des clés étrangères. D'autre part, la classe **NodeSchema** est un schéma Marshmallow pour le modèle **Node**, facilitant la sérialisation et la désérialisation des instances en format JSON. Ce schéma inclut également des relations et assure une validation et un formatage appropriés des données. Ensemble, SQLAlchemy et Marshmallow permettent une gestion robuste des nœuds tout en assurant une interaction fluide avec une API REST.

5.3. Les Fonctionnalités principales

5.3.1. Visualisation de graphes

La représentation visuelle des données est essentielle, surtout pour les graphes. Dans ce projet, j'ai utilisé **vis.js**, une bibliothèque javascript dédiée à la visualisation graphique. Elle offre des fonctionnalités d'interaction directe avec les nœuds et les arêtes notamment le déplacement le zoom, ainsi que des méthodes intégrées pour faciliter la création et la manipulation des graphes. En exploitant vis.js, j'ai pu convertir efficacement les données JSON du graphe en visualisations graphiques interactives, permettant ainsi une analyse et une compréhension optimales des données.[7]

5.3.2. Gestion des comptes

La gestion des comptes sur un site web est une fonctionnalité primordiale, tant pour les propriétaires du site que pour ses utilisateurs. Pour les propriétaires, elle permet de mieux comprendre et segmenter leur audience, d'assurer la sécurité des données et d'offrir des services personnalisés. Pour les utilisateurs, disposer d'un compte offre une expérience personnalisée et la possibilité de sauvegarder et retrouver leurs travaux, en l'occurrence leurs graphes, depuis n'importe quel appareil. C'est une garantie de flexibilité et de continuité dans leur travail. Annexe 3

Cependant, il est également essentiel de prendre en compte les utilisateurs qui préfèrent ne pas s'inscrire ou qui souhaitent simplement tester le service avant de prendre une décision. La fonctionnalité "continuer en tant qu'invité" répond à ce besoin, offrant un accès immédiat et sans friction au site et à ses outils. Néanmoins, pour garantir la pérennité et la sécurité des données, seuls les utilisateurs inscrits peuvent bénéficier d'un stockage durable de leurs graphes. Cela incite non seulement à l'inscription, mais assure également la gestion optimale des ressources du site. Annexe 7

5.3.3. L'importation et l'exportation du graphe JSON

En ce qui concerne les opérations d'importation et d'exportation, j'ai établi un format JSON spécifique. Ce format a été conçu pour garantir une interprétation cohérente lors des importations, en vue de futures utilisations. Voici la structure adoptée :

```
[
  {
    "node_id": "<ID unique du nœud>",
    "outgoing_edges": [
      {
        "edge_id": "<ID unique de l'arête>",
        "arrow": "<Indicateur d'arête ou de flèche>",
        "edge_type": "<Type de relation de l'arête>",
        "source_node_id": "<ID du nœud source>",
        "target_node_id": "<ID du nœud cible>"
      },
      ...
    ],
    "node_name": "<Nom du nœud>",
    "node_type": "<Type du nœud>",
    "x": "<Coordonnée X du nœud>",
    "y": "<Coordonnée Y du nœud>"
  },
  ...
]
```

Des illustrations des dialogues "Export" et "Import Json" sont disponibles dans l'Annexe 2

5.3.4. Sécurité et stockage

Nous avons précédemment abordé la manière dont SQLAlchemy et Marshmallow aident à se prémunir contre les injections de code malveillant, une menace majeure pour la sécurité des bases de données. Cependant, la sécurité ne s'arrête pas à la protection de la base de données. Lors de la gestion des événements tels que l'importation JSON ou la recherche via "search baguette", il est crucial de valider les entrées pour s'assurer de leur intégrité.

Du côté JavaScript, plusieurs vérifications sont effectuées lors de ces événements :

- Vérification que les données importées sont bien au format JSON.
- Vérification que les données importées constituent un tableau.
- Vérification que chaque nœud est un objet (et non un tableau ou null).
- Vérification que chaque objet nœud contient exactement les bonnes clés attendues et que les valeurs correspondent aux types attendus.
- Vérification que les arêtes sont correctement formatées.

- Vérification que le type de chaque arête est valide par rapport à ses nœuds source et cible.

Pour renforcer cette sécurité, ces mêmes tests sont également répétés côté Flask. Cette redondance est essentielle car elle protège contre d'éventuelles attaques qui pourraient désactiver ou contourner les contrôles côté client en JavaScript.

En ce qui concerne le stockage, nos infrastructures ont des capacités limitées. Il n'est donc pas viable d'offrir un stockage illimité des graphes dans la base de données. Pour gérer cela, nous avons décidé de réserver cette fonctionnalité de stockage aux utilisateurs ayant un compte, avec une limitation à trois enregistrements par jour, le choix de trois enregistrement revient en fait aux informations sur le site officiel : Selon le site officiel de sqlite3, un site qui utilise sqlite3 pour sa base de données effectue entre 60 000 et 100 000 modifications. En limitant le nombre d'enregistrements à 3, cela assure qu'entre 20 000 et 33 000 utilisateurs peuvent sauvegarder leurs graphes dans la base de données[1] Cela garantit non seulement une utilisation optimale des ressources, mais incite également les utilisateurs à s'inscrire pour bénéficier de ces avantages.

5.3.5. Optimisation de l'interface utilisateur

Pour garantir une expérience utilisateur optimale, j'ai apporté plusieurs améliorations majeures à l'interface du site. Tout d'abord, j'ai utilisé **TailwindCSS**, un framework CSS hautement personnalisable, qui permet de construire rapidement des interfaces modernes et réactives avec un code épuré. Cela m'a permis d'appliquer un style élégant et fonctionnel à l'ensemble du site.

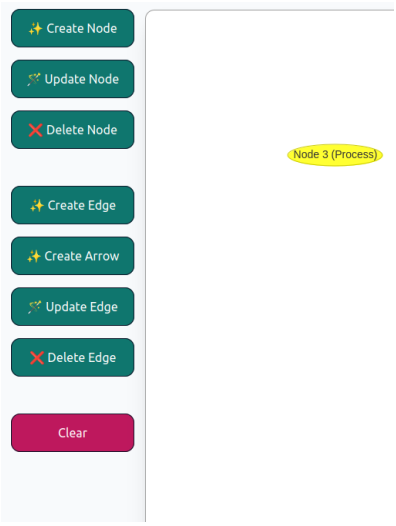


Figure 7 les boutons à gauche du tableau

L'interaction avec les graphes a été rendue intuitive grâce à deux approches : l'utilisation de boutons positionnés à gauche du tableau (voir Figure 7) et l'introduction d'un menu contextuel (voir Figure 8). Ces deux méthodes offrent à l'utilisateur une flexibilité dans la manière de créer et de manipuler les graphes.

