

# RAPPORT DE PROJET

## EPREUVE E6.2

SESSION  
2022-2023

Matthieu LAURENT  
Adrien BRUAS  
Stefen INCE

BTS SNIR

## 1 Table des matières

|       |   |    |
|-------|---|----|
| 1     | Table des matières .....                      | 2  |
| 2     | Glossaire .....                               | 4  |
| 3     | Présentation du projet .....                  | 5  |
| 3.1   | Contexte (présentation de l'entreprise) ..... | 5  |
| 3.2   | Formation de l'équipe .....                   | 6  |
| 3.3   | Répartition des tâches .....                  | 6  |
| 3.4   | Remerciement .....                            | 7  |
| 3.5   | Besoin du client .....                        | 7  |
| 3.6   | Formalisation du besoin .....                 | 7  |
| 3.7   | Définition du cahier des charges .....        | 8  |
| 3.8   | Exigences fonctionnelles et techniques .....  | 8  |
| 4     | Conception préliminaire .....                 | 10 |
| 4.1   | Analyse de l'existant .....                   | 10 |
| 4.2   | Présentation de la solution .....             | 11 |
| 4.3   | Cas d'utilisation .....                       | 13 |
| 4.4   | Présentation de la base de données .....      | 14 |
| 4.5   | Présentation des outils .....                 | 15 |
| 4.6   | Matériels utilisés .....                      | 16 |
| 4.7   | Planning prévisionnel .....                   | 17 |
| 4.8   | DevOps .....                                  | 18 |
| 4.8.1 | Git .....                                     | 18 |
| 4.8.2 | Azure DevOps .....                            | 19 |
| 4.9   | Cahier de recettes .....                      | 22 |
| 5     | Réalisation - Matthieu LAURENT .....          | 23 |
| 5.1   | Choix des outils .....                        | 23 |
| 5.2   | Qualité du code .....                         | 25 |
| 5.2.1 | Documentation .....                           | 25 |
| 5.2.2 | Tests automatiques du code .....              | 26 |
| 5.2.3 | Convention de nommage .....                   | 28 |
| 5.2.4 | Formatage du code .....                       | 29 |
| 5.2.5 | Lintage du code .....                         | 29 |
| 5.2.6 | Etapes pre-commit .....                       | 30 |
| 5.2.7 | Structure du projet .....                     | 30 |
| 5.2.8 | Environnement virtuel .....                   | 31 |
| 5.3   | Outils de template Jinja2 .....               | 31 |
| 5.4   | Structure Flask .....                         | 33 |

|       |   |     |
|-------|---|-----|
| 5.5   | Structure de l'API .....                                  | 35  |
| 5.6   | Formulaires avec WTForms .....                            | 39  |
| 5.7   | Base de données .....                                     | 39  |
| 5.8   | Interaction avec l'hyperviseur .....                      | 43  |
| 5.9   | Accès distant aux machines virtuelles via VNC .....       | 45  |
| 5.10  | Éditions des questions depuis le site .....               | 45  |
| 5.11  | Créations et affichage des VMs sur le site .....          | 46  |
| 5.12  | Profil utilisateur .....                                  | 47  |
| 5.13  | Site web « responsive » .....                             | 48  |
| 5.14  | Interface en ligne de commande Flask .....                | 49  |
| 5.15  | Langage Markdown .....                                    | 50  |
| 5.16  | Conclusion .....  | 52  |
| 6     | Réalisation - Adrien BRUAS .....                          | 53  |
| 6.1   | Procédure de déploiement .....                            | 53  |
| 6.2   | Modulation du site .....                                  | 62  |
| 6.3   | Supervision .....   | 64  |
| 6.4   | Ergonomie du site .....                                   | 65  |
| 6.5   | Structure du site .....                                   | 66  |
| 6.6   | Conclusion .....  | 66  |
| 7     | Réalisation - Stefen INCE .....                           | 68  |
| 7.1.1 | Présentation .....  | 68  |
| 7.1.2 | Analyse préliminaire .....                                | 68  |
| 7.1.3 | Front-end .....   | 69  |
| 7.1.4 | Classement .....  | 74  |
| 7.1.5 | Création VM Attaque .....                                 | 84  |
| 7.1.6 | Création des rooms .....                                  | 89  |
| 7.1.7 | Conclusion .....  | 100 |
| 8     | Conclusion .....  | 101 |
| 9     | Annexes .....   | 102 |
| 9.1   | Extrait de rapport de test HTML .....                     | 102 |
| 9.2   | Classe abstraite VMManager .....                          | 103 |
| 9.3   | Fonction get_n_around .....                               | 104 |
| 9.4   | Tests unitaires pour la fonction get_n_around .....       | 105 |
| 9.5   | Utilisation d'un formulaire WTForm – Template Jinja ..... | 106 |
| 9.6   | Utilisation d'un formulaire WTForm – Code Flask .....     | 107 |
| 9.7   | Ressource Flask-restx pour les questions .....            | 108 |
| 9.8   | Classe modèle SQLAlchemy .....                            | 109 |

|      |   |     |
|------|---|-----|
| 9.9  | Code complet du filtre markdown (fonction markdown_filter).....   | 110 |
| 9.10 | Code HTML complet de la page d'accueil .....                      | 111 |
| 9.11 | Code HTML du header partie utilisateur connecté.....              | 113 |
| 9.12 | Code HTML du header partie l'utilisateur n'est pas connecté ..... | 114 |
| 9.13 | Code HTML du header partie l'utilisateur n'est pas connecté ..... | 115 |
| 9.14 | Liens utilisés dans le footer dans app_config.py.....             | 116 |
| 9.15 | Code HTML tableau classement .....                                | 116 |
| 9.16 | Code HTML création de la route /profile .....                     | 117 |
| 9.17 | Procédure pour convertir une machine virtuelle en template .....  | 118 |
| 9.18 | Procédure de création d'un réseau virtuelle .....                 | 120 |
| 9.19 | Procédure d'installation des requirement.....                     | 121 |
| 9.20 | Procédure installation Python.....                                | 123 |
| 9.21 | Procédure de déploiement du serveur ProxMox.....                  | 125 |
| 9.22 | Procédure de mise en place de dnsmasq (ProxMox) .....             | 125 |
| 9.23 | Template base Jinja .....   | 126 |

## 2 Glossaire

|   |   |
|---|---|
| CTF ( <i>Capture The Flag</i> / Capture de drapeau) | Un type de compétition de sécurité informatique où les participants doivent pirater une machine pour y accéder et y trouver des "flags" (drapeaux) qui sont généralement des chaînes de caractères spécifiques.   |
| VM ( <i>Virtual Machine</i> / Machine virtuelle)    | Un environnement logiciel qui émule un ordinateur physique  |
| Machine d'attaque                                   | Une machine virtuelle accessible à distance par l'utilisateur, contenant tous les outils nécessaires au piratage des machines victimes.   |
| Machine victime                                     | Une machine virtuelle contenant volontairement des failles de sécurité, et avec des flags cachés.   |
| Room (salle)  | Un défi spécifique contenant une série de questions, ainsi qu'optionnellement des machines victimes associées   |
| Hyperviseur   | Logiciel permettant de créer et de gérer des machines virtuelles. Il existe deux types d'hyperviseurs : <ul style="list-style-type: none"> <li>• Type 1 : L'hyperviseur s'exécute directement sur le matériel physique. Plus efficace en termes de performances.</li> <li>• Type 2 : L'hyperviseur est un logiciel s'exécutant sur un autre système d'exploitation (« hôte »).</li> </ul> |
| Template (VM)                                       | Un template de machine virtuelle est un modèle préconfiguré et préinstallé d'un système d'exploitation et de logiciels spécifiques, utilisé pour créer rapidement de nouvelles instances de machines virtuelles.  |
| Décorateur python                                   | Un décorateur Python est une fonction spéciale qui permet de modifier le comportement d'une autre fonction en y ajoutant des fonctionnalités supplémentaires sans la modifier directement.<br>Il s'ajoute en plaçant un @ suivit du nom du décorateur au-dessus de la déclaration de la fonction décorée.   |

### 3 Présentation du projet

#### 3.1 Contexte (présentation de l'entreprise)

Le projet majeur qui nous a été confié dans le cadre de notre BTS SNIR est une étape cruciale pour nous en tant qu'étudiants. Nous sommes appelés à mettre en pratique toutes les compétences acquises au cours de nos deux années de formation et à les appliquer à la réalisation d'un projet ambitieux et innovant. Le processus de réalisation de ce projet est très rigoureux et exigeant, car il comprend toutes les étapes de la modélisation à l'installation, en passant par la mise en place d'une infrastructure solide.

Au cours de ce projet, nous sommes amenés à travailler en équipe et à collaborer étroitement pour atteindre les objectifs fixés. Cela nous donne l'occasion de mettre en pratique nos compétences en communication, en leadership et en gestion de projet, tout en nous familiarisant avec les outils et les technologies modernes utilisées dans l'industrie.

En fin de compte, la réussite de ce projet sera un véritable catalyseur pour notre carrière future. Nous serons en mesure de présenter un projet concret et innovant à nos futurs employeurs, démontrant ainsi notre capacité à travailler en équipe et à mener à bien des projets complexes. En somme, ce projet nous permet de nous mettre en valeur et de nous préparer pour notre vie professionnelle à venir.

C'est dans ce contexte que l'association RootMe a développé des CTF (Capture The Flag) qui permettent de s'initier à la cybersécurité de manière ludique et interactive. Les CTF proposés par RootMe sont accessibles à tous, quels que soient votre niveau de connaissance en informatique et votre expérience.

La plateforme propose des "rooms" qui sont en fait des énigmes ou des défis à résoudre. Chaque room est conçue pour tester une compétence spécifique en matière de cybersécurité, comme le hacking de systèmes d'exploitation, le décryptage de messages ou encore la recherche de failles de sécurité dans un code.

Bien que la plateforme RootMe ne propose pas d'environnement permettant de réaliser les attaques, elle fournit toutes les instructions nécessaires pour réaliser les défis. Les participants doivent donc utiliser leur propre machine pour attaquer des machines virtuelles sur un réseau et résoudre les énigmes.

Ces CTF sont une excellente façon d'apprendre la cybersécurité de manière pratique et ludique. En effet, ils permettent aux participants de mettre en pratique les connaissances acquises et d'expérimenter différentes techniques de hacking en toute sécurité. En résolvant les défis, les participants développent leur sens de la logique et leur capacité à résoudre des problèmes complexes, ce qui est essentiel pour devenir un professionnel de la cybersécurité.

### 3.2 Formation de l'équipe

Dans le cadre de ce projet notre groupe est constitué des trois étudiant.

- Matthieu LAURENT (développeur Backend ; chef de projet ; scrum master)
- Stefen INCE (développeur Frontend ; créateur de room)
- Adrien BRUAS (développeur Frontend ; documentaliste)

### 3.3 Répartition des tâches

Ce projet utilise une méthodologie agile, donc tous les membres touchent à tous.

|                          | Matthieu L. | Adrien B. | Stefen I. |
|--------------------------|-------------|-----------|-----------|
| Backend                  | R           | I         | I         |
| Frontend                 | A/C         | R         | R         |
| Intégration Proxmox      | R           | I         | I         |
| Documentation code       | R           | I         | I         |
| Diagrammes               | R           | R         | R         |
| Design                   | C           | R         | A         |
| Création rooms           | R           | R         | R         |
| Contrôle qualité du code | R           | I         | I         |

| Responsable (R) | Approuveur (A) | Consulté (C) | Informé (I) |
|-----------------|----------------|--------------|-------------|
|-----------------|----------------|--------------|-------------|

### 3.4 Remerciement

Nous tenons à exprimer nos sincères remerciements à tous nos professeurs qui nous ont accompagné au cours de ces deux années. En particulier, nous souhaitons adresser nos plus vifs remerciements à notre professeur référent, Jean-François Marquette. Ses conseils précieux et son expertise ont été indispensables à notre réussite.

### 3.5 Besoin du client

Au commencement de notre projet, nous avons été contactés par un client qui nous a demandé de créer quatre salles sur le site RootMe. Cette demande impliquait la mise en place d'une plateforme capable de déployer une solution CTF, avec des machines d'attaque (Kali Linux) associées. Pour ce faire, nous avons dû travailler à la reproduction du fonctionnement d'une telle plateforme, en veillant à ce qu'elle soit opérationnelle sur un environnement ProxMox. De plus, nous avons dû prévoir l'automatisation des déploiements des défis et des machines d'attaque, pour garantir une expérience utilisateur fluide et sans encombre.

### 3.6 Formalisation du besoin

Pour atteindre ces objectifs, nous avons dû relever plusieurs défis. Tout d'abord, nous avons travaillé sur la mise en place d'une attaque MITM via ARP poisoning, pour laquelle nous avons dû déployer deux machines virtuelles et une machine virtuelle Kali Linux. Ensuite, nous avons travaillé sur une attaque MITM via ICMP, pour laquelle nous avons à nouveau dû déployer deux machines virtuelles et une machine virtuelle Kali Linux. Pour réaliser ces attaques, nous avons eu recours à des techniques de spoofing pour tromper les machines cibles, en faisant passer notre machine pour le routeur.

Ensuite, nous avons travaillé sur le rerouting de RIPv2, qui consistait à casser le hash et à injecter une nouvelle route. Pour cela, nous avons eu recours à GNS3 ou à une machine virtuelle simple avec Quagga. Enfin, nous avons travaillé sur l'hijacking OSPF, pour lequel nous avons également utilisé GNS3 ou une machine virtuelle simple avec Quagga.

Cependant, nous n'avons pas été en mesure de contacter notre client pour obtenir des précisions sur ses attentes. Face à cette situation, nous avons pris la décision de revoir notre cahier des charges avec l'aval de notre professeur référent, afin de rendre le projet réalisable. Notre nouvelle mission consiste donc à créer un environnement CTF, dans lequel les machines d'attaque et les victimes seront entièrement virtualisées. Cette approche nous permettra de travailler sur un environnement contrôlé, en veillant à ce que les attaques soient réalisées dans un contexte sécurisé et sous contrôle. Nous sommes convaincus que cette nouvelle approche nous permettra de réaliser un projet de qualité, qui répondra aux attentes de notre client potentiel et qui pourra également servir à d'autres fins pédagogiques et de recherche dans le domaine de la cybersécurité. Grâce à cela, notre solution va permettre à la nouvelle filière BTS CIEL de présenter une solution pour permettre aux étudiants de travailler en toute simplicité.

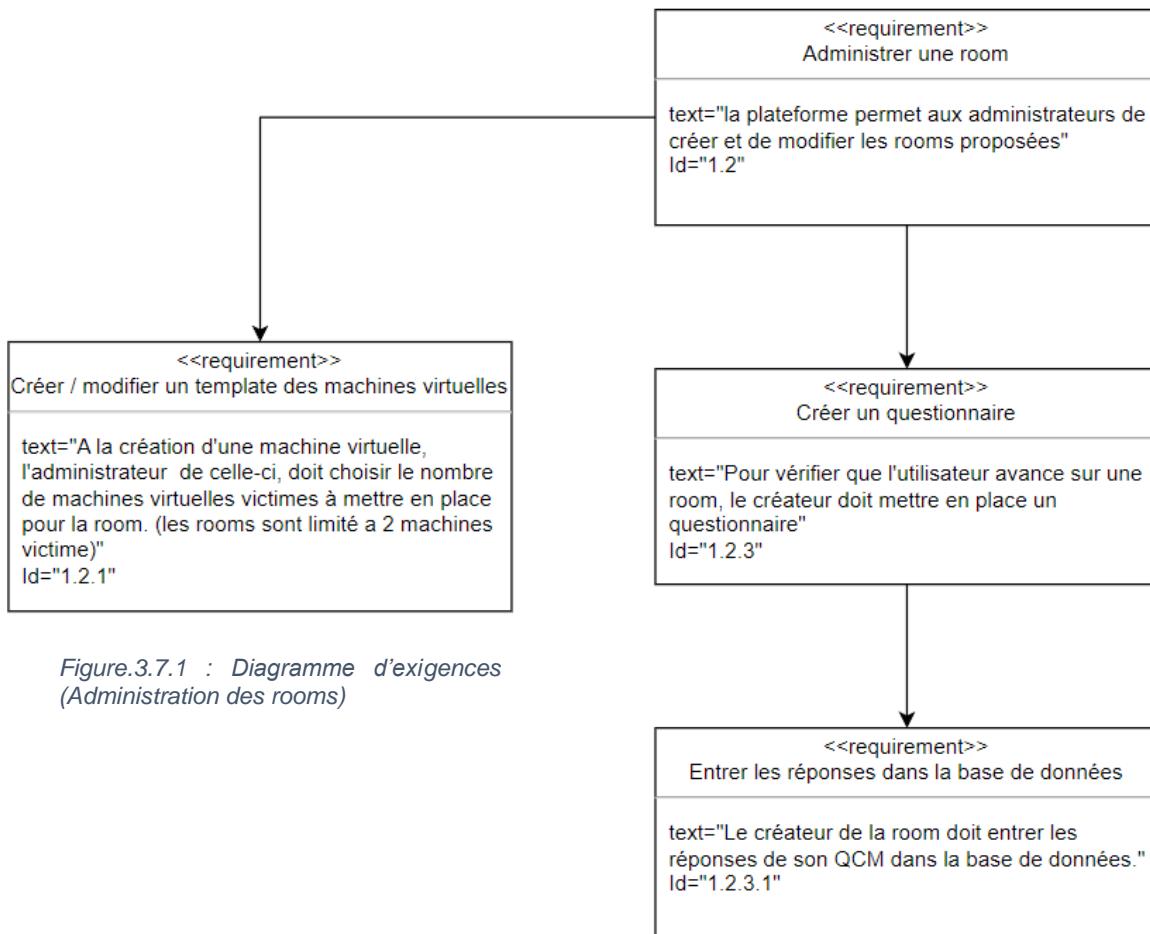
### 3.7 Définition du cahier des charges

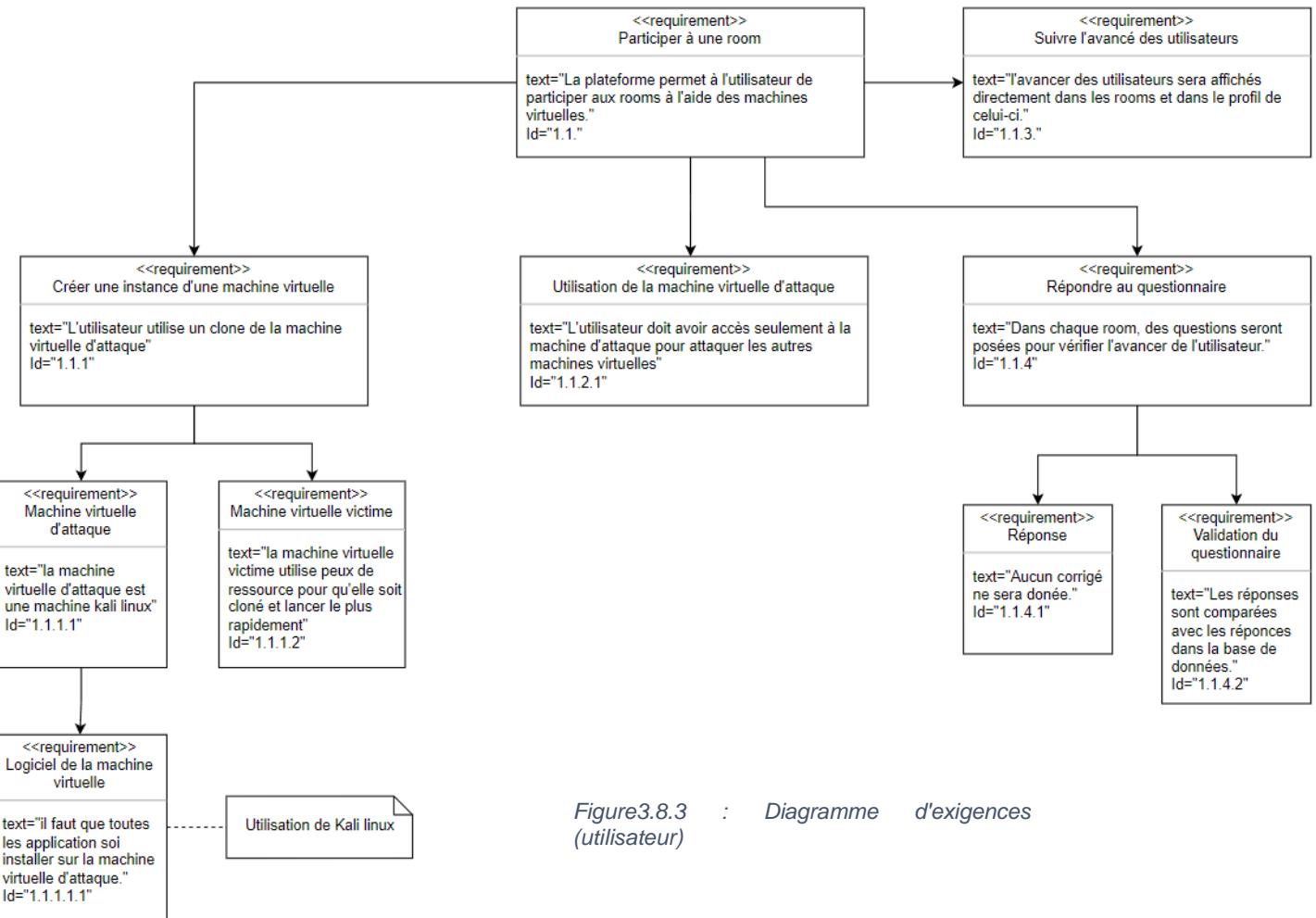
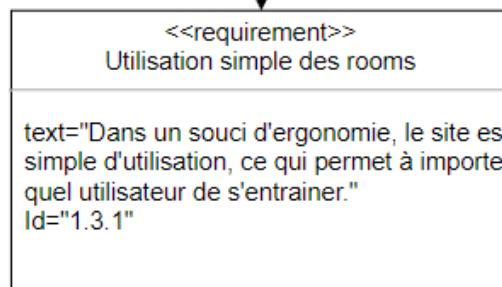
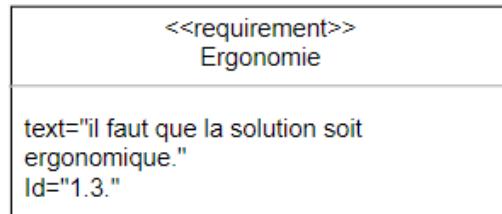
Le cahier des charges spécifie que la solution doit fonctionner sur un environnement ProxMox et qu'elle doit inclure une automatisation des défis et des machines d'attaque. Le cahier des charges exige également la production d'un suivi d'activité des utilisateurs.