J'ai également porté une attention particulière à la structuration des formulaires (Annexe 9 et Annexe 8 par exemple), en utilisant parfois Bootstrap pour une mise en page cohérente et agréable. La création de nœuds et d'autres interactions sont ainsi guidées par des formulaires organisés et intuitifs. De plus, pour informer l'utilisateur et le guider lors de certaines actions, j'ai intégré des notifications et des indications, gérées dynamiquement par JavaScript.

Reconnaissant que certaines fonctionnalités pourraient nécessiter des éclaircissements supplémentaires pour certains utilisateurs, j'ai pris l'initiative d'ajouter un tutoriel, accessible via un lien direct dans l'en-tête du site. Ce guide offre une assistance pas à pas pour ceux qui pourraient rencontrer des difficultés ou souhaitent simplement approfondir leur compréhension des outils disponibles. Annexe 6

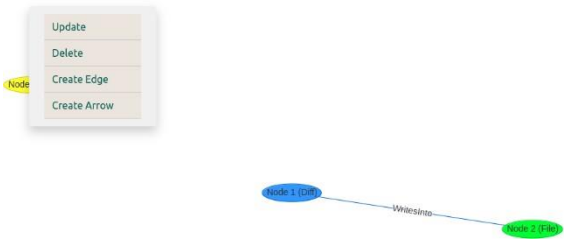


Figure 8 Le menu contextuel

Dans les annexes, vous trouverez des illustrations détaillées des fonctionnalités de la page "Drawing Graph" ainsi que des images de la page d'accueil.

5.3.6. L'envoi des données et leur récupération

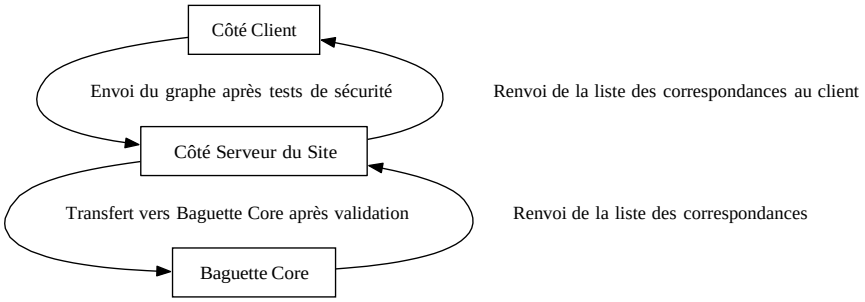


Figure 9 : Processus de Communication et Analyse du Graphe

Côté Client : Avant d'envoyer les données, une série de tests de sécurité est effectuée sur le graphe pour s'assurer de sa validité. Si ces tests sont réussis, une requête est faite vers un endpoint spécifique du serveur principal du site pour transmettre le graphe.

Côté Serveur du Site : À la réception du graphe sur cet endpoint, des tests de sécurité supplémentaires sont effectués pour valider les données. Si ces données sont valides, elles sont converties en JSON et préparées pour être envoyées à un autre composant : Baguette Core. Le transfert vers Baguette Core est effectué en utilisant une requête interne vers un endpoint dédié de ce composant.

Baguette Core : Le graphe reçu sur cet endpoint est converti en une structure compatible avec l'outil Baguette. L'outil Baguette effectue alors une recherche pour trouver des correspondances parmi des échantillons de malwares qui contiennent le graphe dessiné. Une fois la recherche terminée, une liste des correspondances est renvoyée via l'endpoint à partir duquel la requête initiale a été faite.

Retour au Côté Serveur du Site : Le serveur traite la liste des correspondances reçues et les renvoie au client via l'endpoint initial.

Côté Client : À la réception de la liste des correspondances via cet endpoint, celles-ci sont affichées dans un dialogue pour l'utilisateur.

Une illustration de l'affichage des résultats est disponibles dans l'Annexe 1.

5.4. Stratégie du travail – Méthodologie du travail

5.4.1. Méthodologie globale

Tout au long du projet, nous avons adopté une méthodologie de travail qui consiste en des réunions régulières, soit hebdomadaires, soit bimensuelles. Ces sessions étaient cruciales pour évaluer l'avancement du projet, définir des objectifs clairs et prioriser les tâches à venir jusqu'à la prochaine réunion. Cette approche nous a permis de maintenir une direction claire et de gérer efficacement les ressources et le temps. De plus, j'ai gardé un contact constant avec Vincent, ce qui m'a permis de bénéficier de ses conseils et de son assistance chaque fois que j'en avais besoin, garantissant ainsi une progression constante et évitant des blocages inutiles.

5.4.2. Transition vers un traitement côté navigateur

Initialement, chaque opération effectuée sur le graphe, que ce soit la création, la mise à jour ou la suppression d'un nœud ou d'une arête, générait un appel d'API. Cet appel provoquait une modification directe de la base de données pour refléter ce changement. Cependant, cette approche s'est avérée coûteuse en termes d'infrastructure. En effet, chaque modification, aussi minime soit-elle, demandait une interaction avec la base de données, ce qui pouvait potentiellement la solliciter excessivement et risquer de l'endommager à long terme.[1]

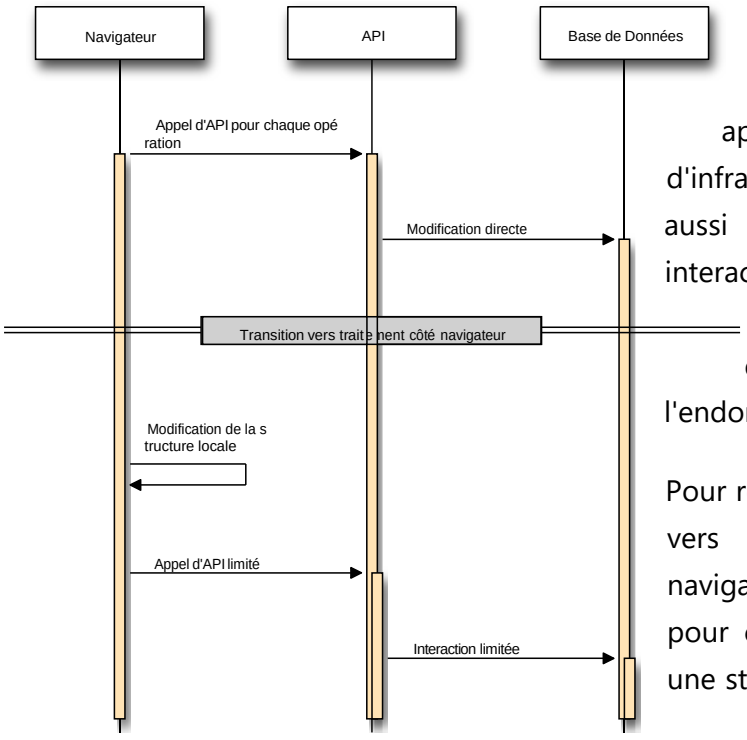


Figure 10 : Optimisation des Interactions avec la Base de Données : Avant vs Après

Pour remédier à cela, j'ai opéré une transition vers un traitement principalement côté navigateur. Au lieu de faire des appels d'API pour chaque modification, j'ai travaillé avec une structure locale, directement sur la page, qui représente le graphe sous forme de JSON. Les opérations telles que les mises

à jour ou les suppressions affectent cette structure locale sans interagir directement avec la base de données. J'ai ainsi limité l'utilisation de la base de données à trois interactions par utilisateur, réduisant considérablement la charge sur nos infrastructures et évitant des appels d'API inutiles

Ce changement a non seulement optimisé l'utilisation des ressources, mais a également amélioré la réactivité et l'expérience utilisateur, puisque les modifications sont appliquées presque instantanément sans le délai d'un appel d'API.

La figure 10 représente le changement au niveau des interactions avec la base de données.

5.4.3. Récapitulatif des réalisations durant le stage

Voici un aperçu des principales missions accomplies chaque semaine :

Semaine 1 :

- Familiarisation avec l'outil BAGUETTE.

Semaine 2-3 :

- Conception de la page d'accueil.

Semaine 3-5 :

- Premières tentatives de conception de l'outil de dessin avec VIS.

Semaine 4-6 :

- Découverte et familiarisation avec Flask.
- Développement de l'API REST.
- Mise en place de la base de données pour le stockage des graphes.

Semaine 7-8 :

- Intégration de l'API, de la base de données et du back-end avec JavaScript et vis.js.
- Manipulation via l'interface UI de Swagger.

Semaine 9 :

- Mise en place des événements pour la création, suppression et stockage des nœuds et des arêtes dans la base de données.

Semaine 10-11 :

- Développement des fonctionnalités d'inscription, de connexion et de déconnexion.

Semaine 12 :

- Amélioration du style de la page de dessin du graphe.
- Optimisation de la gestion des événements, notamment pour la modification des nœuds et des arêtes.

Semaine 13 :

- Intégration du graphe dessiné dans Baguette Core et récupération des résultats.

Semaine 14-15 :

- Transition vers une stratégie axée sur le client pour minimiser les interactions avec la base de données.

Semaine 16 :

- Création des fonctionnalités "Annuler" et "Refaire".
- Peaufinage du style et de l'interface utilisateur.

Le 10 août :

- Déploiement du site avec Ansible.

5.5. Mise en ligne

Lors de la phase de déploiement du site, j'ai eu l'opportunité de participer, bien que dans un rôle secondaire, à la mise en place de la solution avec Ansible. Ansible, étant un outil d'automatisation IT puissant, a été utilisé pour déployer notre site sur une machine virtuelle de manière efficace et standardisée. C'était M. Alexandre Sanchez en coordination avec Vincent Raulin qui a pris la responsabilité principale de cette étape cruciale, assurant que tout était bien configuré. Grâce à cette collaboration et à l'utilisation judicieuse d'Ansible, le site est désormais prêt à être lancé sur internet à tout moment.

6. Conclusion

Au terme de cette période de stage, j'ai acquis une expérience précieuse en matière de développement web et d'intégration d'outils spécifiques, notamment dans le domaine de la visualisation de graphes. J'ai été introduit au domaine passionnant de la cybersécurité. De

plus, j'ai considérablement amélioré mes compétences dans les aspects front-end, back-end, bases de données, et bien d'autres domaines techniques.

La collaboration avec l'équipe, et notamment avec Mme Viet Triem Tong et M. Raulin, a été enrichissante. Les réunions régulières m'ont permis de rester orienté et focalisé sur les objectifs, tout en bénéficiant des conseils et du soutien de mes collègues.

Le défi majeur a été de trouver l'équilibre entre l'efficacité des opérations côté serveur et la nécessité d'offrir une expérience utilisateur fluide et réactive. Grâce à la transition vers un traitement côté navigateur, nous avons pu surmonter ce défi.

Enfin, le déploiement, bien que je n'aie joué qu'un rôle secondaire, m'a offert un aperçu des défis et des solutions associés à la mise en ligne d'une application.

Je tiens à remercier, encore une fois, l'ensemble de l'équipe pour cette opportunité d'apprentissage et de croissance. Je suis convaincu que les compétences et les connaissances acquises au cours de ce stage me seront bénéfiques dans mes futurs projets et initiatives professionnels.

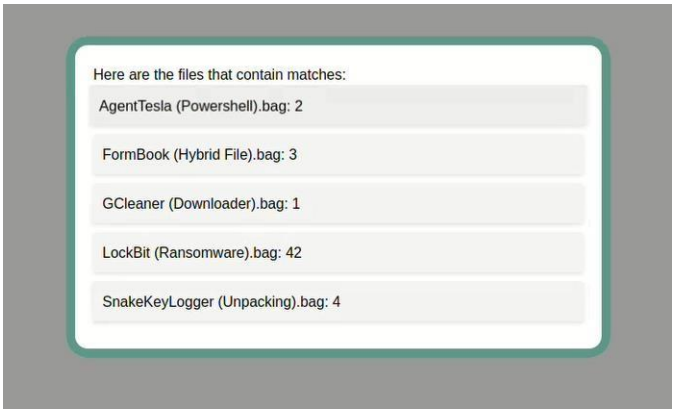
ANNEXES

TABLE DES FIGURES

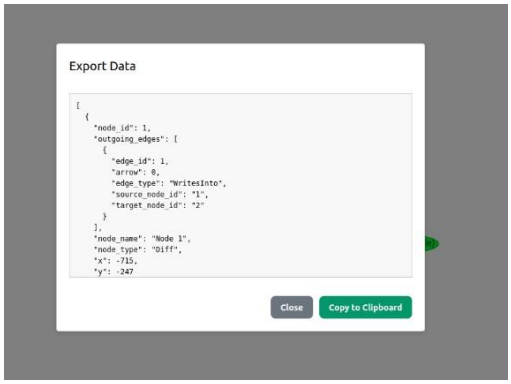
ANNEXES

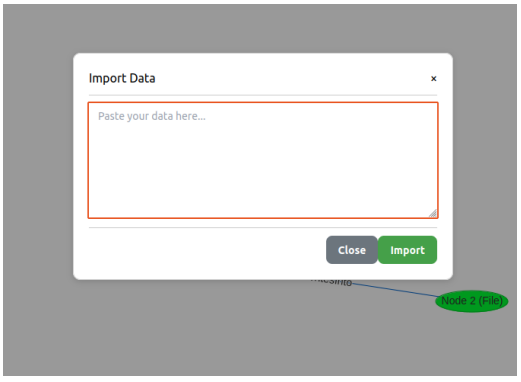
Annexe 1 : Présentation du résultat de la recherche dans Baguette

Lorsqu'on clique sur Search Matches et qu'il y a des correspondances voici comment les résultats s'affichent.