En outre, le document indique que les utilisateurs doivent se connecter pour avoir accès à la room. Cela implique que la solution devra inclure un système de connexion sécurisé pour les utilisateurs.

En résumé, le cahier des charges pour ce projet spécifie les exigences techniques et fonctionnelles pour la reproduction d'une plateforme CTF, y compris les spécifications pour l'environnement de déploiement, l'automatisation des défis et des machines d'attaque, le suivi d'activité des utilisateurs et le système de connexion des utilisateurs.

### 3.8 Exigences fonctionnelles et techniques





## 4 Conception préliminaire

### 4.1 Analyse de l'existant

|  | TryHackMe                      | Hack the box                   | Newbie Contest | W3Challs | Root Me                        | <u>Solution proposée</u> |
|--|--------------------------------|--------------------------------|----------------|----------|--------------------------------|--------------------------|
| <b>Disponible en français ?</b>                | Non                            | Non                            | Oui            | Non      | Oui                            | Oui                      |
| <b>Gratuit ?</b>                               | Partiellement<br>(Version pro) | Partiellement<br>(Version pro) | Oui            | Oui      | Partiellement<br>(Version pro) | Oui                      |
| <b>Machines virtuelles victimes intégré ?</b>  | Oui                            | Oui                            | Non            | Oui      | Oui                            | Oui                      |
| <b>Machines virtuelles d'attaque intégré ?</b> | Oui                            | Oui (payant)                   | Non            | Non      | Non                            | Oui                      |

Figure 4.1.1 : Comparaisons des différents sites de CTF

## 4.2 Présentation de la solution

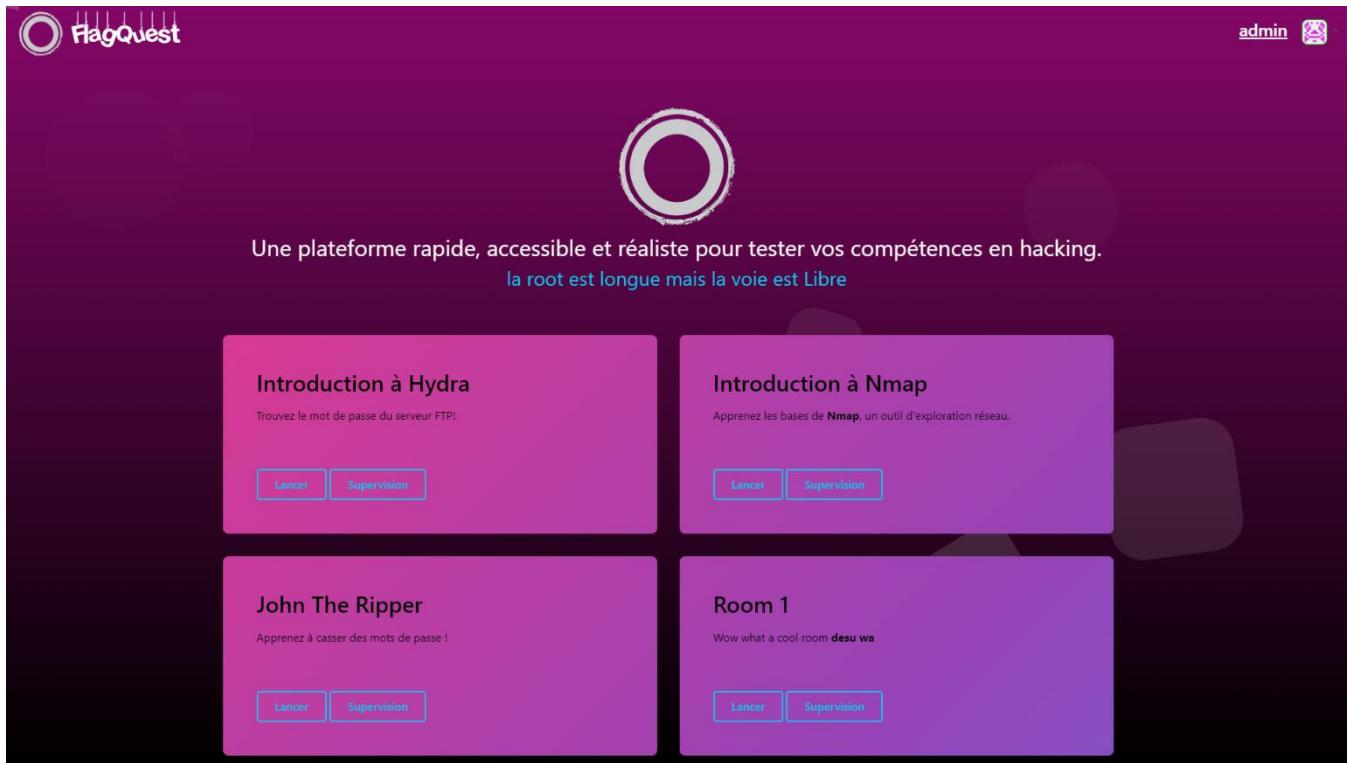


Figure.4.2.1 : Page d'accueil du site

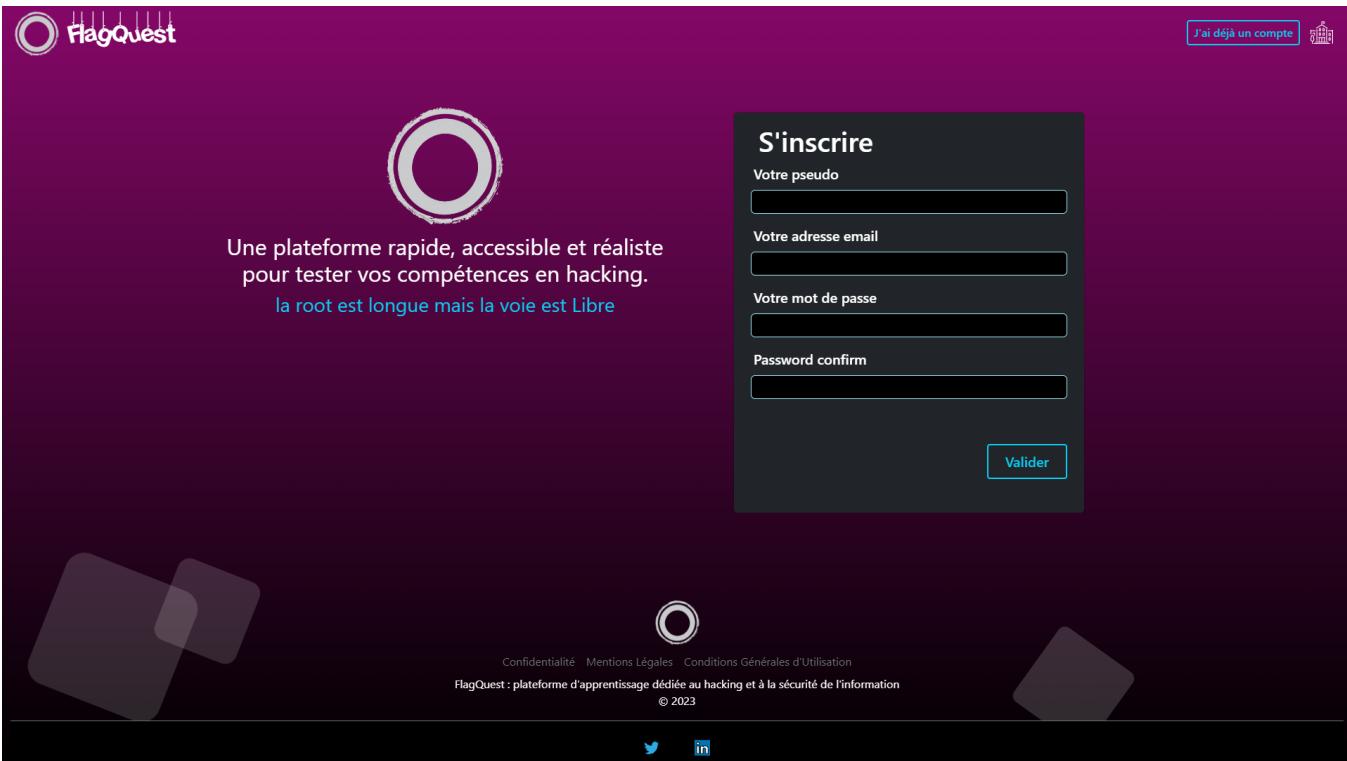


Figure.4.2.2 : Page de création de compte

The screenshot shows a user profile page with the following details:

- User Profile:** Edouard Branty (Avatar: blue and white pixelated circle), Matthieu (Avatar: purple and white pixelated square).
- Ranking:** #1 Position, 31 Points, 3 Challenges.
- Rooms Participated In:**
  - John The Ripper:** Apprenez à casser des mots de passe !
  - Introduction à Nmap:** Apprenez les bases de Nmap, un outil d'exploration réseau.
  - Introduction à Hydra:** Trouvez le mot de passe du serveur FTP!
- Classement (Ranking):**

| Position | Avatar   | Utilisateur | Score |
|----------|----------|-------------|-------|
| #1       | Matthieu | Matthieu    | 31    |
| #2       | miku     | miku        | 21    |
| #3       | john_doe | john_doe    | 2     |
- Progression (Progress):** A line graph showing points over time from 03/05/2023 to 09/05/2023. The points increase from 0 to approximately 30.

Figure 4.2.3: Page du profil utilisateur

The screenshot shows the Hydra room page with the following content:

- Consignes (Instructions):**
  - Quel était le mot de passe du serveur FTP?
  - Avec quel outils graphique (très populaire) peut-on se connecter au serveur FTP?
  - Donnez le flag caché dans le fichier `flag.txt` sur le serveur
- Hydra**: Un logiciel libre permettant de craquer un mot de passe en ligne par **bruteforce**.
- Installation**: Cet outil est déjà installé sur la VM d'attaque, cette explication est simplement à but éducatif.
- Sous Linux**: On peut l'installer via `apt-get` ou `snap`.
 

```
$ sudo apt-get update
$ sudo apt-get install hydra -y
```
- Sous Windows**: Passez par [Cygwin](#) qui est une bibliothèque de logiciels libres permettant d'émuler un système Linux sous différentes versions de Windows.
- Utilisation**
- FTP**: Pour brute forcer un serveur FTP, on utilise la commande suivante :
 

```
$ hydra -l <username> -P </path/to/wordlist> ftp://<ftp-ip-adress> -V
```

  - l - spécifier le nom d'utilisateur du FTP
  - P - indiquer le chemin de la wordlist à utiliser
  - V - affiche le nom d'utilisateur et le mot de passe testés à chaque essai
- SSH**: Pour SSH, c'est légèrement différent :
 

```
hydra -l <username> -P </path/to/wordlist> <ssh-ip-adress> -t 4 ssh
```

  - l - spécifier le nom d'utilisateur
  - P - indiquer le chemin de la wordlist à utiliser
  - t - nombre de connexions en parallèle (16 par défaut)

Figure 4.2.4 : Room Hydra

### 4.3 Cas d'utilisation

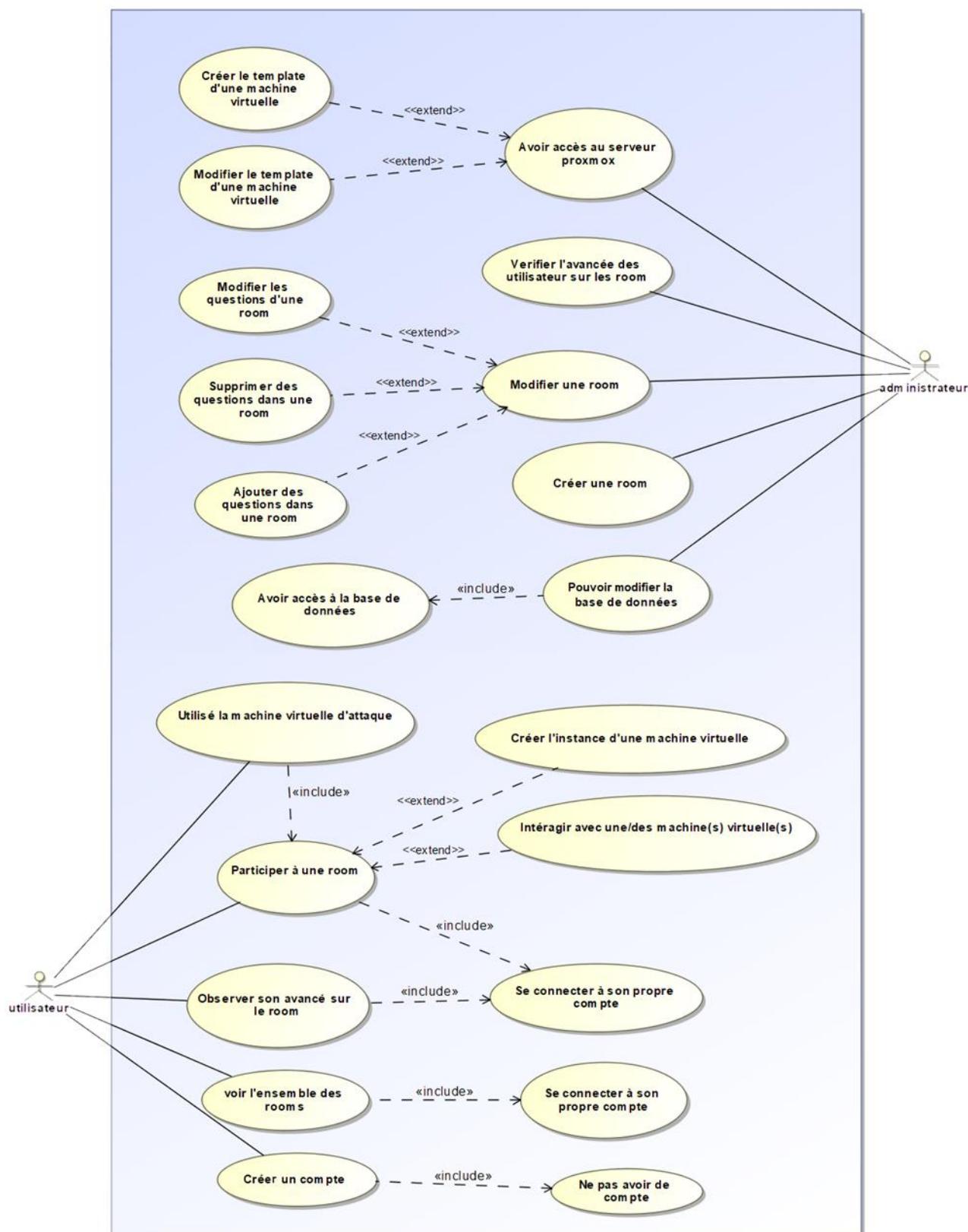


Figure 4.3.1 : Définition des cas d'utilisations

#### 4.4 Présentation de la base de données

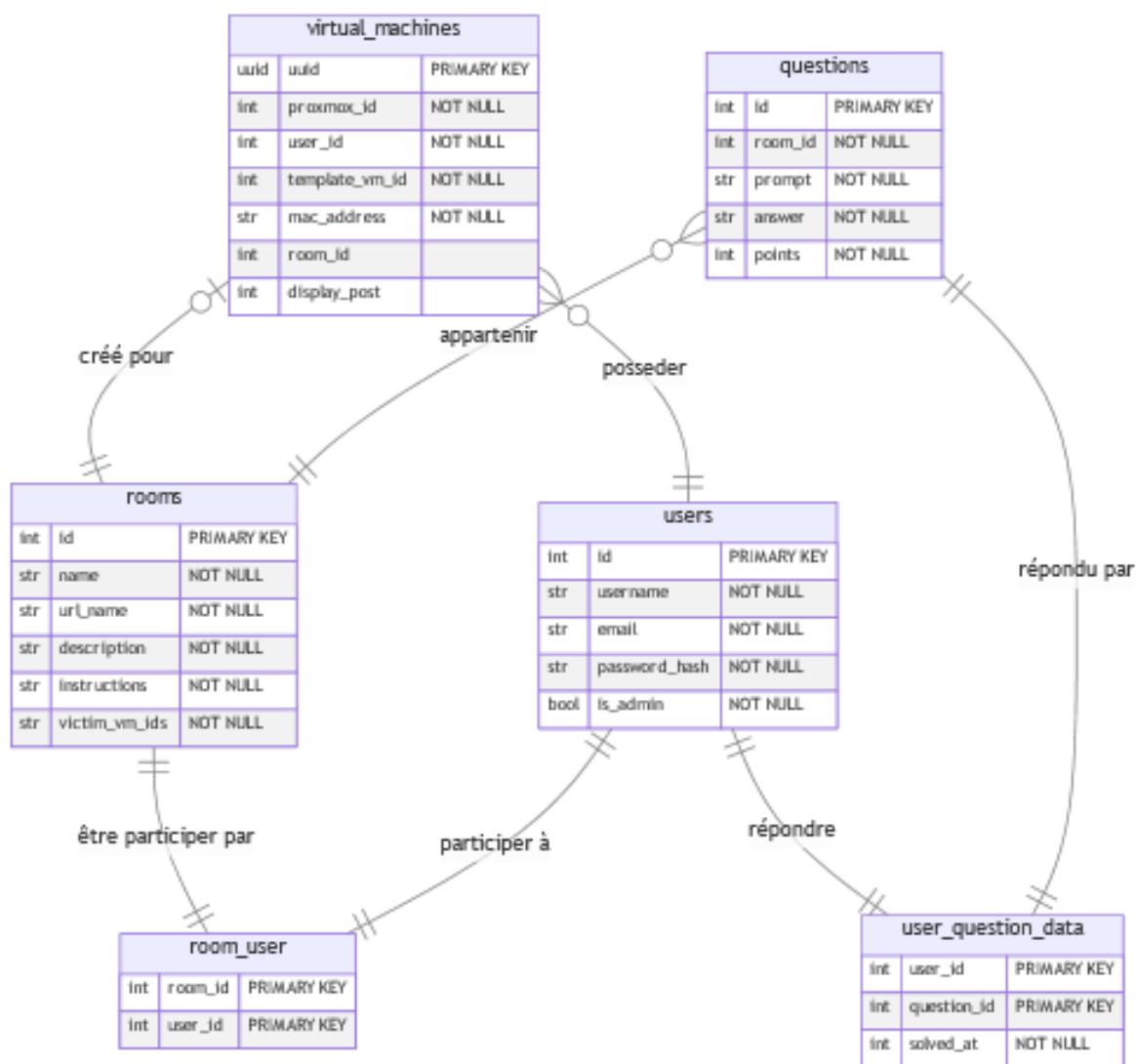


Figure 4.4.1: Diagramme Entité-Relation

## 4.5 Présentation des outils

| Programmation | Base de données | Backend     | Frontend   | Qualité de Programmation | Machines Virtuelles | Documentation |
|---------------|-----------------|-------------|------------|--------------------------|---------------------|---------------|
| VS Code       | SQLalchemy      | Flask       | Bootstrap  | Pytest                   | Proxmox             | PDoc          |
| Git           | SQLite3         | Flask-restx | Jinja      | Flake8                   | Kali Linux          | Swagger UI    |
| Azure DevOps  |                 | WTForms     | HTML       | Pylint                   | Ubuntu              | Draw.io       |
| Notepad++     |                 | Flask-login | CSS        | Docstr-coverage          | BackBox             | GanttProject  |
|               |                 | Marshmallow | JavaScript |                          |                     | Excel         |
|               |                 | Proxmoxer   | Markdown   |                          |                     | Azure DevOps  |

#### 4.6 Matériels utilisés

Chaque membre du groupe utilise 1 ordinateur et 2 écrans pour développer le projet.

Nous utilisons un serveur HP ProLiant DL380p GEN 8 qui accueille l'hyperviseur de type 1 (Proxmox). Il nous est mis à disposition par le lycée, son prix est d'environ 1300 euros.

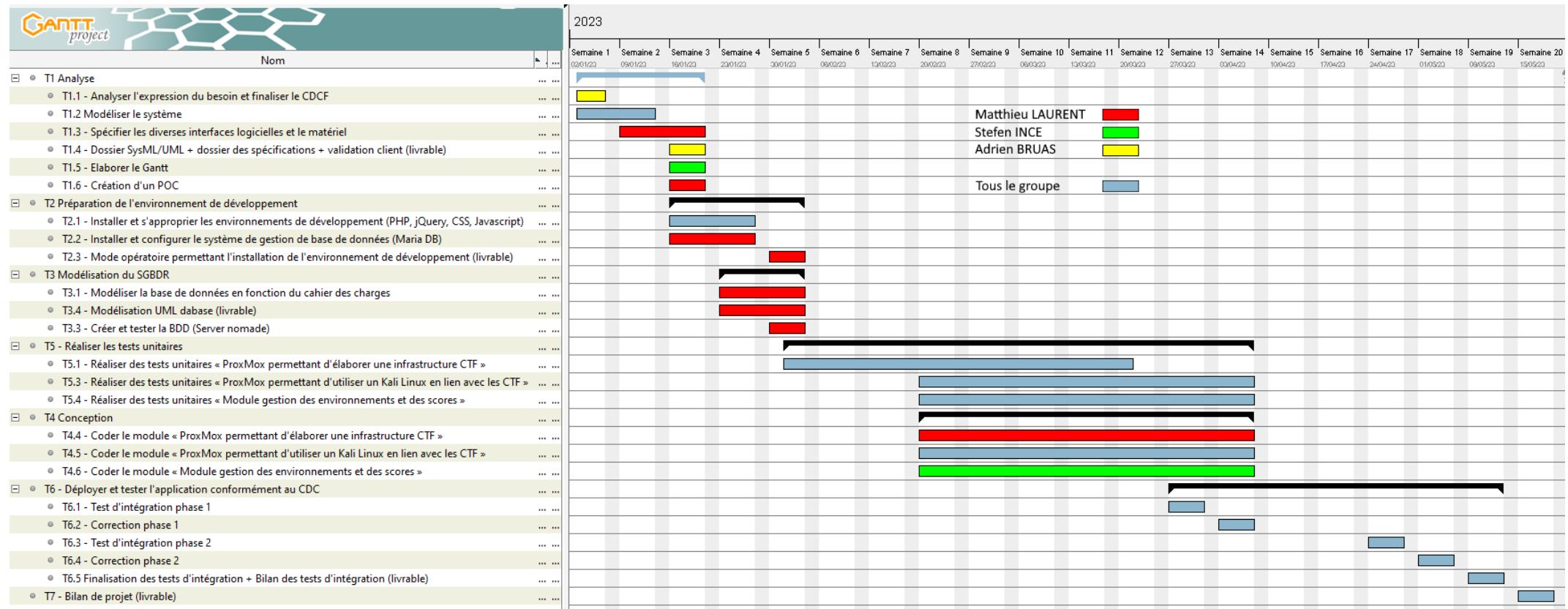


Le tableau ci-après reprend la configuration matérielle du serveur.

|                                 |                  |                   |
|---------------------------------|------------------|-------------------|
| <b>Processeur</b><br>Intel Xeon | <b>Fréquence</b> | 2.80 GHz          |
|                                 | <b>CPUs</b>      | 32                |
|                                 | <b>Sockets</b>   | 2                 |
| <b>RAM</b>                      |                  | 128Go (8 x 16 Go) |
| <b>Stockage</b>                 |                  | 1 SSD de 2To      |

Malgré un SSD de 2T, il semblerait que cela ne suffise pour faire tourner une dizaine de machines virtuelles. Il faudrait donc changer le SSD qui comment à faire son temps.

## 4.7 Planning prévisionnel



## 4.8 DevOps

### 4.8.1 Git

Git est un système de contrôle de version décentralisé et open source qui permet aux développeurs de collaborer efficacement sur un même code source.



Voici quelques points clés de sa fonctionnalité :

1. Suivi des modifications : Git suit toutes les modifications apportées au code source et permet aux développeurs de revenir en arrière à tout moment pour accéder aux versions précédentes du code.
2. Collaboration : Git facilite la collaboration entre les membres de l'équipe en leur permettant de travailler simultanément sur le même code source et de fusionner leurs modifications de manière transparente.
3. Branching et merging : Git permet aux développeurs de créer des branches pour expérimenter de nouvelles fonctionnalités sans affecter le code principal. Il facilite également la fusion des branches pour intégrer les modifications dans le code principal.
4. Gestion des conflits : Git peut détecter les conflits de fusion entre les différentes versions du code et aider les développeurs à les résoudre.
5. Stockage décentralisé : Git stocke les fichiers de code source et leur historique de modifications sur des serveurs locaux ou distants, ce qui permet aux développeurs de travailler de manière décentralisée.
6. Suivi des problèmes : Git intègre des outils de suivi des problèmes qui permettent aux développeurs de suivre les problèmes, les bogues et les améliorations de manière transparente.

En résumé, Git est une plateforme de gestion de version qui permet aux développeurs de travailler de manière collaborative, de suivre les modifications apportées au code source, de gérer les conflits de fusion et de stocker le code source et son historique de modifications de manière décentralisée.

#### 4.8.2 Azure DevOps

La méthodologie Agile Scrum est une approche de gestion de projet itérative et collaborative qui permet aux équipes de développement de logiciels de livrer des fonctionnalités rapidement et de manière flexible. Scrum est conçu pour être adaptable et réactif aux changements, et il repose sur des cycles de développement appelés "Sprints" pour atteindre des objectifs spécifiques.



Le but de la méthodologie Agile Scrum est de fournir une structure claire pour la gestion de projet tout en permettant une grande flexibilité pour répondre aux changements de priorités et aux besoins des clients. Les équipes de développement travaillent en étroite collaboration avec les parties prenantes du projet pour s'assurer que le produit final répond aux besoins de l'utilisateur et est livré en temps voulu.

Pour la gestion et le suivi du projet nous utilisons donc, Azure DevOps. Ce service cloud composé d'un environnement collaboratif intégré, il permet aussi la gestion de projet par une approche Agile avec le suivi des éléments de travail, tableaux Kanban, Azure utilise aussi Git, le système de gestion de versions est disponible (ainsi que TFVC). Il est gratuit à partir du moment où l'équipe ne dépasse pas 5 personnes.

Une fois l'environnement configuré, on accède à Azure DevOps via la barre de navigation de gauche. Elle se divise en 6 parties : Overview, Boards, Repos, Pipelines, Test Plans, Artifacts.

La partie **Overview** englobe la présentation générale du projet et le wiki que nous alimentons au fil de son évolution.

Nous pouvons également trouver les protocoles d'installation du système.

The Boards section displays a Kanban board with three columns: To Do, Doing, and Done. Each column contains several backlog items, each with a title, assignee, state, and creation date.

| Column | Backlog Item Title  | Assignee    | State    | Created Date |
|--------|---|-------------|----------|--------------|
| To Do  | 36 Migration vers flask-restx                             | adrien brus | New      | 21/02/2023   |
| Doing  | 38 Cours  | adrien brus | Approved | 21/02/2023   |
| Done   | 16 Accéder au site web de la plateforme CTF               | adrien brus | Done     | 21/02/2023   |
|        | 35 Machines virtuelles base de donnée                     |             | Done     | 25/01/2023   |
|        | 9 Permettre à l'utilisateur de voir son score sur le site | STEFEN INCE | Done     | 31/01/2023   |

La partie **Boards** liste les tâches à effectuer, qui sont traitées et les tâches à faire. On a accès à la planification et au suivi du travail. Les membres de l'équipe ajoutent et mettent à jour eux-mêmes les éléments en fonction de l'avancement.

The Repos section shows a file tree for a repository named "site-web". The tree includes .vscode, docs, site\_elysium, tests, tools, .flake8, .gitignore, .isort.cfg, .pylintrc, check\_coverage..., PY main.py, MI README.md, and requirements.de...

**Repos** est la section qui assure la gestion de code source et de version. Chaque membre de notre équipe peut donc apporter sa contribution au code. Tous les fichiers en lien avec le projet se trouvent à cet endroit.

Chaque modification apportée est enregistrée de manière à ce qu'on sache qui a fait tel ou tel modification à l'aide des commentaires faits avant chaque commit

The Commits section shows a timeline of commits:

- room john the ripper qcm (1905f9a8) by STEFEN INCE Today at 08:38
- Merge branch 'edit-room' of https://dev.azure.com/Root-Me-BTS-2022/Root%20Me%20... (dccb4a42) by INCE STEFEN mar. at 13:48
- Merge branch 'edit-room' of https://dev.azure.com/Root-Me-BTS-2022/Root%20Me%20... (01cd17b5) by Matthieu LAURENT mar. at 13:33
- Amélioration du widget question (079cecc5) by Matthieu LAURENT mar. at 13:33
- room2 (7c665398) by INCE STEFEN mar. at 13:48
- creation room john the ripper (fe093798) by STEFEN INCE 6 avr. at 17:51
- commentaires (3ead769a) by STEFEN INCE 6 avr. at 14:19
- Amélioration de l'interface d'édition (ebd16ea8) by Matthieu LAURENT 5 avr. at 16:48
- Merge branch 'edit-room' of https://dev.azure.com/Root-Me-BTS-2022/Root%20Me%20... (de0a16a1) by INCE STEFEN 5 avr. at 16:48

#### 4.9 Cahier de recettes

Ce projet est aujourd'hui fini. Toutes les options sont fonctionnelles, c'est pour cela que le cahier de recette est complet. Cependant, notre projet ne cesse de progresser et de nouvelles options supplémentaires ont été installées.

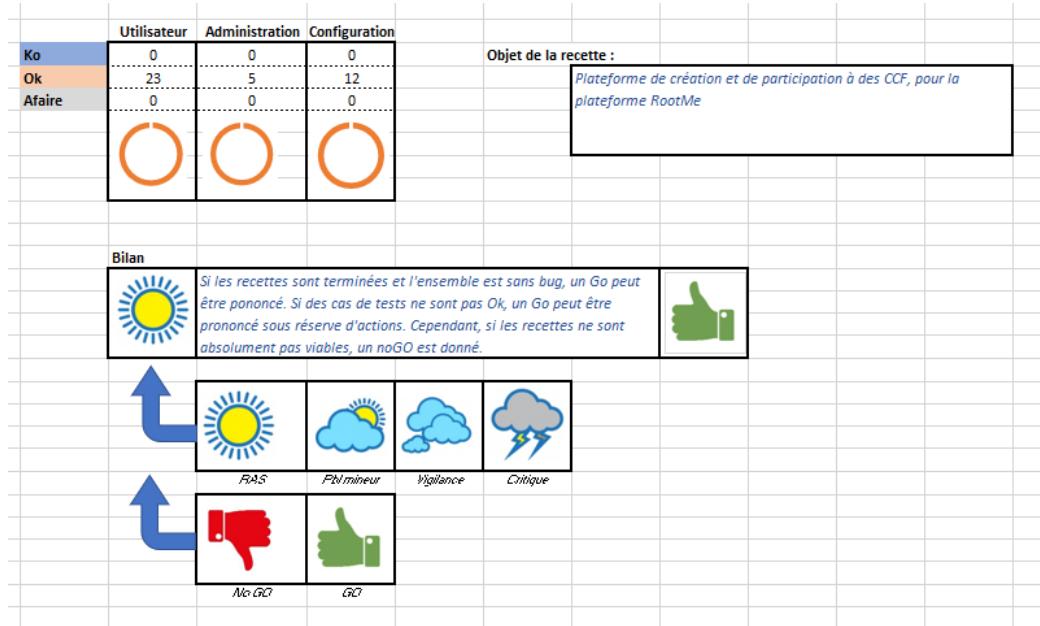


Figure 4.9.1 : Bilan du cahier de recette

| id | Prérequis                               | Description du cas de test   | résultat   | Etat |
|----|---|--|--|------|
| 1  |   | Se rendre sur la page d'accueil.   | Page d'accueil affiché sans erreurs.   |      |
| 2  | Etre connecté avec son compte.          | Quand un utilisateur avance sur une room il doit pouvoir voir sur avancé sur celle-ci. | Il voit directement sur la room sont avancé de plus il peut voir dans son profil son avancé globale sur le site. | Ok   |
| 3  | être connecté en tant qu'administrateur | Se rendre sur le dashboard administrateur  | Le dashboard est affiché et permet de modifier les informations stocké dans la base de données                   | Ok   |
| 4  | être connecté en tant qu'utilisateur    | Se rendre sur son profil   | Un tableau de classement est affiché   | Ok   |
| 5  | Etre connecté avec son compte.          | L'utilisateur peut répondre au sondage.  | Les réponses sont complétées par l'utilisateur.  | Ok   |
| 6  | Le serveur wrb doit être lancé.         | Accéder au site depuis une page internet   | La page s'affiche.   | Ok   |
| 7  | Ne pas avoir de compte.                 | L'utilisateur doit pouvoir se connecter en créant un compte.                           | L'utilisateur est connecté.  | Ok   |
| 8  | Avoir un compte.                        | Il faut pouvoir se connecter.  | L'utilisateur est connecté.  | Ok   |
| 9  | Etre connecté avec son compte.          | Participer à une room.   | L'utilisateur est sur dans la room.  | Ok   |
| 10 | Etre dans une room.                     | Lancer la machine virtuelle d'attaque.   | Les informations nécessaires pour se connecter à la machine sont affichées.                                      | Ok   |
| 11 | Etre dans une room.                     | Lancer la machine virtuelle victime.   | Les informations nécessaires de la machine virtuelle victime pour faire la room sont affichées.                  | Ok   |
| 12 | Etre dans une room.                     | Avoir accès à une machine virtuelle d'attaque.   | L'utilisateur utilise la machine virtuelle à distance.   | Ok   |
| 13 |   |  |  |      |

Figure 4.9.2 : Cahier de recette partie Utilisateur

## 5 Réalisation - Matthieu LAURENT

### 5.1 Choix des outils

Le choix d'outils adapté pour réaliser un projet est primordial. Beaucoup de réflexion a donc été apporté quant aux outils utilisés.

Pour la réalisation de ce projet, nous étions majoritairement libres de choisir les outils utilisés. Les seuls outils imposés étaient Microsoft Azure DevOps, présenté plus tôt, et Proxmox.

Proxmox est une plateforme open-source de virtualisation de serveurs basée sur la distribution Linux Debian. Elle offre une interface web permettant aux administrateurs de gérer leurs machines virtuelles.

La plateforme Proxmox propose une API RESTful qui permettent aux développeurs de contrôler les machines virtuelles et les conteneurs à distance de manière automatisé.

La gestion des interactions avec l'hyperviseur Proxmox est détaillé plus en détails dans la partie [Interaction avec l'hyperviseur](#).

Nous avons opté pour l'utilisation de Python en tant que langage de programmation, car il possède une syntaxe claire et possède de nombreux modules simples d'utilisation. Le développement en python est également extrêmement rapide. Bien que deux membres de l'équipe n'eussent jamais utilisé Python auparavant, la grande simplicité du langage leurs a permis de pouvoir rapidement apprendre.

Quant au choix du framework web pour Python, nous avons choisis entre Flask et Django :

|                                    | <b>Flask</b>   | <b>Django</b>   |
|------------------------------------|--|---|
| <b>Simplicité</b>                  | Flask est facile à apprendre et à prendre en main                                    | Django est plus compliqué à utiliser, avec une courbe d'apprentissage plus raide          |
| <b>Flexibilité</b>                 | Flask possède une grande flexibilité, avec un contrôle total des routes              | Django est plus rigide  |
| <b>Composants</b>                  | Léger, et fonctionnalités a ajouté avec des modules                                  | Batteries incluses, grand nombre de modules par défaut                                    |
| <b>Sécurité</b>                    | Fonctionnalités de sécurité offerte par les modules « Flask-Login » et « Flask-WTF » | Module de comptes, de session et de formulaire intégré, avec une forte sécurité           |
| <b>Intégration base de données</b> | Intégration via l'ORM « SQLAlchemy », supportant un grand nombre d'engines           | Intégration incluse avec Django.  |
| <b>Templates</b>                   | Utilisation de Jinja2, qui est très similaire syntaxiquement à Python.               | Utilisation du Django Template Engine (DTL), avec possibilité d'utiliser d'autre engines. |

Nous avons opté pour Flask, notamment pour sa flexibilité et sa facilité d'utilisation. Flask étant modulaire, nous avons utilisé un grand nombre de modules pour étendre les fonctionnalités :

| <b>Module Flask</b> | <b>Fonctionnalité(s) ajouté</b>  |
|---------------------|--|
| Flask-restx         | <ul style="list-style-type: none"> <li>• Création d'une API rest à partir de classes « ressources »</li> <li>• Création automatique d'une documentation OpenAPI</li> <li>• Interface web interactive Swagger UI permettant de visualiser l'API et de tester les différent endpoints.</li> <li>• Validation automatique des entrées utilisateurs</li> </ul> |

|                  |   |
|------------------|---|
| Flask-SQLAlchemy | Intégration avec l'ORM <u>SQLAlchemy</u> :<br><ul style="list-style-type: none"> <li>• Définition des tables sous forme de classes Python</li> <li>• Gestions des relations entre les modèles</li> <li>• Abstraction des requêtes à la base de données</li> </ul>   |
| Flask-WTF        | Intégration avec <u>WTForms</u> :<br><ul style="list-style-type: none"> <li>• Définition des formulaires sous forme de classes python</li> <li>• Génération automatique du code HTML pour les formulaires</li> <li>• Validation automatique des entrées utilisateurs, et gestion des erreurs</li> <li>• Protection contre les attaques de type CSRF (Cross-Site Request Forgery)</li> </ul> |
| Flask-Login      | <ul style="list-style-type: none"> <li>• Gestion automatique des sessions utilisateurs et des cookies</li> <li>• Possibilité de restreindre l'accès à des pages aux utilisateur authentifié</li> <li>• Stocke les informations de l'utilisateur dans une classe de notre choix, permettant une intégration à Flask-SQLAlchemy</li> </ul>  |
| Flask-Admin      | <ul style="list-style-type: none"> <li>• Interface web d'administration de la base de données, avec une intégration à SQLAlchemy</li> </ul>   |

Nous avons également utilisé d'autres librairies python, qui ne sont pas directement lié à Flask :

| Librairie Python | Fonctionnalité(s)   |
|------------------|---|
| SQLAlchemy-Utils | Ajoute de nombreuses fonctionnalités et de nouveaux types pour SQLAlchemy, notamment le type « <code>UUIDType</code> » permettant de stocker des UUID de manière optimisée. |
| marshmallow      | Librairie de sérialisation et désérialisation, permettant convertir des objet python vers et depuis du JSON. Intégré avec SQLAlchemy.                                       |
| markdown         | Conversion de markdown (un langage populaire de markup) vers de l'HTML.   |
| Pygments         | Syntax-highlighting du code, et permet à <u>markdown</u> de faire du rendu en couleur des « code blocks »   |
| pyChart.JS       | Intégration python de la librairies JavaScript Chart.JS, permettant d'utiliser des classes Python pour représenter des graphes puis de les utiliser dans de l'HTML          |
| customidenticon  | Génère des « identicons » à partir d'une chaîne de texte, ce qui permet de créer une photo de profile unique pour chaque utilisateur  |
| proxmoxer        | Permet d'interagir de manière simplifiée avec l'API de Proxmox.   |

En outil de développement, nous avons opté pour VS Code. Nous l'avons choisi pour sa simplicité d'utilisation, ainsi que son intégration avec les outils mentionnées précédemment.