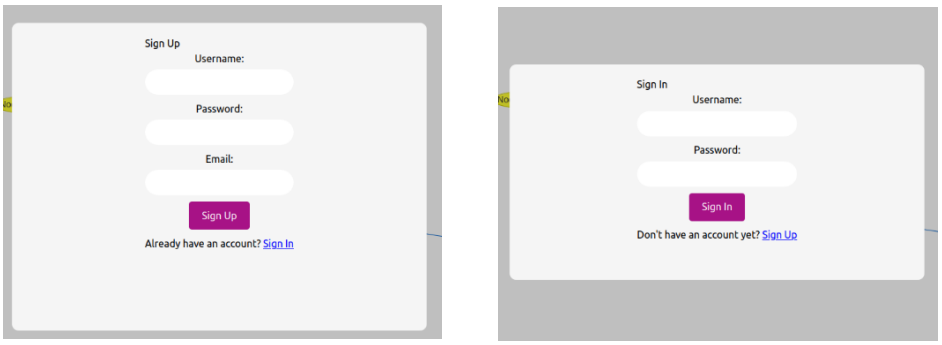


Annexe 2 : Présentation des dialogues Export & Export JSON

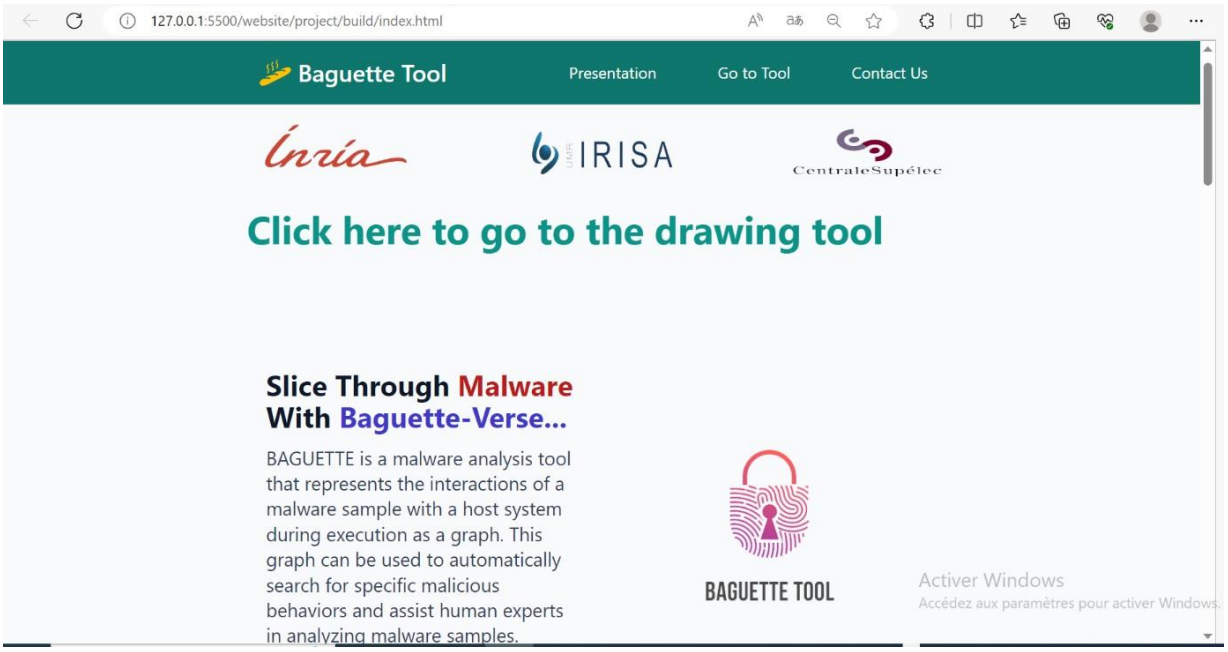


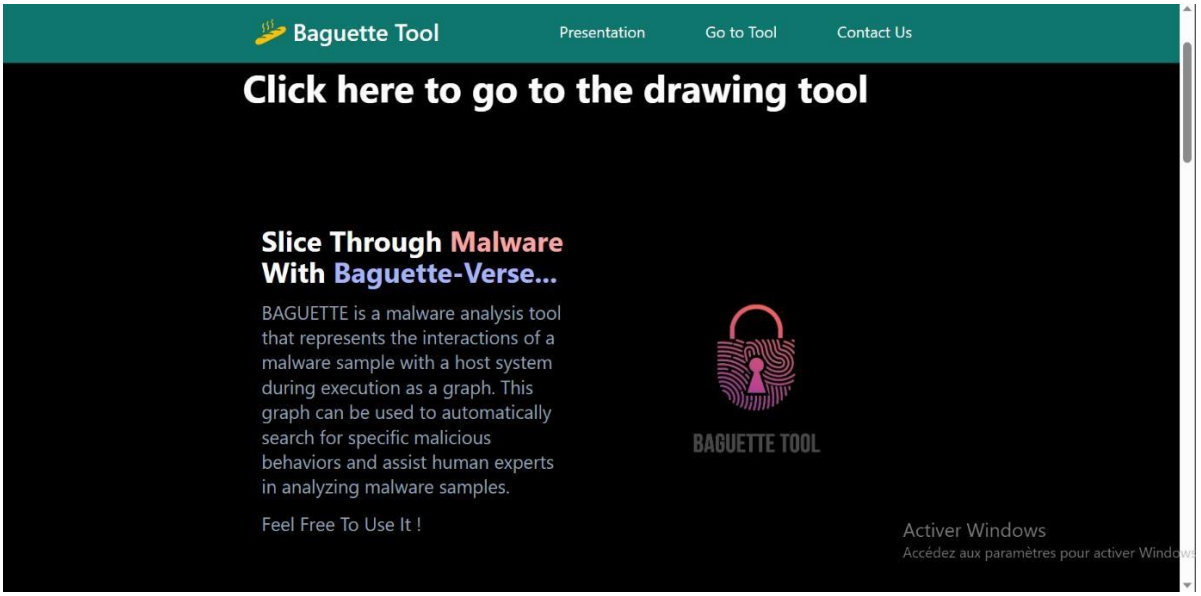