## 5.2 Qualité du code

### 5.2.1 Documentation

Pour la documentation du code, toute nos fonctions, classes et méthodes, possède une « docstring » (commentaires de documentation) avec le style **Google**.

```
def get_n_around(lst: list[T], index: int, amount: int) -> list[T]:
    """Renvois les n éléments les plus proches de l'index "index", incluant
    l'élément à la position index.

    Args:
        lst (list[T]): La liste contenant les éléments
        index (int): L'index servant de centre dans lst
        amount (int): Combien d'objets récupéré

    Raises:
        ValueError: amount est négatif, ou index n'est pas dans la liste

    Returns:
        list[T]: Une liste de "amount" éléments, ou de len(lst) si len(lst) < amount
    """

```

Figure 5.2.1: Fonction documenté avec une docstring utilisant le style "Google"

Nous avons également utilisé l'outil **docstr-coverage** qui permet de mesurer la couverture de la documentation des fichiers, des fonctions et des classes dans un module. Un exemple de rapport de couverture est présent sur la *Figure 5.2.2* ci-dessous.

```
(.venv) PS D:\LaurentM.SNIRW\Projet_BTS\ELYSIUM\site-web> docstr-coverage .\site_elysium\
Overall statistics for 37 files:
Needed: 132 - Found: 132 - Missing: 0
Total coverage: 100.0% - Grade: AMAZING! Your docstrings are truly a wonder to behold!
```

Figure 5.2.2: Rapport généré par docstr-coverage à la fin du projet. 100% du code a été documenté.

L'intérêt d'utilisé un format prédéfini pour les docstrings est qu'il est possible pour d'autres outil tel que Sphinx ou Pdoc de les extraire automatiquement puis de générer automatiquement une documentation.

Pour ce projet, nous avons opter pour l'utilisation de Pdoc, car la documentation qu'il génère est plus facile à naviguer et plus esthétique que celle générée par Sphinx. De plus, cet outil supporte les diagrammes **mermaid** (nombreux types de diagrammes réalisés automatiquement à partir de texte) et la syntaxe **markdown** (langage de balisée léger) directement dans les docstrings, et contrairement à Sphinx, il est 100% automatique.

The screenshot displays the API Documentation for the `site_elysium.classes.VMManager` module. At the top, there's a search bar and a link to "Edit Source". Below the title, a brief description states: "Une classe qui gère les interactions avec l'hyperviseur (proxmox) ainsi que les VMs". The class hierarchy is shown with `VMManger<ABC>` at the top, followed by `ProxmoxVMManger`, and finally `VMManager`. The `VMManager` class has methods: `start_vm()`, `stop_vm()`, `setup()`, and `delete_vm()`. The `ProxmoxVMManger` class inherits these methods and adds its own: `ProxmoxVMManger()`, `start_vm()`, `stop_vm()`, `setup()`, and `delete_vm()`. A note at the bottom credits "FlagQuest - Matthieu L. / Adrien B. / Stefen I.". The footer indicates the documentation was built with `pdoc`.

**VMManager<ABC>**

- +None `start_vm(int vm_id, bool wait_until_on = True)`
- +None `stop_vm(int vm_id, bool wait_until_off = True)`
- +dict `setup(int template_id, str vm_name, bool vnc = False)`
- +None `delete_vm(int vm_id)`

**ProxmoxVMManger**

- + `ProxmoxAPI api`
- + `str node_name`
- `Lock _vm_modification_lock`
- `Allocator _mac_manager`
- `Allocator _display_port_manager`
- +`__init__(ProxmoxAPI api, str node_name, Allocator[str] mac_manager, Allocator[int] display_port_manager) : None`
- +None `start_vm(int vm_id, bool wait_until_on = True)`
- +None `stop_vm(int vm_id, bool wait_until_off = True)`
- +dict `setup(int template_id, str vm_name, bool vnc = False)`
- +None `delete_vm(int vm_id)`
- None `_test_auth()`

**VMManager(abc.ABC):**

Abstract Base Class d'un gestionnaire de VM

`@abstractmethod`

**def start\_vm(self, vm\_id: int, \*, wait\_until\_on: bool = True):**

Allume une machine virtuelle.

Arguments:

- `vm_id (int)`: L'id de la machine virtuelle.
- `wait_until_on (bool)`: Bloque jusqu'à ce que la VM soit en ligne.

Figure 5.2.3: Extraits de la documentation HTML générée par Pdoc

### 5.2.2 Tests automatiques du code

Le processus de test est essentiel pour plusieurs raisons. Tout d'abord, il permet d'éviter les régressions, c'est-à-dire de s'assurer que les fonctionnalités qui ont été précédemment implémentées et fonctionnaient correctement continuent de le faire après des modifications ultérieures. En testant, on peut identifier les problèmes potentiels avant que la solution soit déployée.

De plus, les tests permettent de vérifier des situations spécifiques auxquelles le programmeur n'aurait peut-être pas pensé. En identifiant et en testant ces cas particuliers, on peut découvrir des bugs ou des comportements inattendus qui pourraient affecter l'expérience utilisateur ou la stabilité du système.

Enfin, les tests permettent de vérifier le bon fonctionnement du logiciel dans divers états initiaux. Par exemple, il est important de tester le comportement du système lorsque l'utilisateur est connecté ou non, lorsque la base de données est vide ou contient des données spécifiques, etc. Cela permet de garantir que le logiciel est robuste et capable de gérer différentes situations.

Pour ce projet, nous avons utilisé la librairie Pytest.



Figure 5.2.4: Rapport de test Pytest intégré à VS code

Pytest possède plusieurs avantages qu'il est le plus adapté à ce projet :

- **Syntaxe simple** : pytest utilise une syntaxe simple et concise pour l'écriture des tests, ce qui rend le code de test plus lisible et facile à comprendre, et rend les cas de test plus facile à maintenir.
- **Puissance et flexibilité** : pytest propose une gamme de fonctionnalités avancées pour les tests, telles que la découverte automatique des tests, l'exécution parallèle des tests, la gestion des fixtures (configuration de l'environnement de test), la génération de rapports détaillés, etc. Ces fonctionnalités permettent d'écrire des tests complexes et de les exécuter de manière efficace.
- **Extensibilité** : pytest est hautement extensible grâce à son architecture modulaire. Il propose de nombreux plugins qui étendent ses fonctionnalités de base, permettant ainsi d'ajouter des fonctionnalités supplémentaires spécifiques aux besoins.

Notre éditeur de code, VS code, possède une intégration avec Pytest, qui permet de lancer les tests et de voir les résultats, ainsi que les erreurs et le traceback si un test échoue. Un exemple de rapport de test dans VS code est visible sur la *Figure 5.2.4* ci-dessus.

L'utilisation des **fixtures** permet de mettre en place l'environnement de test, à la demande de certains tests. Il permet également de nettoyer l'environnement de test, et les fixtures peuvent dépendre d'autres fixtures, ce qui permet une architecture très modulaire.

Les fixtures permettant de mettre en place l'application Flask ainsi qu'une connexion à la base de données sont dans la *Figure 5.2.5* ci-contre.

```
@pytest.fixture(scope="session")
def app():
    app = create_app(config=TestConfig)

    yield app

@pytest.fixture(autouse=True, scope="session")
def database(app: Flask):
    """BDD mais avec des données de test."""

    # On créer les tables
    with app.app_context():
        db.create_all()

    # On créer les données de tests
    make_test_data(app, db)
    yield db

    # C'est la fin du test, on supprime les tables
    with app.app_context():
        db.drop_all()
```

Figure 5.2.5: Exemple de fixture pytest

Pytest permet de créer de nombreux cas de tests à partir d'une seule fonction à l'aide du décorateur **parametrize**.

Ce décorateur crée automatiquement de nombreux cas de test à partir d'une liste de paramètres d'entrées.

Sur la *Figure 5.2.6* ci-contre, on utilise parametrize pour créer 9 cas de tests à partir d'une seule fonction. On peut voir les tests créés dans la *Figure 5.2.7*.

Cela permet de respecter le principe DRY (*Don't Repeat Yourself*), et dans notre cas concret, de facilement tester que certains types d'utilisateur ai ou n'ai pas accès à certaines pages avec peu de ligne de code.

Plusieurs décorateurs parametrize peuvent être ajouté à une même fonction, et toutes les combinaisons seront alors essayé.

Enfin, une extension pytest nommé `pytest-html` permet de générer un rapport de test au format HTML. Un exemple de rapport de test est disponible en [Annexe 9.1](#).

```
@pytest.mark.parametrize(
    "endpoint,expected_code,login_level",
    [
        ("main.acceuil", 200, LoginLevel.NOT_LOGGED_IN),
        ("main.acceuil", 200, LoginLevel.REGULAR_USER),
        ("main.liste_room", 200, LoginLevel.NOT_LOGGED_IN),
        ("main.liste_room", 200, LoginLevel.REGULAR_USER),
        ("main.connexion", 200, LoginLevel.NOT_LOGGED_IN),
        ("main.deconnexion", 401, LoginLevel.NOT_LOGGED_IN),
        ("main.deconnexion", 200, LoginLevel.REGULAR_USER),
        ("admin.index", 403, LoginLevel.REGULAR_USER),
        ("admin.index", 200, LoginLevel.ADMIN),
    ],
)
def test_route_access(
    client: FlaskClient,
    endpoint: str,
    expected_code: int,
    login_level: LoginLevel,
):
    """
    Vérifie que toutes les routes soient accessibles et ai le bon status code.
    """

```

*Figure 5.2.6: Utilisation de parametrize (simplifié)*

- ✓ `test_route_access`
  - ✓ main.acceuil-200-LoginLevel.NOT\_LOGGED\_IN
  - ✓ main.acceuil-200-LoginLevel.REGULAR\_USER
  - ✓ main.liste\_room-200-LoginLevel.NOT\_LOGGED\_IN
  - ✓ main.liste\_room-200-LoginLevel.REGULAR\_USER
  - ✓ main.connexion-200-LoginLevel.NOT\_LOGGED\_IN
  - ✓ main.deconnexion-401-LoginLevel.NOT\_LOGGED\_IN
  - ✓ main.deconnexion-200-LoginLevel.REGULAR\_USER
  - ✓ admin.index-403-LoginLevel.REGULAR\_USER
  - ✓ admin.index-200-LoginLevel.ADMIN

*Figure 5.2.7: Tests créés par paramétrise*

### 5.2.3 Convention de nommage

La convention de nommage utilisé est celle de la PEP 8, qui est la recommandation officielle pour les programmes python.

Il faut d'ailleurs noter que python ne possédant pas d'attributs privé, mais uniquement des attributs publics, le fait de préfixer le nom d'une variable par un underscore est purement conventionnel.

La convention de nommage de la PEP 8 est présente sur la *Figure 5.2.8* ci-contre.

Des outils ont été mis en place pour vérifier le respect de cette convention, que nous verrons prochainement.

| Type                       | Public                                | Internal                               |
|----------------------------|---------------------------------------|--|
| Packages                   | <code>lower_with_underscores</code>   |  |
| Modules                    | <code>lower_with_underscores</code>   | <code>_lower_with_underscores</code>   |
| Classes                    | <code>CapWords</code>                 | <code>_CapWords</code>                 |
| Exceptions                 | <code>CapWords</code>                 |  |
| Functions                  | <code>lower_with_underscores()</code> | <code>_lower_with_underscores()</code> |
| Global/Class Constants     | <code>CAPS_WITH_UNDER</code>          | <code>_CAPS_WITH_UNDER</code>          |
| Global/Class Variables     | <code>lower_with_underscores</code>   | <code>_lower_with_underscores</code>   |
| Instance Variables         | <code>lower_with_underscores</code>   | <code>_lower_with_underscores</code>   |
| Method Names               | <code>lower_with_underscores()</code> | <code>_lower_with_underscores()</code> |
| Function/Method Parameters | <code>lower_with_underscores</code>   |  |
| Local Variables            | <code>lower_with_underscores</code>   |  |

*Figure 5.2.8: Convention de nommage de la PEP 8*

#### 5.2.4 Formatage du code

Afin de s'assurer une uniformité du style, d'améliorer la lisibilité du code, et d'économiser du temps, un outil de formatage du code a été mis en place.

L'outil utilisé est **Black**, un formateur de code utilisé par exemple par Facebook, Dropbox et Mozilla. Il permet de rendre le code plus lisible et facile à comprendre, sans en changer le fonctionnement.

De plus, cet outil est intégré directement à VS code, et est configuré pour automatiquement formater le fichier actuel lorsque celui-ci est sauvegardé, permettant une grande efficacité.

#### 5.2.5 Lintage du code

Le lintage du code est crucial pour plusieurs raisons. Tout d'abord, il permet de détecter rapidement les erreurs de programmation, les problèmes potentiels et les pratiques non conformes aux conventions. En identifiant ces problèmes dès le départ, on évite qu'ils ne se transforment en bugs plus importants ou en erreurs de fonctionnement de l'application.

Ensuite, il contribue à améliorer la lisibilité et la maintenabilité du code. En appliquant des conventions et des bonnes pratiques de codage, le code devient plus clair, plus cohérent et plus facile à comprendre pour les développeurs. Cela facilite également la collaboration au sein d'une équipe de développement, car tous les membres adoptent un style de code uniforme.

Enfin, le lintage favorise le respect des normes et des recommandations de la communauté Python. En suivant les conventions établies, le code devient plus conforme aux standards Python, ce qui facilite la collaboration avec d'autres développeurs. De plus, le linting encourage l'adoption de bonnes pratiques de développement, ce qui se traduit par un code de meilleure qualité, plus fiable et plus performant.

Pour ce projet, nous avons utilisé deux outils de lintage : **Flake8** et **Pylint**. Les deux sont intégrés directement à VS Code, ce qui permet d'immédiatement voir les erreurs faites.

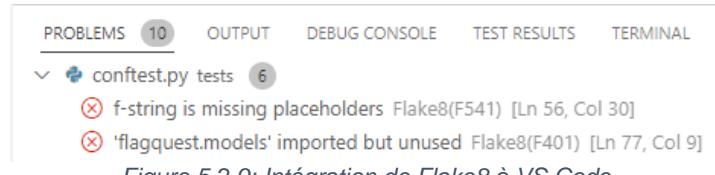


Figure 5.2.9: Intégration de Flake8 à VS Code

L'intérêt d'utiliser deux outils de lintage différent est que cela permet une vérification plus complète. En effet, Pylint effectue une analyse statique approfondie du code, tandis que Flake8 se concentre principalement sur la détection des erreurs de style, notamment du respect de la PEP 8, qui sont les recommandations officielles pour python.

## 5.2.6 Etapes pre-commit

Afin d'assurer que les bonnes pratiques et les outils mis en place précédemment soit correctement utilisé, une suite d'étapes pre-commit ont été créer.

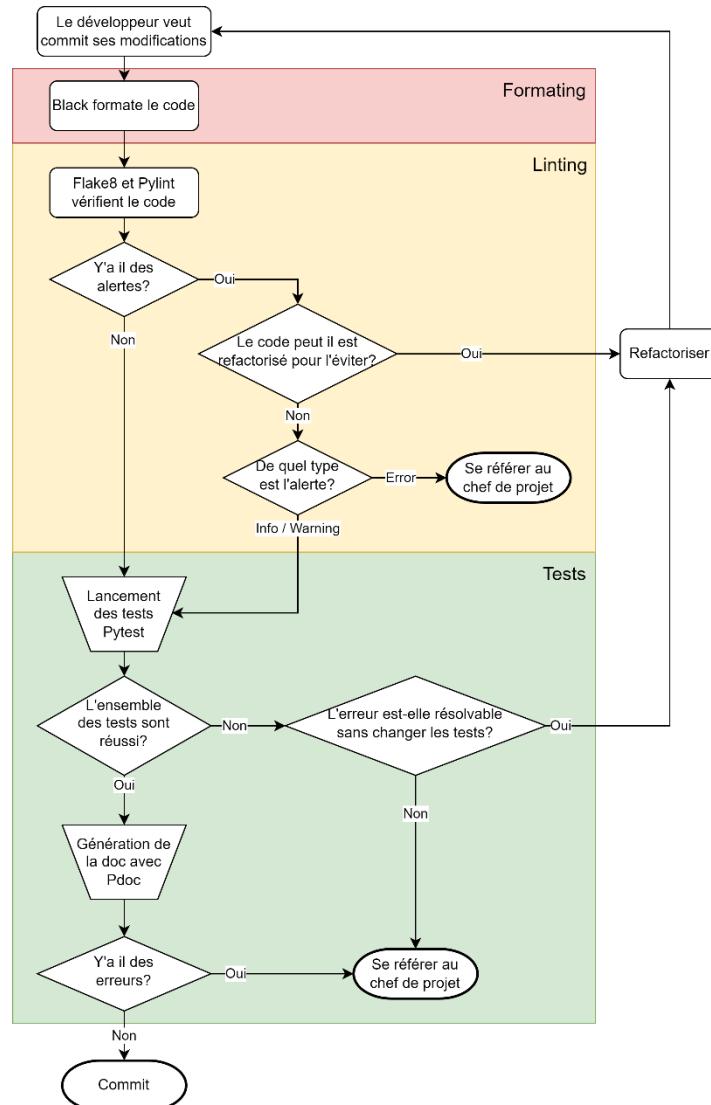
Elles doivent être réalisé par les développeurs avant chaque « commit » (publication des modifications locales vers le serveur Azure DevOps).

Ces étapes sont automatisées, à l'exception des étapes représenté par un trapèze, permettant une grande efficacité.

L'entièreté de ce processus pre-commit est rapide à réalisé, prenant en général moins d'une minute à se finir.

Certain cas particulier nécessite cependant parfois de revoir certains aspects du projet. Pour ces cas-là, il est indiqué de se référer au chef de projet.

Les étapes à réaliser sont représenter en tant que flowchart dans la *Figure 5.2.10* ci-contre.



*Figure 5.2.10: Etapes pre-commit*

## 5.2.7 Structure du projet

Il est crucial de bien structurer les fichiers de son projet pour plusieurs raisons. Tout d'abord, une structure claire et organisée facilite la navigation et la compréhension du projet pour les développeurs qui y travaillent, ainsi que pour les futurs administrateurs qui souhaitent comprendre comment le projet fonctionne. Une structure logique et intuitive peut également faciliter la maintenance du projet en permettant de trouver plus rapidement les fichiers pertinents pour effectuer des modifications.

C'est pour cela que la structure du projet a été décidé avant d'entamer la programmation.

Cette structure est indiquée sur la *Figure 5.2.11* ci-dessous.

En programmation python, il est courant de faire de son projet un module importable, qui expose les fonctions nécessaires au lancement dudit projet. C'est pour cela que le dossier « flagquest » est un module python importable, exposant la fonction requise au lancement du site, la fonction **App factory** (sur laquelle nous reviendront).

Ce module est détaillé sur la *Figure 5.2.12* ci-dessous.

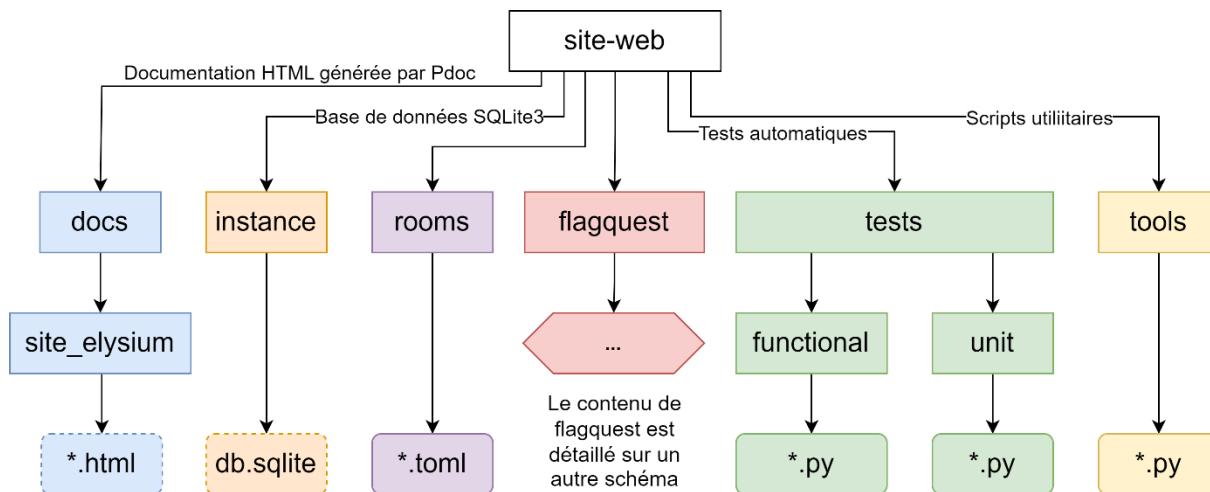


Figure 5.2.11: Arborescence du projet. Les rectangles sont des dossiers, et les bords ronds sont des fichiers. Des bords en pointillé signifie que le fichier est créé automatiquement par le programme.

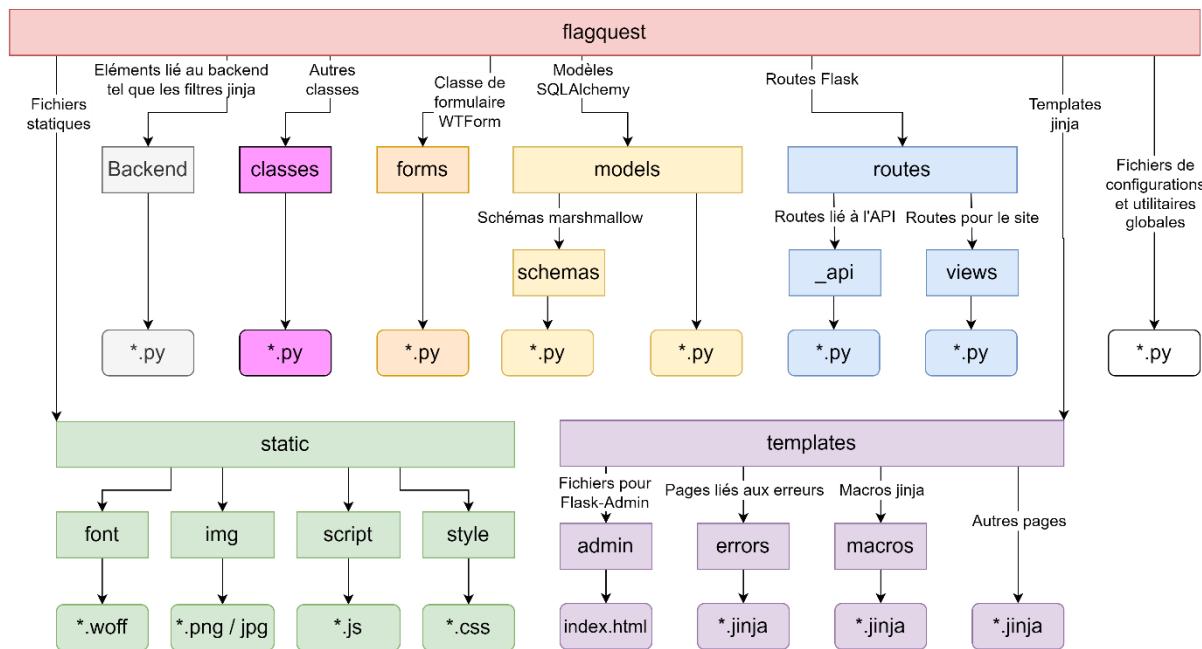


Figure 5.2.12: Arborescence du module

### 5.2.8 Environnement virtuel

Afin d'isoler le projet du reste de l'interpréteur python par défaut, un environnement virtuel, appeler un **venv**, a été mis en place. Celui-ci permet au projet d'utiliser sa propre version de python, et les librairies utilisé sont installer dans l'environnement virtuel afin d'éviter tout conflits avec d'autre projets sur la même machine.

### 5.3 Outils de template Jinja2

**Jinja2** est un moteur de templates pour Python. Il permet de générer des documents dynamiques en combinant du code Python avec des templates HTML ou d'autres formats de fichiers. C'est le moteur de templates par défaut de Flask.

Un template, dans le contexte de Jinja, est un fichier contenant une structure de base prédéfinie, généralement écrite dans un langage de balisage comme HTML. Il sert de modèle pour générer des documents dynamiquement.

Les templates Jinja peuvent contenir des **blocks**, qui sont en fait des sections du document. D'autre template peuvent **étendre** (extend) ce template et remplacer uniquement certains blocks. Cela permet donc d'avoir un template de base, qui contient par exemple le header, le footer, et d'autres éléments qui doivent apparaître sur toutes les pages du site.

Notre projet a donc un template « base.jinja » que tous les autres templates étendent, ce qui permet d'écrire des templates très concis, de ne pas avoir de duplication du code, et d'avoir une charte graphique uniforme.

```
% extends 'base.jinja'

{% block title %}
{{ super() }} - Utilisation
{% endblock %}

{% block content %}
<div class="container text-light">
<h1 class="display-3">Utilisation élève</h1>

<h2 id="participer-room">Participer à une room</h2>
<p>Pour participer à une room, il suffit de se rendre sur la page de la room, puis de cliquer le boutton <code>Rejoindre la room</code>. <br>
Ensuite, il ne te reste plus qu'a <a href="#utilisation-vms">lancer les machines virtuelles</a> et répondre aux questions!
</p>

<h2 id="utilisation-vms">Utilisations des machines virtuelles.</h2>
<p>Une fois que tu a rejoint une room, il apparait en haut de la page un boutton permettant de lancer une VM d'attaque. Clique simplement ce boutton, et les informations de la VM tel que l'adresse IP et le port VNC seront affiché. <br>
Si la room contient des machines victimes, tu pourras similairement les lancées via un autre boutton en haut de la page.
</div>
{% endblock %}
```

Figure 5.3.1: Exemple d'une template Jinja, remplaçant les blocks title et content du template base.jinja

Les templates peuvent également utiliser des variables et même du code, comme on peut le voir dans la *Figure 5.3.1* ci-dessus. Ici, la macro super() est utilisé, qui renvoie le contenu du block correspondant dans le parent (ici, le contenu du block « title » dans base.jinja).

Une autre fonctionnalité particulièrement utile de Jinja est l'utilisation de macros. Les macros sont l'équivalent Jinja des fonctions. Elles peuvent prendre des arguments en entrées, et lorsqu'elles sont appelé, elles renvois leur contenus. Cela peut être utilisé pour éviter la duplication de code, et pour respecter le principe DRY (Don't Repeat Yourself)

```
% macro room_widget(name, description, url_name, size=6) -%
<div class="col-md-{{size}} p-3">
<div class="{{config.BACKGROUND_BLOCK}} h-100 p-5 rounded-3 {{config.TEXT_COLOR_BLOCK}}>
<h2>{{name|e}}</h2>
<p>{{description|markdown}}</p>
<a href="{{url_for('main.room', room_url_name=url_name) }}"
    class="btn rounded-1 border-2 active text-info fw-semibold px-4 py-2 mt-5" role="button">Lancer</a>
{%- if current_user and current_user.is_admin==true %}
<a href="/room/supervision/{{url_name}}"
    class="btn rounded-1 border-2 active text-info fw-semibold px-4 py-2 mt-5" role="button">Supervision</a>
{%- endif%}
</div>
</div>
{%- endmacro %}
```

Figure 5.3.2: Exemple de macro Jinja2

La macro peut ensuite être importé puis utilisé dans d'autres templates, comme on peut le voir sur la *Figure 5.3.3* ci-dessous.

```
{% for r in rooms %}  
{{room_widget(r.name, r.description, r.url_name)}}  
{% endfor %}
```

*Figure 5.3.3: Exemple d'utilisation d'une macro*

Enfin, il est possible de transmettre des variables aux templates Jinja depuis le code python. Cela permet de par exemple transmettre des objets SQLAlchemy au template, comme on peut le voir sur la *Figure 5.3.4* ci-dessous.

```
@main.route("/classement")
def classement():
    """Un classement de tout les utilisateurs"""

    # tri par ordre score : décroissant
    users = sorted(User.query.all(), key=lambda u: u.score, reverse=True)

    return render_template("classement.jinja", users=users)
```

Figure 5.3.4: Les utilisateurs de la base de données sont transmis au template

## 5.4 Structure Flask

Lors de la conception de l'application Flask, nous avons suivi le principe recommandé d'*app factory*.

Une *app factory* Flask est une approche organisée pour la création d'applications web extensibles et maintenables. Plutôt que de construire une seule instance d'application Flask, l'approche *app factory* consiste à créer une fonction « usine » qui génère une nouvelle instance d'application à chaque appel. Cela permet une meilleure modularité, par exemple en permettant de lancer le serveur avec plusieurs configurations à la fois.

Dans notre cas, nous avons ajouté un argument « config » à notre fonction app factory, ce qui permet de créer une application avec une configuration spécifique.

```
def create_app(config: object = Config) -> Flask:  
    """App factory pour l'application Flask  
  
    Args:  
        config (object, optional): L'objet de configuration à charger. Defaults to Config.  
  
    Returns:  
        Flask: l'app Flask  
    """  
  
    app = Flask(  
        __name__,  
        static_folder="static",  
        template_folder="templates",  
        static_url_path="/static",  
    )  
    ...
```

*Figure 5.4.1: Début de la fonction app factory*

Cela nous permet de changer la configuration en fonction de si l'application est en déploiement ou en train de réaliser des tests unitaires par exemple.

Changer la configuration permet également de changer l'esthétique du site, mais cela sera traité plus en détail dans la partie de Adrien BRUAS.

Cette app factory fonctionne en créant une instance d'une application Flask, en y chargeant toutes les extensions mentionnées précédemment (Flask-SQLAlchemy par exemple), en chargeant la configuration, ici depuis l'objet config et également depuis le fichier app\_config.py (l'objet config prenant préférence en cas de clés dupliquées), et enfin en y enregistrant des « blueprints ».

Les blueprints sont des composants essentiels de l'architecture d'une application Flask. Ils permettent de structurer le code en regroupant des fonctionnalités spécifiques par thème ou par module. Les blueprints agissent comme des mini-applications indépendantes, avec leurs propres routes, modèles, vues et fichiers statiques. Ils peuvent être enregistrés auprès de l'application principale pour être utilisés et étendus, ce qui permet une plus grande modularité.

Notre application comprend 2 blueprints, un pour les pages web qui sont prévus pour les utilisateurs depuis un navigateur web, par exemple la page d'accueil, et un pour l'API.

Sur ces blueprints, on enregistre des « routes ». Une route permet d'associer une règle d'URL avec une fonction, ce qui est fait avec un décorateur avant la fonction :

```
@main.route("/profile", defaults={"username": None})
@main.route("/profile/<username>")
def profile(username: str | None):
    """La page de profil de l'utilisateur"""
```

Figure 5.4.3: Les décorateurs de la route pour la page profile

La route sur la Figure 5.4.3 ci-dessus associe les pages /profile et /profile/<username> avec la fonction profile. Cette route est enregistrée dans le blueprint main.

Le <username> dans la deuxième route est une variable qui est passé à la fonction. Cela signifie que se rendre sur la page /profile/matthieu appellera la fonction profile avec la variable username qui vaudra matthieu. Si on se rend simplement sur la page /profile, cette même variable vaudra None, ce qui permet de gérer ce cas particulier.

```
class Config:
    """
    La configuration de production de Flask.
    """

    SECRET_KEY = "ThisIsASecureKey"

    # SQLAlchemy
    SQLALCHEMY_DATABASE_URI = "sqlite:///db.sqlite"

class TestConfig(Config):
    """
    La configuration de test de Flask.
    """

    TESTING = True

    # SQLAlchemy
    SQLALCHEMY_DATABASE_URI = "sqlite:///test_db.sqlite"

    # WTForms
    WTF_CSRF_ENABLED = False
```

Figure 5.4.2: Les différentes configurations utilisées

## 5.5 Structure de l'API

Les routes de l'API sont structurées différemment que les pages principales. En effet, à la place de routes individuelles, l'API est structurée par rapport à des ressources.

Une ressource représente un objet ou une entité spécifique avec lequel les utilisateurs de l'API peuvent interagir. Par exemple, dans le cadre de notre projet, nous avons une ressource **Question** qui correspond à une question d'une room.

Au lieu de définir des routes séparées pour chaque opération, les routes de l'API sont généralement conçues en utilisant des conventions RESTful (Representational State Transfer). Ces conventions définissent des modèles de routes basés sur les opérations CRUD (Create, Read, Update, Delete) effectuées sur les ressources.

Ces ressources sont représentées par une classe python héritant de la classe **Resource** de Flask-restx.

Les ressources sont associées à une règle d'URL, tout comme une route classique. Mais on peut également y ajouter d'autre décorateurs permettant de documenter le code réponse.

Cette classe implémente des méthodes correspondant au différentes opération CRUD. Ces méthodes peuvent également être décoré avec le décorateur `marshal_with`, qui permet de définir quel sont les champs qui seront présent dans la réponse, à des fins de documentation mais également de validation, Flask-restx se chargeant de retirer tout champ en trop et de vérifier que tous les champs documentés sont bien présents, ainsi que de convertir les champs dans le bon type et de sérialisé le tout en JSON.

Un extrait d'une ressource Flask-restx est présent sur la *Figure 5.5.1* ci-dessous, et est présent en détail en *annexe 9.7*.

```

@room_namespace.route("/question/<id>")
@room_namespace.response(200, "Succès")
@room_namespace.response(401, "L'utilisateur n'est pas connecté")
@room_namespace.response(403, "L'utilisateur n'a pas les privilèges requis")
@room_namespace.response(404, "La question n'existe pas")
class QuestionResource(Resource):
    """Informations lié à une question"""

    @room_namespace.marshal_with(question_model, as_list=False)
    def get(self, id: str):
        """Récupère les informations lié a une question."""
        ...

    @room_namespace.marshal_with(question_model, as_list=False)
    def post(self, id: str):
        """Modifie les informations lié a une question."""
        ...

    def delete(self, id: str):
        """Supprime une question"""
        ...

```

*Figure 5.5.1: Classe représentant la ressource Question. Le contenu des fonctions a été retiré par simplification.*

Comme expliqué plus tôt, les méthodes sont documentées avec la sortie attendue. Ceci est fait à l'aide de modèles qui définissent explicitement chaque champ de la réponse JSON ainsi que son type.

```
question_model = room_namespace.model(
    "Question",
    {
        "room": fields.Integer,
        "id": fields.Integer,
        "solved_questions_data": fields.List(
            fields.Nested(
                room_namespace.model(
                    "SolvedQuestionData",
                    {"user_id": fields.Integer, "question_id": fields.Integer},
                )
            )
        ),
        "points": fields.Integer,
        "prompt": fields.String,
    },
)
```

Figure 5.5.2: Modèle correspondant à la représentation d'une question

Ensuite, pour simplifier, ces ressources sont regroupées en fonction de leur thème dans des groupes appelés **namespace**. Ces namespaces sont ensuite ajoutés à un objet appelé **API** de Flask-restx, qui contient les métadonnées de l'API et est une sous-application. Enfin, cet objet API est ajouté à un blueprint Flask, qui est ensuite enregistré dans l'application Flask en utilisant l'app factory.

L'intérêt d'un tel formalisme est qu'il permet à Flask-restx de convertir automatiquement ce code en une documentation OpenAPI, ainsi qu'en une interface web interactive documentant l'API via SwaggerUI.

| Room          |                            | Opérations liés aux rooms  |
|---------------|----------------------------|--|
| <b>POST</b>   | /answer_question           | Permet à l'utilisateur de répondre à une question et de savoir si il a juste |
| <b>POST</b>   | /join_room/{room_url_name} | Permet à un utilisateur de rejoindre une room                                |
| <b>DELETE</b> | /question/{id}             | Supprime une question  |
| <b>POST</b>   | /question/{id}             | Modifie les informations lié à une question                                  |
| <b>GET</b>    | /question/{id}             | Récupère les informations lié à une question                                 |
| <b>GET</b>    | /room/{url_name}           | Récupère les informations lié à une room                                     |

Figure 5.5.3: Extrait de la documentation interactive SwaggerUI

Ce qui rend cette interface particulièrement intéressante est qu'elle permet à l'utilisateur de voir à quoi ressemble la réponse d'une page donnée, et à quoi correspond chaque status code qu'il peut obtenir.

The screenshot shows the SwaggerUI interface for a 'Question' resource. At the top, it displays a 'GET /question/{id}' endpoint with a description: 'Récupère les informations liées à une question'. Below this, the 'Parameters' section lists two fields: 'X-Fields' (optional) and 'id' (required). The 'Responses' section shows four status codes: 200 (Success), 401 (User not connected), 403 (User does not have the required privileges), and 404 (Question does not exist). The 200 response example is shown as a JSON object:

```
{
  "room": 0,
  "id": 0,
  "solved_questions_data": [
    {
      "user_id": 0,
      "question_id": 0
    }
  ],
  "points": 0,
  "prompt": "string"
}
```

Figure 5.5.4: Extraits de la documentation SwaggerUI, montrant la ressource Question

Cette interface permet également de tester directement l'API, en fournissant les paramètres d'entrée et en obtenant la réponse du serveur. Cela fournit également la commande curl associé.

The screenshot shows the SwaggerUI interface for a REST API. At the top, there's a header bar with a 'GET' button, the URL '/question/{id}', and a description 'Récupère les informations liées à une question'. Below this is a 'Parameters' section with two entries: 'X-Fields' (an optional fields mask) and 'id \* required' (a string path parameter). There are 'Execute' and 'Clear' buttons at the bottom of this section. Under 'Responses', there's a 'Curl' section with a command line, a 'Request URL' section with 'http://127.0.0.1:5000/api/question/1', and a 'Server response' section showing a JSON response body. The response body is a JSON object with fields like 'room': 1, 'id': 3, 'solved\_questions\_data': [{}], 'points': 5, and 'prompt': 'Quel était le mot de passe du serveur FTP?'. The 'Response headers' section shows 'connection: close', 'content-length: 225', 'content-type: application/json', 'date: Tue, 23 May 2023 13:05:37 GMT', and 'server: Werkzeug/2.2.3 Python/3.10.10'.

Figure 5.5.5: Exemple d'une requête interactive depuis l'interface SwaggerUI pour la ressource Question

Pour conclure, l'application Flask est structuré de manière très modulaire, ce qui permet de facilement réutiliser, ajouter, supprimer ou modifier des composants.

Seul les routes Flask sont indissociables de leur blueprint, tous les autres composants sont interchangeable car utilisant un couplage léger.

Une représentation de la structure de l'application Flask est présent sur la Figure 5.5.6 ci-contre. Il utilise les mêmes conventions qu'un diagramme de classe, bien que les routes Flask ne soient pas des classes mais des fonctions.