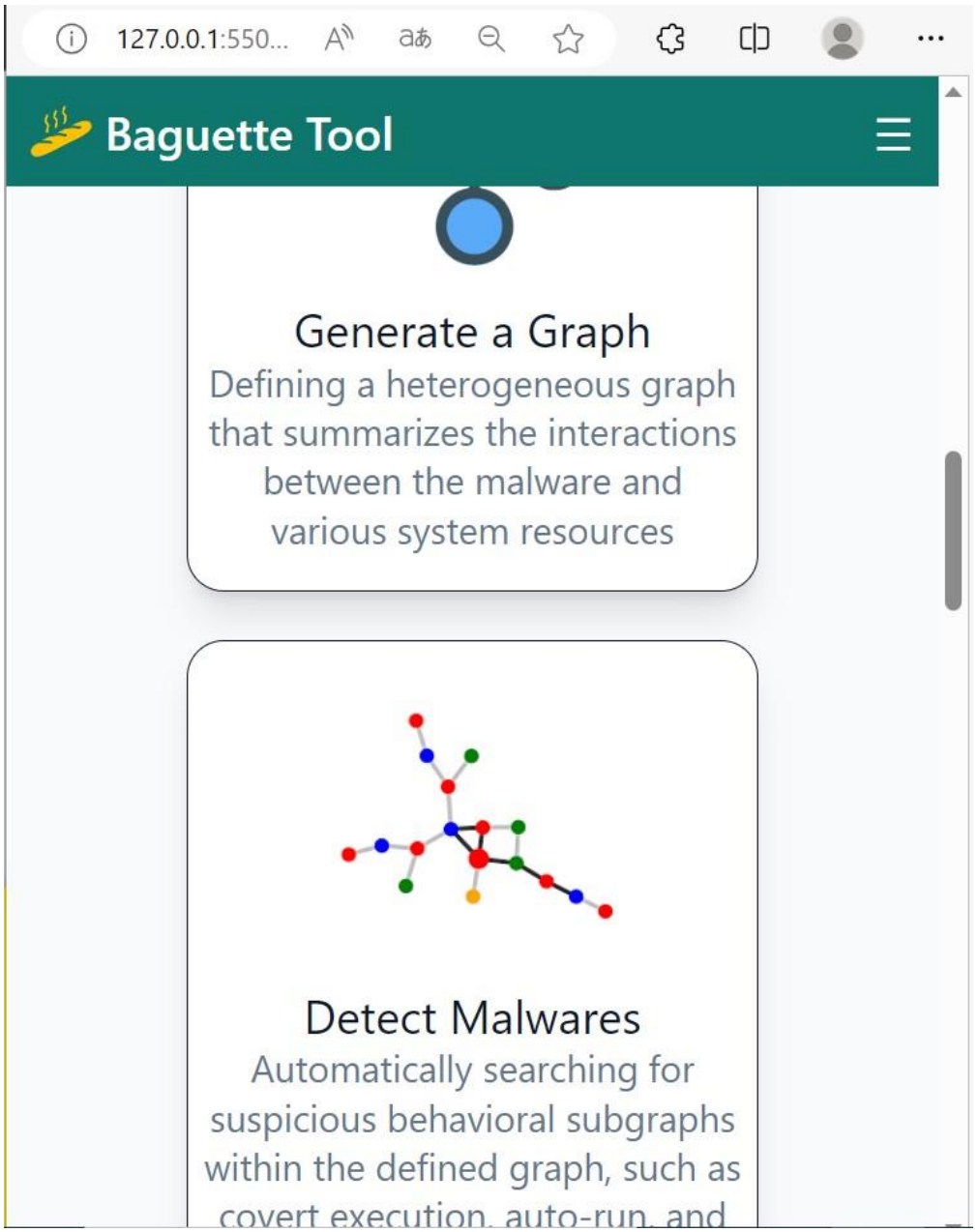
Annexe 3 : Présentation des dialogues Sign In & Sign Out



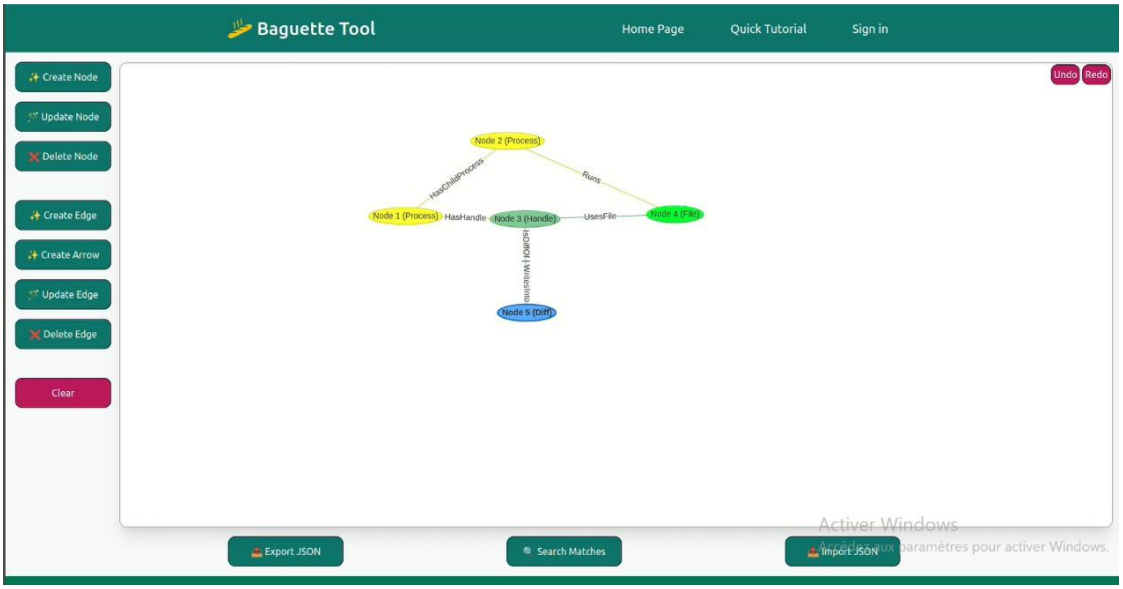
Annexe 4 : Presentation de la page d'acceuil