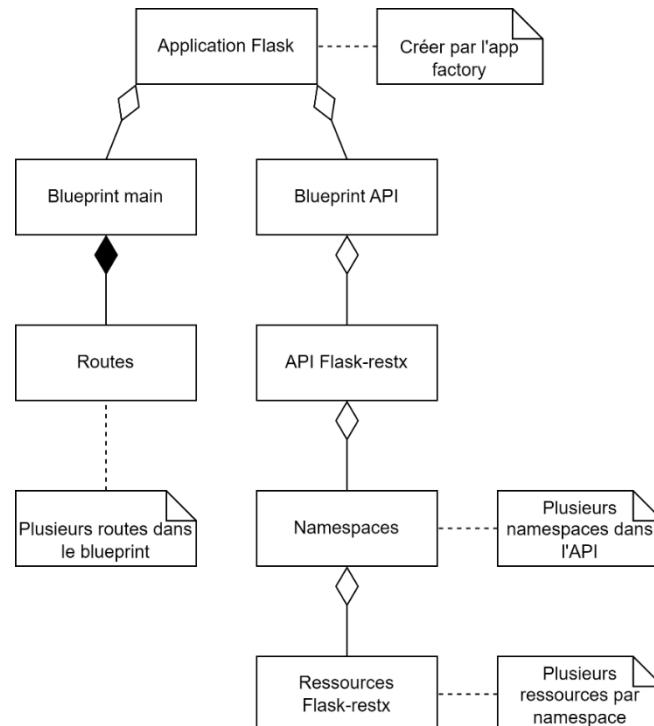


Figure 5.5.6: Structure de l'application Flask

## 5.6 Formulaires avec WTForms

Pour les formulaires du site, nous avons utilisé la bibliothèque WTForms. Elle permet de créer des formulaires HTML à l'aide de classes python, et elle présente plusieurs avantages :

- Création automatique du code pour le formulaire, avec validations coté client
- Validation du formulaire coté serveur
- Protection contre les attaques CSRF (Cross-Site Request Forgery)

De plus, WTForms peut être utilisé directement dans un template Jinja, simplifiant grandement le code.

```
class LoginForm(FlaskForm):
    """Formulaire de connexion"""

    login = wtforms.StringField(
        "Nom d'utilisateur", validators=[wtforms.validators.DataRequired()])
    )
    """Le nom d'utilisateur de l'utilisateur"""
    password = wtforms.PasswordField(
        "Mot de passe", validators=[wtforms.validators.DataRequired()])
    )
    """Le mot de passe de l'utilisateur"""
    submit = wtforms.SubmitField("Valider")
    """Le bouton pour envoyer le formulaire"""
```

Figure 5.6.1: Un formulaire WTForms servant pour la connexion des utilisateurs

Un exemple de l'utilisation de WTForms avec Flask est disponible en annexe 9.5 (template jinja) et annexe 9.6 (code Flask).

## 5.7 Base de données

Ce projet nécessite une persistance des données, et l'utilisation d'une base de données est donc particulièrement adapté.

Afin d'identifier les données devant être stocké, ainsi que les différentes entités et leurs relations, le **MCD** (Modèle Conceptuel des Données) a été réalisé (Figure 5.7.1 ci-contre).

Ce diagramme nous a permis d'identifier les quatre entités principales.

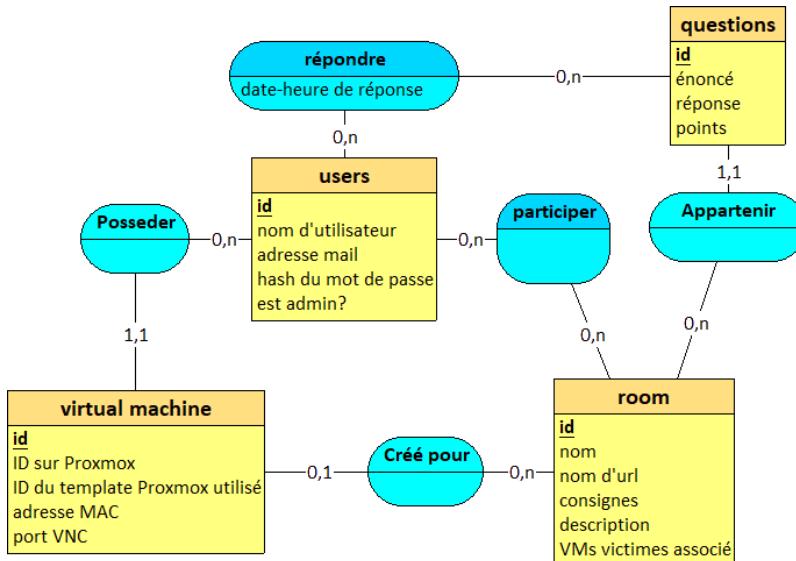


Figure 5.7.1: Modèle Conceptuel des Données

Pour la réalisation de la base de données, nous avons opté pour l'utilisation d'une **ORM** (Object-Relational Mapping). L'ORM que nous avons utilisé est **SQLAlchemy**, qui propose une intégration avec Flask.

Une ORM permet de faire le lien entre une base de données relationnelle et un langage de programmation orienté objet. L'ORM agit comme une couche d'abstraction entre le code et la base de données. L'utilisation d'une ORM permet d'interagir avec la base de données en manipulant des objets plutôt qu'à l'aide de requêtes SQL.

```
class User(db.Model):
    """Un utilisateur du site web"""

    __tablename__ = "users"

    id: Mapped[int] = mapped_column(primary_key=True)
    # collation="NOCASE" signifie que les vérifications ne sont pas sensibles à la case
    username: Mapped[str] = mapped_column(String(collation="NOCASE"), unique=True)
    email: Mapped[str] = mapped_column(String(collation="NOCASE"), unique=True)
    password_hash: Mapped[str]
    """

    Hash du mot de passe, de la forme 'method$salt$hash'
    Voir également https://werkzeug.palletsprojects.com/en/1.0.x/utils/#werkzeug.security.generate\_password\_hash
"""

    is_admin: Mapped[bool] = mapped_column(default=False)

    solved_questions_data: Mapped[list["SolvedQuestionData"]] = relationship(
        back_populates="user", cascade="all, delete, delete-orphan"
    )
    joined_rooms: Mapped[list["Room"]] = relationship(
        secondary=room_user, back_populates="users"
    )
```

Figure 5.7.2: Exemple d'une classe représentant une entité dans l'ORM

Un exemple d'un modèle complet est disponible en annexe 9.8.

Les avantages de l'utilisation d'une ORM sont :

- Portabilité : Grâce au niveau d'abstraction, le code est indépendant de moteur de base de données sous-jacent. Il est donc possible de facilement migrer d'un moteur vers un autre, par exemple de passer de SQLite3 à PostgreSQL ou MariaDB.
- Simplification du code : Comme le code manipule des objets et non plus requêtes SQL, le code est bien plus simple à comprendre et à écrire
- Gestion automatique des relations : Une ORM permet d'automatiquement récupérer les objets liés via un attribut, ce qui évite d'avoir à se soucier des relations. Par exemple, il est possible d'accéder à la liste des questions d'une room avec le code **my\_room.questions**, et même d'ajouter ou retirer des questions via cette liste. L'ORM permet aussi de gérer automatiquement la suppression des objets orphelin, ce sur quoi nous reviendront.

Cependant, l'utilisation d'une ORM ne supprime pas le besoin pour des tables de jointures, que l'utilisateur doit créer via des classes.

Nous avons donc réalisé un diagramme Entité-Association (*Figure 5.7.3 ci-dessous*) afin de mettre en évidence les différentes classes python nécessaire.

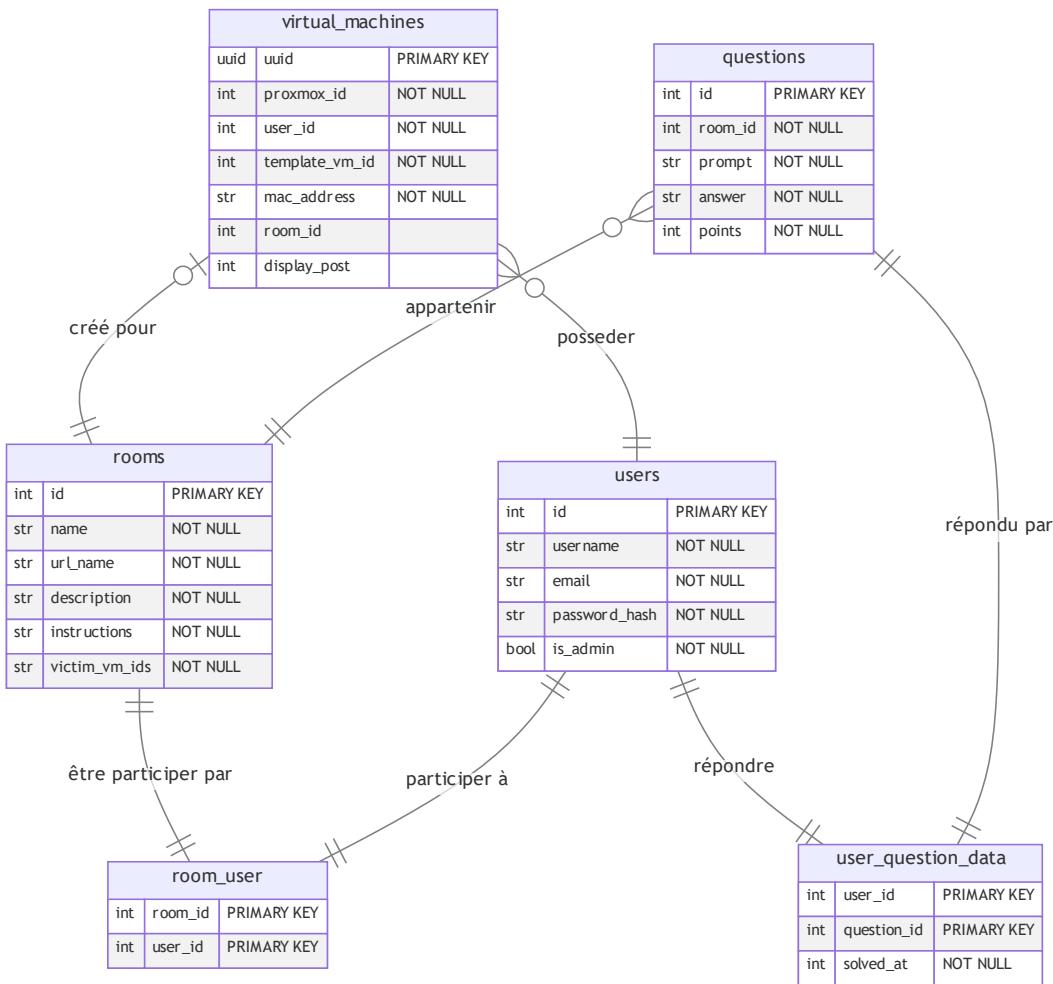


Figure 5.7.3: Diagramme Entité-Association, utilisant la notation "Crow's Foot"

Comme on peut le constater, certaines de ces entités peuvent être considérées comme entité « parents » d'une relation. Par exemple, dans la relation entre rooms et questions, la table rooms peut être considérée comme parent, car une question ne peut pas exister en dehors d'une room.

Pour éviter qu'une telle situation arrive, nous avons configuré des « suppression en cascade ». Une suppression en cascade dans SQLAlchemy est un mécanisme qui permet de supprimer automatiquement les objets associés à un objet principal lorsqu'il est supprimé de la base de données.

Il existe deux types de suppression en cascade dans SQLAlchemy :

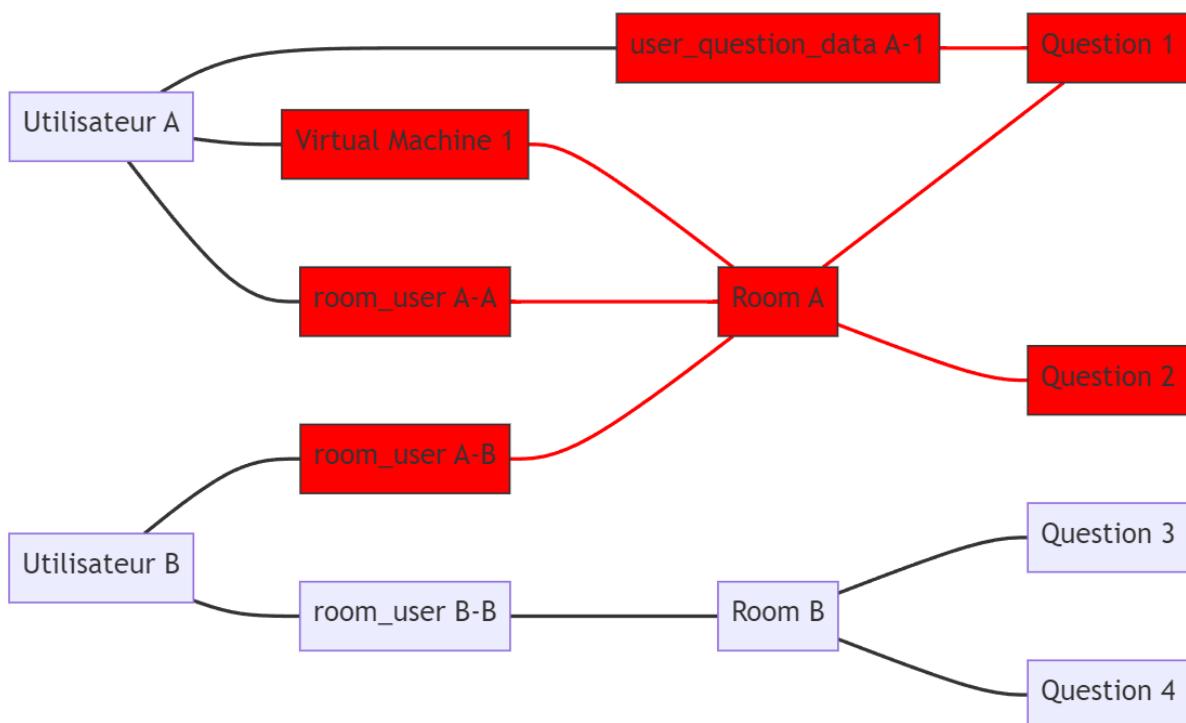
- delete : Si un objet parent est supprimé, alors les objets enfants qui lui sont associés sont également supprimés.
- delete-orphan : Si un objet enfant se retrouve désassocié d'un objet parent et qu'un nouveau parent ne lui est pas associé avant la commit, alors cet objet enfant est supprimé.

Il est possible pour une relation de posséder les deux types de suppressions en cascade à la fois.

Ces suppressions en cascade permettent donc de ne pas avoir d'objets « orphelin » dans la base de données, mais également de ne pas avoir de problèmes avec les clés étrangère avec des contraintes NOT NULL. Par exemple, une question doit avoir une room associé, car la clé étrangère à une contrainte NOT NULL. Mais étant donné que la suppression d'une room supprime automatiquement les questions associées, cette contrainte ne pose pas de problème.

Cette suppression en cascade est configurée automatiquement coté base de données si le moteur de base de données le supporte (par exemple avec ON DELETE CASCADE dans MySQL), ce qui permet une optimisation. Sinon, SQLAlchemy se charge de la suppression, assurant que le code fonctionnera peu importe le moteur utilisé.

Un exemple de suppression en cascade est présent sur la *Figure 5.7.4* ci-dessous. La seule entité supprimée explicitement est « Room A », les autres suppressions sont causées par la suppression en cascade.



*Figure 5.7.4: Exemple de suppression en cascade lors de la suppression de Room A. Les entités supprimées sont marquées en rouge.*

De plus, on veut éviter une désynchronisation entre la base de données et l'hyperviseur. Cela signifie que l'on veut que la machine virtuelle sur l'hyperviseur soit automatiquement supprimée lors de sa suppression dans la base de données.

Pour ce faire, on utilise un **listener**. C'est un décorateur de fonction qui permet d'appeler cette fonction automatiquement lors d'un certain évènement dans la base de données. Ici, on appelle une fonction automatiquement lorsqu'une VirtualMachine (machine virtuelle) est supprimé dans la base de données.

Cette fonction se chargeant de la suppression sur l'hyperviseur est représenté sur la *Figure 5.7.5* ci-dessous. Comme on peut le voir, SQLAlchemy permet de gérer ce cas de manière très simple grâce au lien réalisé entre python et la base de données.

```

@listens_for(VirtualMachine, "after_delete")
def after_vm_delete(mapper, connection, target: VirtualMachine):
    """Supprime automatiquement la VM proxmox correspondante lorsqu'elle est supprimé de la VM"""
    mgr = get_vm_manager()
    mgr.delete_vm(target.proxmox_id)

```

Figure 5.7.5: Listener permettant la suppression automatique des VMs sur l'hyperviseur

## 5.8 Interaction avec l'hyperviseur

Pour ce projet, l'hyperviseur utilisé est Proxmox. Cependant, il peut être intéressant d'utiliser un autre hyperviseur plus tard, c'est pour cela que nous avons utilisé une **ABC** (*Abstract Base Class*, Classe de Base Abstraite).

Une ABC définit des méthodes abstraites, que les classes héritant de l'ABC doivent implémenter. Cela permet aux développeurs les classes héritant de l'ABC sans se soucier de l'implémentation, car l'interface est déjà clairement définie. Cela permet aussi de pouvoir changer d'implémentation sans changer le code qui dépendait sur l'ABC.

Comme on peut le voir sur la *Figure 5.8.1* et la *Figure 5.8.2* ci-contre, l'ABC ne possède aucun code concret, mais définit uniquement la **signature** de la fonction (les arguments qu'elle prend en entré, et ce qu'elle renvoie en sortie). Les classes héritant de cette ABC, elles, possèdent une implémentation concrète.

Les fonctions ayant besoin d'un gestionnaire de VM en entré peuvent simplement définir qu'ils ont besoin d'un VM manager, et n'importe quelle classe héritant de VMManger pourra lui être donnée.

Le code complet pour la classe abstraite VMManger est disponible en annexe 9.2.

```

class VMManger(ABC):
    """
    Abstract Base Class d'un gestionnaire de VM
    """

    @abstractmethod
    def start_vm(self, vm_id: int, *, wait_until_on: bool = True):
        """Allume une machine virtuelle.

        Args:
            vm_id (int): L'id de la machine virtuelle.
            wait_until_on (bool): Bloque jusqu'à ce que la VM soit en ligne.
        """
        ...

```

Figure 5.8.1: ABC définissant l'interface d'un gestionnaire de VM

```

class ProxmoxVMManger(VMManger):
    """
    Gère les interactions avec l'hyperviseur (proxmox) ainsi que les VMs
    """

    def start_vm(self, vm_id: int, *, wait_until_on: bool = True):
        self.api.nodes(self.node_name).qemu(vm_id).status.start.post()
        if wait_until_on:
            # Tant que le status de la VM n'est pas "running"
            while (
                self.api.nodes(self.node_name)
                .qemu(vm_id)
                .status.current.get()["status"]
                != "running"
            ):
                time.sleep(1)

```

Figure 5.8.2: Implémentation concrète d'un gestionnaire de VM

Comme on peut le voir sur la *Figure 5.8.3* ci-contre, l'implémentation concrète du gestionnaire de VM utilisant proxmox implémente beaucoup d'autre méthode et attributs, mais qui sont préfixé d'un underscore, afin d'indiquer qu'elles ne sont privées et que les utilisés est une mauvaise idée, car elle pourrait ne pas être présentes dans une autre implémentation.

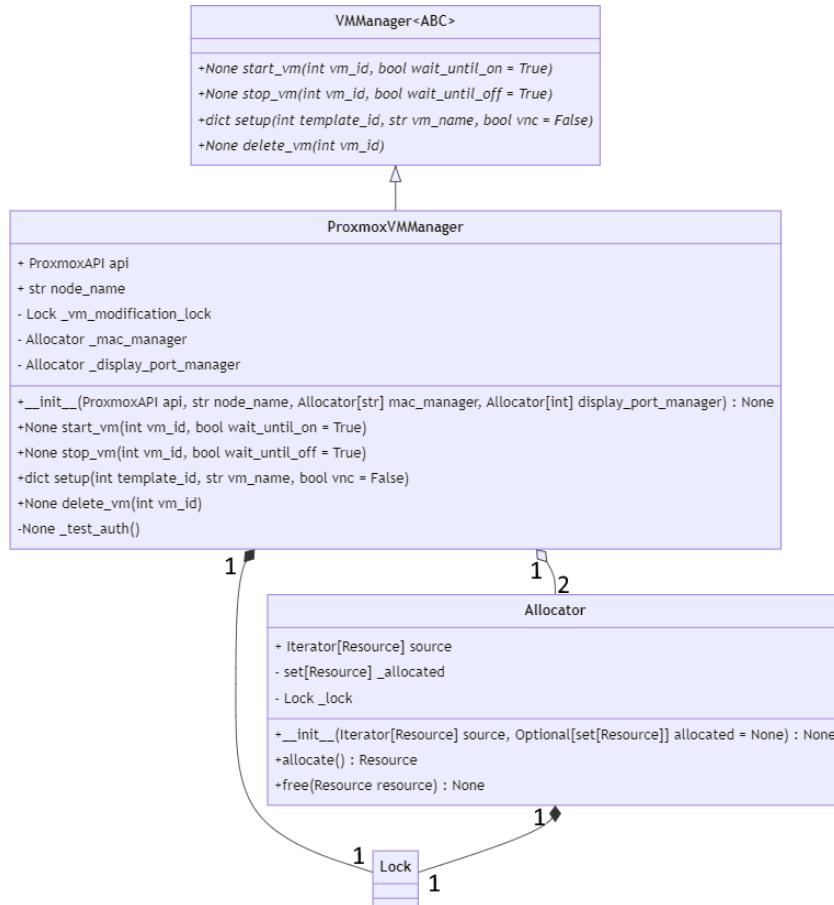


Figure 5.8.3: Diagramme de classe des gestionnaires de VMs

Ce gestionnaire de VM permet au serveur web de créer, allumer, éteindre et supprimer des machines virtuelles sur l'hyperviseur.

Le cas d'utilisation typique, où l'utilisateur crée une machine virtuelle depuis le site web afin de participer à une room, est représenté sur un diagramme de séquence sur la *Figure 5.8.4* ci-contre.

Ce diagramme représente à la fois la création d'une VM attaque et d'une VM victime, les deux cas étant assez similaires.

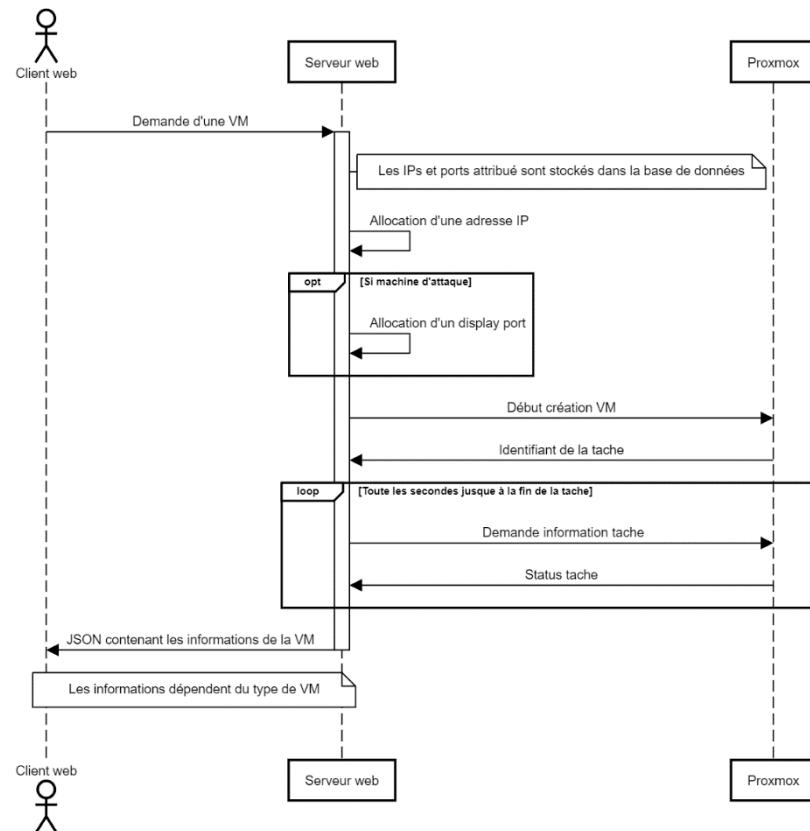


Figure 5.8.4: Demande d'une VM par un utilisateur

## 5.9 Accès distant aux machines virtuelles via VNC

Par souci de sécurité, les machines virtuelles doivent être complètement isolé du réseau externe. Cela peut poser problème, car les utilisateurs doivent pouvoir accéder aux machines virtuelles d'attaques afin de pouvoir attaquer les machines victimes.

Heureusement, Proxmox offre un accès aux machines VNC directement, et ce sans qu'elles n'aient besoin d'avoir un accès au réseau externe. Le protocole VNC est un protocole qui permet de contrôler une machine à distance graphiquement.

La connexion VNC est établi depuis le PC de l'utilisateur vers le serveur proxmox. Le port VNC détermine la machine à laquelle l'utilisateur est connecté.

Promox sert ensuite de passerelle entre la machine virtuelle et l'utilisateur, transmettant le contenu de l'écran, les entrées souris et clavier. Cela permet donc de faire exactement comme une connexion VNC classique, mais sans le risque d'exposer les machines sur le réseau externe.

Un client VNC externe au site est utilisé, car plus adapté à ce cas d'utilisation. Un client VNC intégré au site (avec NoVNC) a été considéré, mais plusieurs raisons nous ont fait favoriser un client externe :

- La page du site pour la room est déjà très chargé, et il n'y a pas de place pour y ajouter un écran entier.
- Un client VNC externe permet d'utiliser un deuxième écran pour la machine virtuelle, offrant un plus grand confort d'utilisation.
- Le serveur VNC offert par Proxmox ne permet pas de se connecter via WebSocket (ce qui est requis par NoVNC) sans un cookie de connexion venant du même domaine que l'hyperviseur Proxmox, ce qui ajoute la contrainte que le site web doit être sur le même domaine que l'hyperviseur Proxmox. De plus, ce même cookie permettrait à l'utilisateur d'accéder à l'hyperviseur, représentant une faille de sécurité.

## 5.10 Éditions des questions depuis le site

Il est important qu'un administrateur du site soit capable d'ajouter, de modifier et de supprimer des questions. Pour ce faire, nous avons opter pour une édition directement depuis la page de la room.

Si un utilisateur est administrateur, alors un bouton « Modifier » lui est proposé sur chaque question, lui permettant ainsi d'activer et désactiver le mode édition pour cette question. Cela lui permet de modifier l'énoncé, la réponse, et le nombre de points que vaut la question. Un bouton « Supprimer » est également présent.

Quel est le mot de passe correspondant au hash suivant (sha256):  
4cffb4ed84e2986f067c9e373ef87bf6d5eddc7866fb2cdd41eb48429743f50d. Utilisez le mode simple.

*2 points*

Réponse

**Valider**

**Modifier** **Supprimer**

Figure 5.10.1: Question en affichage normal

Quel est le mot de passe correspondant au hash suivant (sha256):  
 \*4cffb4ed84e2986f067c9e373ef87bf6d5eddc7866fb2cdd41eb48429743f50d\*. Utilisez le mode simple.

ludovic4000

**Enregistrer les modifications**    **Modifier**    **Supprimer**

Figure 5.10.2: Question en mode édition

La modification des questions se fait en utilisant l'API du site web. Le bouton « Supprimer » effectue un requête DELETE pour la question correspondante, tandis que le bouton « Enregister les modifications » effectue une requête POST.

Des messages d'erreurs sont également prévu en cas d'erreur des requêtes, afin que l'utilisateur sachent qu'un problème est survenu.

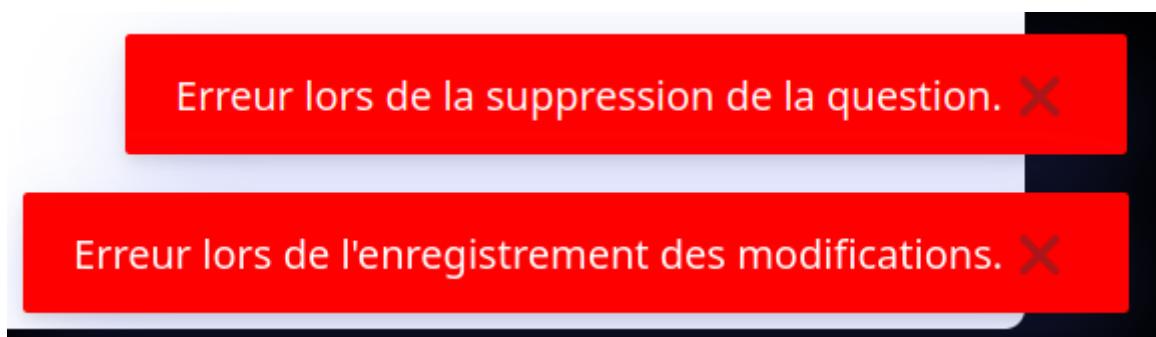


Figure 5.10.3: Messages d'erreurs affiché en haut à droite de la page

Ces messages d'erreurs utilisent la librairie **toastify-js**, et une fonction showErrorToast a été créée afin de garder un style uniforme pour tous les messages d'erreurs à travers le site.

## 5.11 Créations et affichage des VMs sur le site

Étant donné que les machines peuvent être créer et supprimé dynamiquement depuis le site web, afficher les informations des VMs directement depuis le template Jinja n'est pas une approche adaptée, car ne permettant pas de mettre à jour ces informations une fois la page chargée.

C'est pour cela que la technique de développement AJAX (Asynchronous JavaScript and XML) a été utilisé. Cette technique permet d'envoyer des requêtes au serveur en arrière-plan, de recevoir la réponse et de mettre à jour dynamiquement le contenu de la page sans recharger toute la page.

Pour cette approche, les boutons permettant de créer les machines virtuelles lancent une requête vers le serveur web, puis le client reçois une réponse JSON (JavaScript Object Notation) contenant les informations des machines virtuelles. Le code javascript se charge ensuite d'afficher ces informations sur la page.

Si l'utilisateur se rend sur la page d'une room en possédant déjà des VMs (car il a simplement rafraîchi la page par exemple), il faut que leurs informations soient affichées. Pour ce faire, la page effectue automatiquement une requête vers le serveur pour obtenir les informations sur les machines virtuelles existantes de l'utilisateur lors du chargement de la page. Elles sont ensuite affichées, ou s'il n'existe pas de machines virtuelles, les boutons offrant à l'utilisateur la possibilité d'en créer sont affichés.

Enfin, afin d'indiquer à l'utilisateur que la requête est en cours, le bouton de création de VM est désactivé et grisé une fois cliqué. Le bouton est ensuite complètement retiré une fois que la requête est terminée et que les informations de la VM sont affichées. Cependant, en cas d'erreur, un message d'erreur est affiché, et les boutons sont réactivés afin que l'utilisateur puisse réessayer.

## 5.12 Profil utilisateur

Afin de motiver les utilisateurs à toujours repoussé leurs limites, leur profile affiche leurs progressions ainsi que leur classement global.

De plus, une photo de profile de type « identicon » est attribué à chaque utilisateur afin d'avoir un moyen graphique de se différencier.



Figure 5.12.1: Le profil d'un utilisateur

Lorsque l'on résout une question, la date et l'heure à laquelle l'utilisateur a répondu est stocké dans la base de données. Il est donc possible de calculer le nombre de point qu'avait l'utilisateur à une date donnée, ce que fait la méthode `points_at_date` de la classe `utilisateur`.

Cela permet donc d'afficher un graphe de progression du nombre de point au court du temps. Ce graphe est réalisé à l'aide de la librairie python **pychartjs**, qui permet de créer un graphe à partir d'une classe python puis de le convertir en code HTML utilisable avec la librairie javascript **chart.js**.

Pour le classement, il est nécessaire d'obtenir les utilisateurs les plus proches de l'utilisateur en termes de score. Pour ce faire, la fonction générique `get_n_around` a été réalisée. Elle permet d'obtenir les n éléments les plus proches d'un index donné à partir d'une liste. Cette fonction doit fonctionner dans de nombreux cas différents, et ce fut donc un défi de prévoir toutes les possibilités. De nombreux cas de tests ont ainsi été réalisés pour s'assurer que cette fonction renvoie les bonnes valeurs dans tous les cas de figures.

```

def get_n_around(lst: list[T], index: int, amount: int) -> list[T]:
    """Renvois les n éléments les plus proches de l'index "index", incluant
    l'élément à la position index.

    Args:
        lst (list[T]): La liste contenant les éléments
        index (int): L'index servant de centre dans lst
        amount (int): Combien d'objets récupérer

    Raises:
        ValueError: amount est négatif, ou index n'est pas dans la liste

    Returns:
        list[T]: Une liste de "amount" éléments, ou de len(lst) si len(lst) < amount
    """

```

Figure 5.12.2: La signature de la fonction `get_n_around`

Le code entier de la fonction `get_n_around` est disponible en annexe 9.3. Les tests unitaires associés sont disponibles en annexe 9.4.

Enfin, pour l'identicon, il est généré automatiquement à partir du hash du nom de l'utilisateur, à l'aide de la librairie **customidenticon**. La méthode `get_profile_picture` de la classe `utilisateur` permet d'obtenir l'image au format PNG. Une route Flask permet quant à elle d'obtenir la photo de profile d'un utilisateur à partir de son nom.

### 5.13 Site web « responsive »

Afin que les utilisateurs puissent utiliser le site à partir de tout type de périphériques, par exemple avec un écran vertical ou un téléphone, le site est conçu pour s'adapter aux différentes tailles d'écrans. Cela est majoritairement réalisé à l'aide de **Bootstrap**, qui possède de nombreux éléments responsive par défaut, c'est-à-dire qui s'adapte dynamiquement à la taille de l'écran. Un effort a donc été fait afin d'utiliser uniquement des éléments responsive, et le site a été testé avec de nombreuses résolutions d'écrans.

Cependant, une page a nécessité une approche particulière afin d'être responsive. Il s'agit de la page des rooms.

The screenshot shows a web page titled 'Rooms'. At the top left, there is a progress bar labeled 'Progression: 0/6' and a red button 'Supprimer les VMs'. Below the progress bar, there is a section titled 'Consignes' with a blue background. It contains several questions:

- Quel est l'argument permettant d'identifier l'OS **et** la version?
- Quels sont les machines qui ont un serveur FTP appartenant au réseau **10.10.12.128/25**?
- Combien de services sont disponibles sur ce serveur?
- Quel est la version du serveur Web?
- Que trouvons-nous sur le port 21?
- Quel est le titre de la page web?

Each question has a 'Réponse' input field and a 'Valider' button. To the right of each question, there is a '2 points' label and a 'Modifier' or 'Supprimer' button. The overall layout is designed to be responsive, suitable for both desktop and mobile devices.

Figure 5.13.1: Page des rooms sur un écran d'ordinateur de taille standard

En effet, cette page est sectionnée en deux parties : la partie gauche avec la barre de progression et la liste des questions, et la partie droite avec les questions.

La liste des consignes n'est pas nécessaire, il n'est donc pas grave si l'écran est trop petit. Cependant, la barre de progression est importante, car elle contient le bouton permettant de supprimer les machines virtuelles.

Lorsque l'on manque d'espace horizontal, la partie gauche est automatiquement caché, car elle est contenue dans une div avec la classe **d-lg-block**. Cette classe, fourni par Bootstrap, n'affiche la div que sur un écran large.

Pour remplacer cette div caché, une autre div est placé avant les questions. Elle possède la classe **d-lg-none**, qui fait qu'elle ne s'affiche que sur les écrans plus petits, et donc elle n'est affichée que si la partie gauche est caché.

Cela permet de conserver toutes les fonctionnalités.

Un autre problème a été que le bouton « Supprimer les VMs » existait maintenant deux fois, et qu'il n'était donc plus possible de l'identifier avec un id unique. Pour pallier à ce problème, l'id a été remplacé par une classe, permettant au code javascript de cacher et d'afficher les boutons de suppression des VMs.

Progression: 0/6

**Supprimer les VMs**

Après avoir dressé et borné la prestation de tests d'intrusion, vous devez commencer par la première étape de votre mission : Enumération  
Pour mener à bien cette phase, nous allons apprendre à utiliser l'application nmap.

## NMAP

Nmap (« Network Mapper ») est un outil open source d'exploration réseau et d'audit de sécurité.  
Les commandes principales sont :

```
nmap -v <ip>
```

Cette option scanne tous les ports réservés TCP sur la machine.  
L'option -v active le mode verbeux.

```
nmap -ss -o <ip>/<CIDR>
```

Figure 5.13.2: Page des rooms sur un écran moins large

## 5.14 Interface en ligne de commande Flask

Flask offre une interface en ligne de commande (CLI) via la librairie **click**.

Cette interface offre par défaut quelques commandes, notamment une permettant de lancer le serveur.

Flask offre la possibilité d'ajouter des commandes personnalisées, ce qui permet aux administrateurs de facilement administrer le site. Cela offre également une certaine sécurité, car nécessitant un accès direct au serveur Flask.

C'est pour cette raison que nous avons opter pour l'interface CLI comme moyen d'ajouter un nouvel administrateur au site. Pour ce faire, nous avons créé une commande `make_admin`, qui prend en entrée un nom d'utilisateur.

Cette commande met simplement le booléen `is_admin` à True (vrai) pour cet utilisateur, et enregistre cette modification dans la base de données. Cette commande est visible sur la Figure 5.14.2 ci-dessous.

### Commands:

|                     |                                 |
|---------------------|---------------------------------|
| <code>routes</code> | Show the routes for the app.    |
| <code>run</code>    | Run a development server.       |
| <code>shell</code>  | Run a shell in the app context. |

Figure 5.14.1 : Commandes Flask par défaut

```

@app.cli.command("make-admin")
@click.argument("username")
def make_admin(username):
    """Donne les priviléges administrateur à un admin"""
    u = User.query.filter_by(username=username).first()
    if u is None:
        click.echo(f"L'utilisateur '{u.username}' n'existe pas", err=True)
    elif u.is_admin:
        click.echo(f"L'utilisateur '{u.username}' est déjà admin", err=True)
    else:
        u.is_admin = True
        db.session.commit()
        click.echo(f"L'utilisateur '{u.username}' est maintenant admin!")

```

Figure 5.14.2: Commande make-admin

L'administrateur du serveur peut donc ensuite utiliser la commande afin d'ajouter de nouveaux administrateurs au site, comme démontré sur la *Figure 5.14.3* ci-dessous.

```

(.venv) PS D:\LaurentM.SNIRW\Projet_BTS\ELYSIUM\site-web> flask --app main.py make-admin matthieu
L'utilisateur 'matthieu' est maintenant admin!
(.venv) PS D:\LaurentM.SNIRW\Projet_BTS\ELYSIUM\site-web> flask --app main.py make-admin matthieu
L'utilisateur 'matthieu' est déjà admin
(.venv) PS D:\LaurentM.SNIRW\Projet_BTS\ELYSIUM\site-web>

```

Figure 5.14.3: Ajout d'un nouvel administrateur au site

## 5.15 Langage Markdown

Sur le site, les administrateurs peuvent créer, modifier, et supprimer des rooms. Ils peuvent faire de même avec les questions.

Cependant, cela pose un problème. Il faut à la fois que les administrateurs puissent modifier **simplement** le contenu des questions, ainsi que de la description et des consignes des rooms, tout en permettant un **formatage** du texte (gras, italique, etc...).

Le texte doit également pouvoir inclure des **extrais de codes**, ainsi que des **tableaux**.

Enfin, ce texte doit pouvoir être stocké en tant que texte (dans une base de données), et doit pouvoir être convertis en HTML.

L'HTML, bien qu'offrant la plus grande flexibilité, ne permet pas une édition simple. En effet, il nécessite une connaissance des balises, et ce n'est pas simple pour quelqu'un de non-intuité d'en écrire.