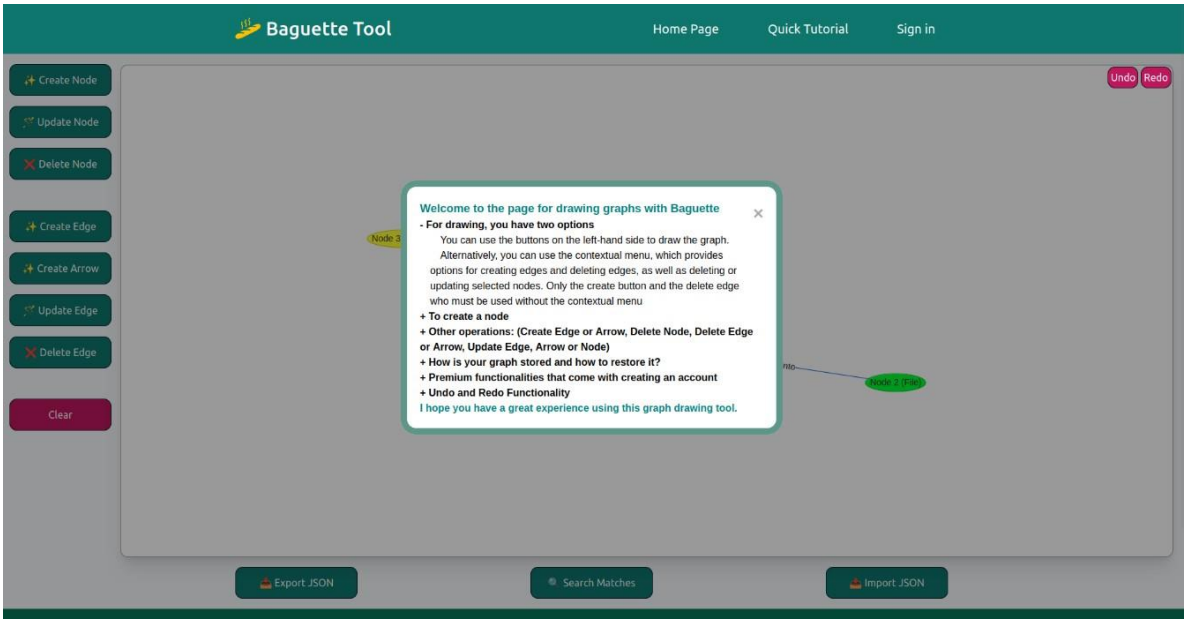



127.0.0.1:550...
Baguette Tool
Generate a Graph
Defining a heterogeneous graph that summarizes the interactions between the malware and various system resources
Detect Malwares
Automatically searching for suspicious behavioral subgraphs within the defined graph, such as

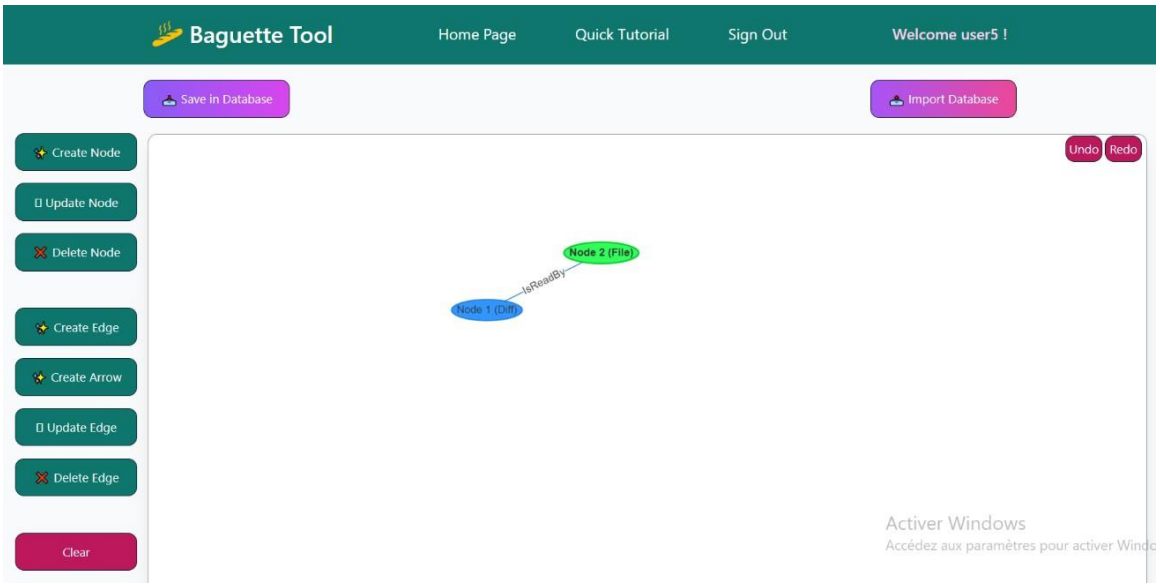
Annexe 5 : Présentation de la page du traçage des graphes



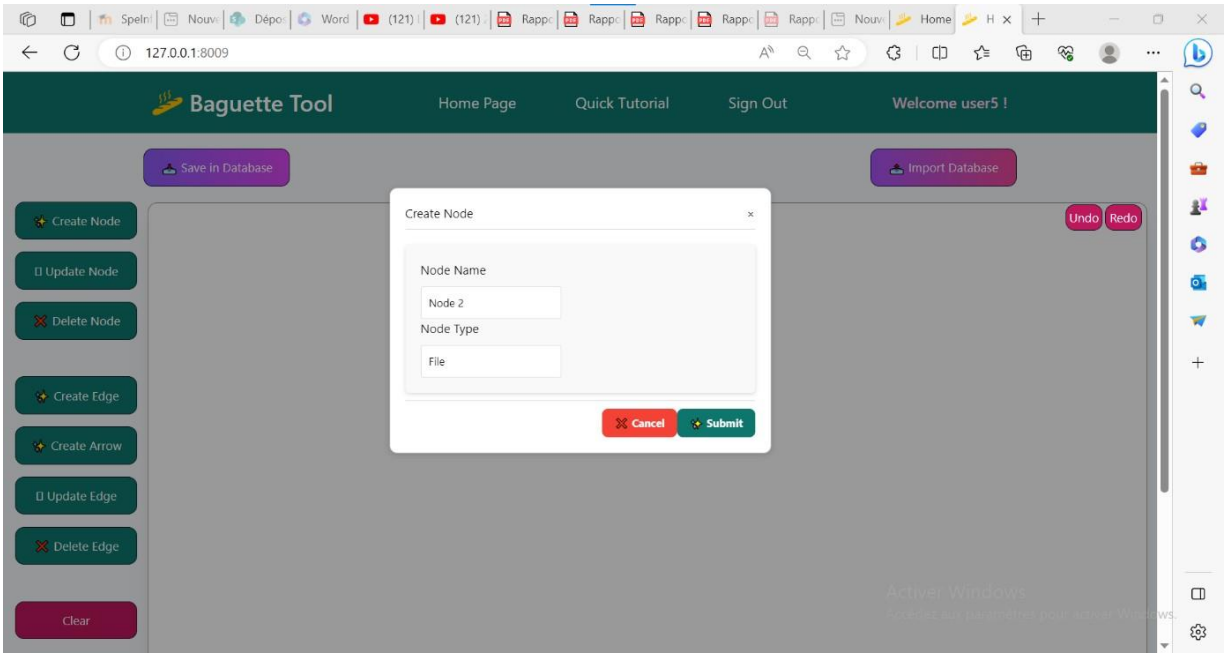
Annexe 6 : Présentation du tutoriel



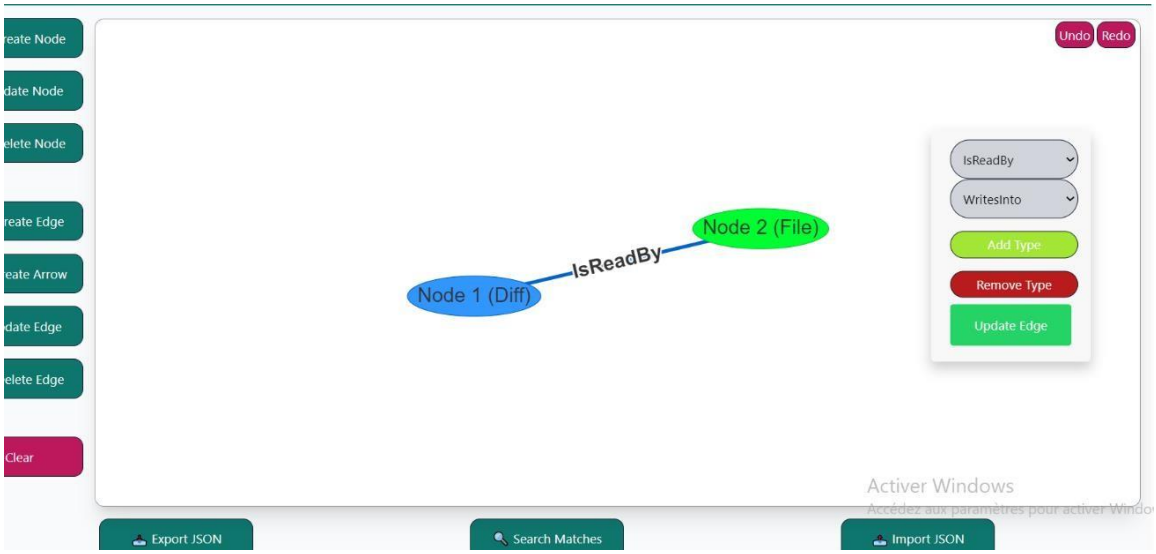
Annexe 7 : Présentation de la fonctionnalité Import & Export in Database (Premium Feature)



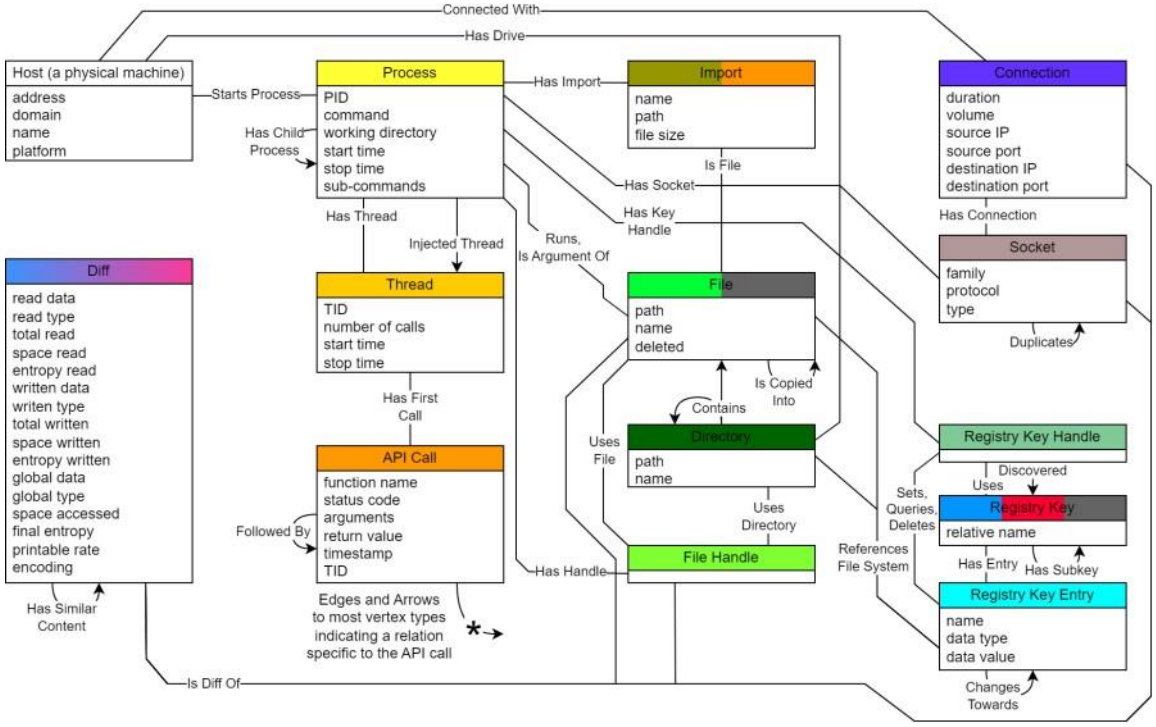
Annexe 8 : Présentation du dialogue create node



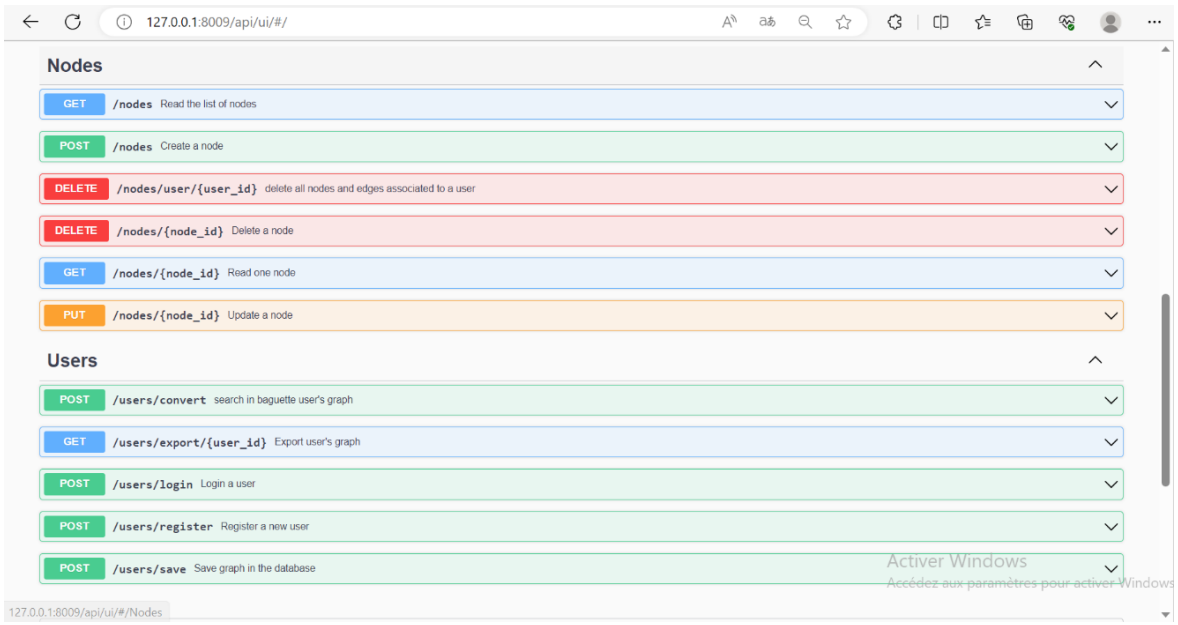
Annexe 9 : Présentation du dialogue Update Edge

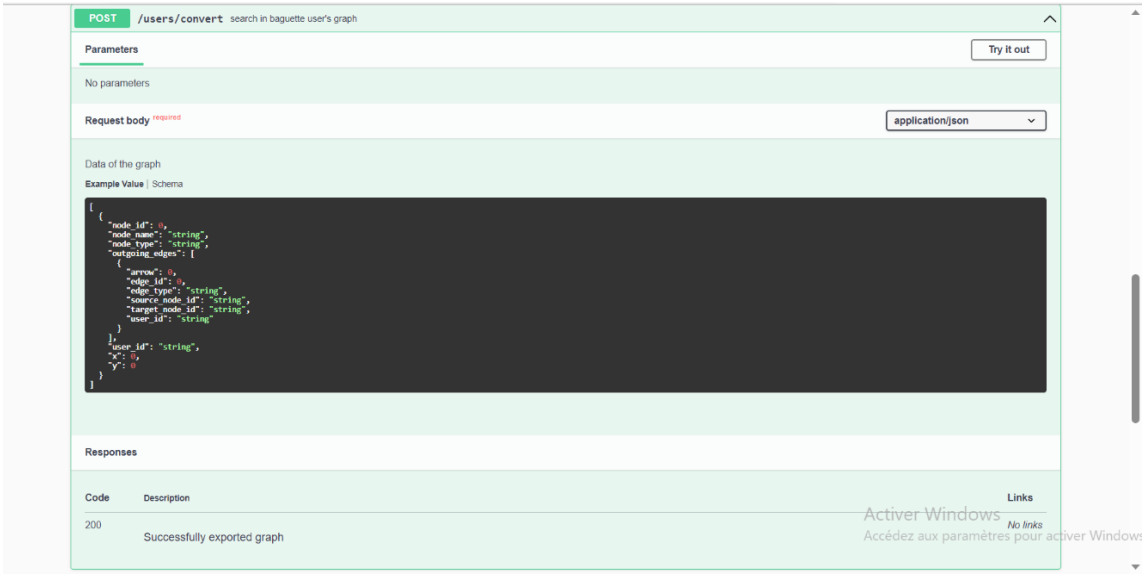


Annexe 10 : Presentation de la structure d'un graphe BAGUETTE et ses composants



Annexe 11 : Présentation de l'Interface Utilisateur (UI) de l'API Documentée avec Swagger


127.0.0.1:8009/api/ui/#/
Nodes
GET /nodes Read the list of nodes
POST /nodes Create a node
DELETE /nodes/user/{user_id} delete all nodes and edges associated to a user
DELETE /nodes/{node_id} Delete a node
GET /nodes/{node_id} Read one node
PUT /nodes/{node_id} Update a node
Users
POST /users/convert search in baguette user's graph
GET /users/export/{user_id} Export user's graph
POST /users/login Login a user
POST /users/register Register a new user
POST /users/save Save graph in the database
Activer Windows
Accédez aux paramètres pour activer Windows
127.0.0.1:8009/api/ui/#/Nodes


POST /users/convert search in baguette user's graph
Parameters
Try it out
No parameters
Request body required
application/json
Data of the graph
Example Value | Schema
Responses
Code
Description
Links
200
Successfully exported graph
No links
Activer Windows
Accédez aux paramètres pour activer Windows

REFERENCES

[1] https://www.sqlite.org/whentouse.html
[2] Grinberg, M. (2018). Flask Web Development: Developing Web Applications with Python (2nd ed.). O'Reilly Media.
[3] https://www.inria.fr/en/research-and-innovation
[4] https://www.inria.fr/fr/securite-numerique
[5] https://hal.science/hal-04102144/site contenant l'article de recherche à propos de Baguette
[6] https://realpython.com/flask-connexion-rest-api-part-2
[7] https://visjs.github.io/vis-network/docs/network/

Résumé

Au cours de mon stage cette année, j'ai non seulement enrichi mes compétences, mais j'ai également eu le privilège de collaborer avec des experts dévoués et compétents en informatique, surtout dans le domaine de la cybersécurité. Ma principale mission était de développer un site permettant de dessiner aisément des graphes représentant des comportements malveillants. Ces graphes sont ensuite traités par l'outil "Baguette" pour afficher les malwares dans la base de données des malwares ayant ce comportement. En d'autres termes, l'outil détecte les malwares dont les graphes générés , par l'outil même, contiennent le graphe qu'on a dessiné. L'outil "Baguette", un projet innovant dirigé par Vincent Raulin, m'a particulièrement impressionné, stimulant ainsi ma motivation à donner le meilleur de moi-même. Durant cette expérience, j'ai élargi mes compétences en développement backend, frontend et gestion de bases de données. De plus, ma perception et ma compréhension de la cybersécurité se sont considérablement approfondies.

Mots Clés : Cybersécurité, BAGUETTE, Graphe, Développement, Malware

Summury

During my internship this year, I not only honed my skills but also had the privilege of collaborating with dedicated and knowledgeable IT experts, especially in the field of cybersecurity. My primary task was to develop a website that allows for the easy drawing of graphs representing malicious behaviors. These graphs are then processed by the "Baguette" tool to display malwares in the malware database containing this behavior. In other words, the tool identifies malwares whose generated graphs, by the same BAGUETTE Software, contain the graph we had drawn. The "Baguette" tool, an innovative project led by Vincent Raulin, particularly impressed me, thereby boosting my motivation to give my best. Throughout this experience, I expanded my skills in backend development, frontend development, and database management. Moreover, my perception and understanding of cybersecurity have deepened considerably.

Keywords : CyberSecurity, BAGUETTE, Graph, Development, Malware

Ecole Publique d'Ingénieurs en 3 ans

6 boulevard Maréchal Juin, CS 45053
14050 CAEN cedex 04