L'alternative que nous avons choisie est le **markdown**. Markdown est un langage de balisage léger avec une syntaxe simple, un grand éventail de fonctionnalités, et pouvant être compilé vers de l'HTML.

Python permet de convertir du markdown vers de l'HTML à l'aide de la librairie éponyme **markdown**. La librairie **Pygment** est également requise, pour permettre la colorisation des blocks de code (*Syntax Highlighting*).

Un exemple de texte markdown et son rendu en HTML est présent sur la *Figure 5.15.1* ci-dessous.

```

# Titre 1
## Titre 2
Texte normal, italique, **gras**, ***italique+gras***, ==surligné==, ^superscrip^, _subscript_
Texte contenant echo "du code", et des notes en pied de page [^1]
...py
my_string = "Ceci est un block de code"
print("Il permet d'afficher du code sur plusieurs lignes")
def my_func(a: str) -> None:
    print("Et de coloriser le code")
...
| Colonne 1 | Colonne 2 |
|-----|-----|
| On peut même y ajouter | des tableaux |
| ou d'autres fonctionnalités | avec des plugins |
- des listes
[des liens](https://example.com)
> des citations
et même des emojis :smile:

Glossaire
:definition

[^1]: Note en pied de page

```

**Titre 1**

**Titre 2**

Texte normal, *italique*, **gras**, **italique+gras**, surligné, superscrip, subscript  
Texte contenant echo "du code", et des notes en pied de page [1](#)

```

my_string = "Ceci est un block de code"
print("Il permet d'afficher du code sur plusieurs lignes")
def my_func(a: str) -> None:
    print("Et de coloriser le code")

```

| Colonne 1                   | Colonne 2        |
|-----------------------------|------------------|
| On peut même y ajouter      | des tableaux     |
| ou d'autres fonctionnalités | avec des plugins |

- des listes
- [des liens](#)

des citations

et même des emojis 😊

Glossaire

definition

1. Note en pied de page [2](#)

Figure 5.15.1: Exemple de texte markdown montrant quelques fonctionnalités

Pour réaliser ce rendu en HTML, nous avons créé un **filtre Ninja**. Un filtre Ninja est une fonction qui permet de réaliser un traitement supplémentaire sur une variable depuis un template Ninja. De nombreux filtres sont présent par défaut, mais il est possible d'en ajouter des supplémentaires.

Tout d'abord, nous devons écrire une fonction prenant du texte markdown en entré et renvoyant de l'HTML. La fonction `markdown_filter` remplis ce rôle. Le code de cette fonction est présent en *annexe 9.9*.

Ensuite, ce filtre doit être enregistré dans l'application Flask. Cela est réalisé dans l'app factory.

```
# Register the filters
app.jinja_env.filters["markdown"] = markdown_filter
```

Figure 5.15.2: Enregistrement du filtre markdown

Nous avons enregistré ce filtre avec le nom « `markdown` », comme on peut le voir dans la *Figure 5.15.2* ci-dessus.

Enfin, il ne nous reste plus qu'à utiliser ce filtre dans nos templates Ninja. Un exemple d'utilisation est présent sur la *Figure 5.15.3* ci-contre, pour l'affichage des instructions de la room.

```
<div id="roomInfoDisplay">
    {{ room.instructions | markdown }}
</div>
```

Figure 5.15.3: Utilisation du filtre markdown

Un autre avantage d'avoir un format de balisage purement textuel est qu'il peut être inclus dans des fichiers textes. Cela nous permet de stocker toutes les informations d'une room dans un fichier **TOML**, ce qui nous permet de créer et d'importer des rooms facilement.

Bien que je me sois chargé de la réalisation pour l'import des fichier TOML, cela sera traité lors de la partie de Stefan INCE, car il s'est chargé de la création des rooms.

## 5.16 Conclusion

Ce projet m'a offert une opportunité précieuse de consolider mes connaissances en développement web avec Flask et Python, mais également en développement d'APIs, en web-design et en structuration de projet.

J'ai pu mettre en pratique les compétences acquises, approfondir mes connaissances techniques et relever de nombreux défis pour parvenir à des résultats concrets.

De plus, ce projet a été l'occasion de renforcer mes compétences en gestion de projet. J'ai dû planifier, organiser, coordonner les différentes étapes et prendre des décisions pour mener le projet à bien.

J'ai également dû m'assurer que la qualité du projet était au rendez-vous, en relisant chaque commit, et en aidant mes collaborateurs à réaliser leurs objectifs.

J'ai aussi pu mettre en pratique le frameworks SCRUM et la méthodologie DevOps, que j'avais appris lors de mon stage de première année. Ces outils nous ont permis de réaliser un développement bien plus rapide qu'une approche traditionnelle.

En conclusion, ce projet a été une véritable opportunité d'améliorer mes compétences dans de nombreux domaines à l'aide d'une réalisation concrète.

## 7 Réalisation - Stefen INCE

### 7.1.1 Présentation

Après avoir pris connaissance du cahier des charges, on constate que la réalisation du site est au cœur du projet. C'est cet élément qui sera en interaction continu avec l'utilisateur. Lui permettant d'avoir un fil conducteur que ce soit avec les instructions ou avec les questions pour les rooms. Le joueur peut accéder au classement général et comparer sa progression à celle des autres. Une page profil utilisateur propre à chaque joueur, aussi visible par les autres, liste les rooms lancées, affiche le nombre de points, la position et le nombre de challenges lancées. Cette dernière comporte également un autre classement et un graphique suivant l'évolution du joueur sur 7 jours.

L'administrateur à la possibilité se rendre sur l'interface administrateur du site. Répertoriant les informations des utilisateurs, des rooms avec leurs questions, leurs réponses et ses instructions.

Il n'est pas forcément question de n'utiliser le projet que pour réaliser des attaques, cela va au-delà de la cybersécurité.

### 7.1.2 Analyse préliminaire

Pour la structure du site et son contenu on utilise du HTML couplé à Jinja avec Flask (voir partie commune Choix des outils). Pour l'apparence ce sera du CSS « pur » par moments (classement, profil utilisateur, page de connexion, d'inscription) ou à l'aide du framework choisi.

Afin d'accélérer la vitesse de développement pour s'adonner aux tâches les plus complexes, nous avons décidé d'utiliser un framework CSS pour la partie front-end sur site web.

#### 7.1.2.1 Framework CSS

Parmi les 4 frameworks les plus populaires dont Skeleton, Foundation, Boostrap et Tailwind CSS.

##### 7.1.2.1.1 Avantage

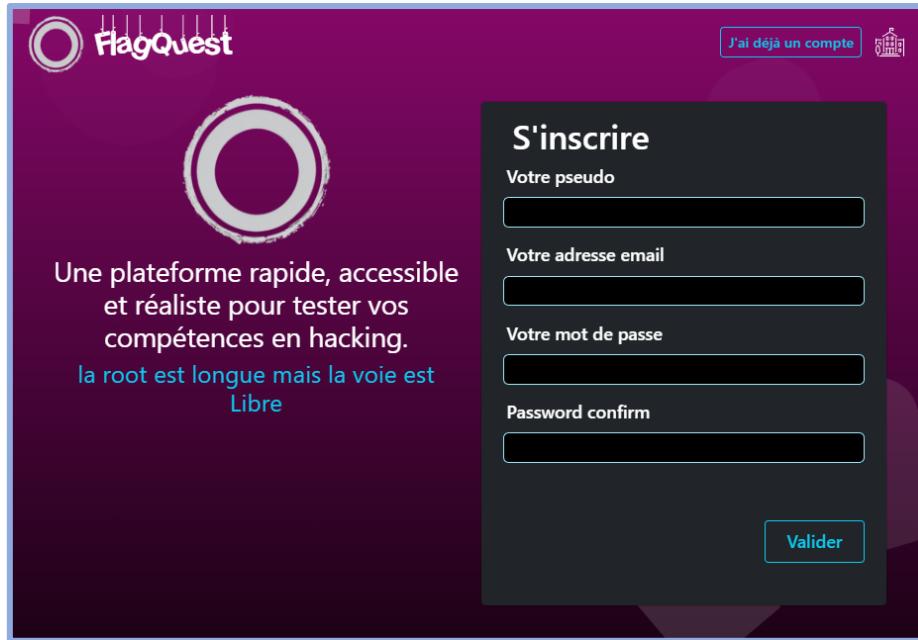
Nous avons choisi **Bootstrap** dans la mesure où nous l'avions tous déjà utilisé auparavant. Assurant une certaine rapidité et efficacité de développement comme nous n'avons pas à désigner de boutons et à définir chaque élément un à un. De plus, il possède une large communauté active avec des forums d'aide dédiés et une documentation complète truffé d'exemples concrets qui facilitent son utilisation et sa maintenance. De surcroît, c'est un framework libre et open source sous [licence MIT](#). La réutilisation du code instaure de ce fait peu de limites.

##### 7.1.2.1.2 Inconvénients

Bien que les thèmes proposés soient personnalisables et qu'on ait la possibilité d'en créer un, je trouve qu'il bride la créativité. Bon nombre de site fait avec ce framework se ressemblent et ce n'est pas un hasard avec les plus de 20 millions de sites qui s'en servent. Enfin, j'estime qu'on devient vite dépendant de sa structure.

### 7.1.3 Front-end

Dans la capture suivante, nous avons la page d'accueil du site qui permet de s'inscrire.



Aperçu du code HTML du formulaire, code complet en annexe XX :

```

<form method="POST" action="{{ url_for('main.inscription') }}">
  {% Token CSRF %}
  {{ signup_form.csrf_token }}

  <!-- NOM UTILISATEUR / PSEUDO -->
  <div class="mb-3 col-12">
    <label for="username">
      | class="form-label text-white fw-semibold ms-1 fs-5">{{signup_form.username.label}}</label>
      {{ signup_form.username(class_="input-login-username form-control text-white bg-black rounded-2 p-1 ps-3 border border-1 border-info-subtle") }}
    </div>

  <!-- Email -->
  <div class="mb-3 col-12">
    <label for="email">
      | class="form-label text-white fw-semibold ms-1 fs-5">{{signup_form.email.label}}</label>
      {{ signup_form.email(class_="input-login-username form-control text-white bg-black rounded-2 p-1 ps-3 border border-1 border-info-subtle") }}
    </div>

  <!-- PASSWORD -->
  <div class="mb-3 col-12">
    <label for="password">
      | class="form-label text-white fw-semibold ms-1 fs-5">{{signup_form.password.label}}</label>
      {{ signup_form.password(class_="input-login-password form-control text-white bg-black rounded-2 p-1 ps-3 border border-1 border-info-subtle") }}
    </div>

  <!-- PASSWORD CONFIRMATION -->
  <div class="mb-3 col-12">
    <label for="password_confirmation">
      | class="form-label text-white fw-semibold ms-1 fs-5">{{signup_form.password_confirmation.label}}</label>
      {{ signup_form.password_confirmation(class_="input-login-password form-control text-white bg-black rounded-2 p-1 ps-3 border border-1 border-info-subtle") }}
    </div>

  (# bouton valider #)
  <div class="d-grid gap-2 d-md-flex justify-content-md-end">
    {{ signup_form.submit(class_="btn-valider btn rounded-1 border-2 active text-info fs-5 fw-semibold px-4 py-2 mt-5") }}
  </div>
</form>
```

Par-ci, le formulaire de connexion pour les utilisateurs déjà inscrits.

Le code complet de cette page est joint en annexe.

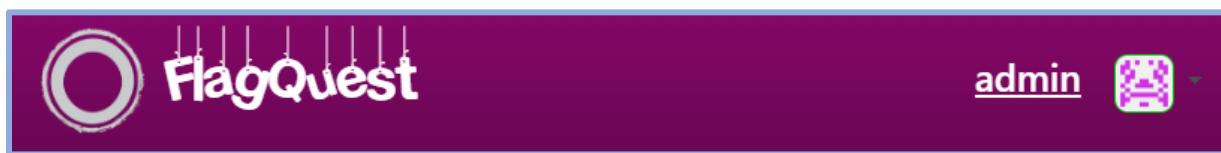
On a 2 headers, un lorsqu'on n'est pas connecté (voir entièreté du code annexe) :



Le code suivant correspond à la partie de gauche avec le bouton « J'ai déjà un compte » :

```
{% else %}
<a href="{{ url_for('main.connexion') }}"
    class="btn rounded-1 border-2 active text-info fw-semibold px-2 py-1" role="button"
    aria-pressed="true">J'ai déjà un compte</a>
<a class="a-rootme-pro" href="http://www.lyceebranly.com/">
    
</a>
{% endif %}
```

L'autre header, ci-dessous lorsqu'on est connecté, celui-ci possède un **menu défilant** ou **dropdown** qui affiche plus de possibilités pour un utilisateur administrateur :





Le footer est tout le temps le même, son code est disponible en annexe :



## Flashing

Le flashing permet à l'utilisateur de recevoir des commentaires sur les actions qu'il est en train de faire. J'ai utilisé Flask-flash pour le mettre en place. Il suffit de l'importer au haut du fichier.

```
from flask import flash
```

On entre le message que l'on souhaite voir affiché et la couleur est définie en fonction du second paramètre indiqué dans la fonction `flash()` grâce aux conditions de Jinja.

```
@main.route("/connexion", methods=["GET", "POST"])
def connexion():
    """Formulaire permettant la connexion."""
    form = LoginForm()
    # Si le formulaire a été correctement rempli
    if form.validate_on_submit():
        # On cherche l'utilisateur demandé
        user: User = User.query.filter_by(username=form.login.data).first()
        if user is None:
            form.login.errors.append("Cet utilisateur n'existe pas.")
            flash("Utilisateur inexistant", "error")

        # Vérification du hash du mot de passe
        elif not user.verify_password(form.password.data):
            form.password.errors.append("Mot de passe invalide.")

    # Pas d'erreur, on connecte l'utilisateur avec Flask-login (ajout des cookies de session)
    else:
        login_user(user)

        flash("Logged in successfully.", "success")

        next_url = request.args.get("next")
        # is_safe_url should check if the url is safe for redirects.
        # See http://flask.pocoo.org/snippets/62/ for an example.
        # if not is_safe_url(next):
        #     return flask.abort(400)

        # On redirige vers la page d'accueil
        return redirect(next_url or url_for("main.acceuil"))

    # Si on a pas submit une form valide, ou alors que l'on avait
    # un MDP invalide ou un login qui n'existait pas
    return render_template("connexion.jinja", login_form=form)
```



## Démonstration :

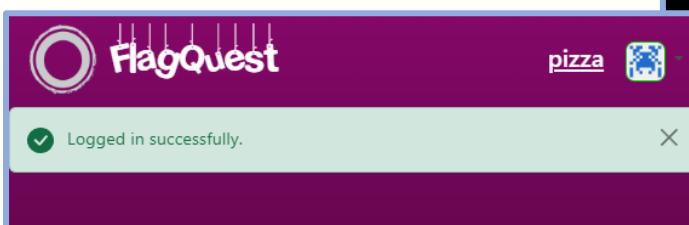


Figure 6.6.1: aperçu d'un message lorsque le joueur se connecte.

Selon la valeur de ce paramètre, on peut afficher 3 types de commentaires : les succès (en vert), les erreurs/avertissemens (en rouge) et les informations (en bleu).

```

<!-- ROUGE alert warning -->
{% with messages = get_flashed_messages(with_categories=true) %}
{% if messages %}
{% for category, message in messages %}
{# VERT - success #}
{% if category == 'success' %}


<svg class="bi flex-shrink-0 me-2" width="24" height="24" role="img" aria-label="Success:">
| <use xlink:href="#check-circle-fill" />
</svg>
{{ message }}
<button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>


{% endif %}
{# ROUGE - warning #}
{% if category == 'error' %}


<svg class="bi flex-shrink-0 me-2" width="24" height="24" role="img" aria-label="Danger:">
| <use xlink:href="#exclamation-triangle-fill" />
</svg>
{{ message }}
<button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>


{% endif %}
{# BLEU - info #}
{% if category == 'info' %}


<svg class="bi flex-shrink-0 me-2" width="24" height="24" role="img" aria-label="Info:">
| <use xlink:href="#info-fill" />
</svg>
{{ message }}
<button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>


{% endif %}
{% endfor %}
{% endif %}
{% endwith %}

```

Figure 6.6.2: définition du style pour chaque catégorie de message flash avec Jinja

Je fais apparaître ces animations interactives lors de la connexion et de la déconnexion au compte. Mais également si une personne essaie tente de se connecter à un compte non enregistré.

```

@main.route("/deconnexion")
@login_required
def deconnexion():
    """Déconnecte automatiquement les utilisateurs."""
    logout_user()
    flash("Vous avez été déconnecté.", "info")
    return redirect(url_for("main.acceuil"))

```



#### 7.1.4 Classement

Afin d'établir un palmarès selon l'avancement des points obtenus par les joueurs, j'ai mis en place un système de classement. Celui-ci se fait en fonction du nombre de points que possède chaque utilisateur sur la plateforme toutes rooms confondues.

Il est accessible par tous les utilisateurs. La personne connectée qui le consulte apparaît en rose afin qu'elle sache plus facilement où elle se positionne parmi les autres.

Nous avons la possibilité de cliquer sur les joueurs afin de voir leur profil (voir la partie traitant du profil utilisateur).

The screenshot shows the FlagQuest application's user ranking page. At the top, there is a navigation bar with the logo 'FlagQuest' and a user profile for 'admin'. Below the header, the title 'Classement' is displayed next to a trophy icon. A note indicates '6 utilisateurs'. The main content is a table listing the users and their scores:

| Position | Avatar | Utilisateur | Score |
|----------|--------|-------------|-------|
| # 1      |        | catherine   | 25    |
| # 2      |        | admin       | 19    |
| # 3      |        | sebastien   | 14    |
| # 4      |        | david       | 12    |
| # 5      |        | benoit      | 6     |
| # 6      |        | gregory     | 2     |

At the bottom of the page, there is a footer with links to 'Confidentialité', 'Mentions Légales', 'Conditions Générales d'Utilisation', the 'FlagQuest' platform description, and the copyright year '© 2023'. There are also social media icons for Twitter and LinkedIn.

Figure 6.6.3: classement des utilisateurs en fonction de leur score

Comme on dédie une page entière à l'affichage du classement. J'ai créé une nouvelle route. En Flask, le routage permet de définir des URLs plus simple à mémoriser et de les lier à une fonction.

Ici, au lieu d'avoir à entrer : <http://127.0.0.1:5000/classement.jinja>

L'utilisateur à juste à entrer : <http://127.0.0.1:5000/classement>

Cette URL est liée à la fonction `classement()` défini juste après avoir créé notre route.

Ladite fonction est plutôt simple à comprendre.

Elle crée une variable `user` dans laquelle on répertorie tous les joueurs contenus dans la base de données et en les triant en fonction de leur score dans l'ordre décroissant. Cette variable est liste.

La fonction `sorted()` permet de faire ce tri avec 3 paramètres :

1. `User.query.all()` – liste tous les joueurs.
2. `key=lambda u : u.score` – le paramètre « `key` » spécifie l'élément qu'on tri parmi les informations utilisateur listées, ici on tri en fonction du score des joueurs.
3. `reverse=True` – renvoie les éléments triés dans l'ordre décroissant.

Ensuite, je crée la variable « `nbr_user` » qui compte le nombre total de joueurs inscrit sur le site grâce à l'instruction « `User.query.count()` ».

Cette information n'a pas vraiment d'importance, je trouvais ça simplement intéressant de la mettre.

Enfin, on renvoie le modèle « `classement.jinja` » et les 2 variables créées précédemment afin que leurs valeurs soit affichées.

Le modèle « `classement.jinja` » se trouve dans le dossier « `template` » (voir la partie Base de données ed Matthieu Laurant)

```
@main.route("/classement")
def classement():
    """Un classement de tout les utilisateurs"""

    # tri par ordre score : décroissant
    user = sorted(User.query.all(), key=lambda u: u.score, reverse=True)

    # compte nombre total d'utilisateurs
    nbr_user = User.query.count()

    return render_template("classement.jinja", user=user, nbr_user=nbr_user)
```

Figure 6.6.4 Route du classement utilisateur issue du fichier `flagquest/route/views/user.py`

Une fois ceci fait, on peut réutiliser les variables renvoyées par la fonction dans notre fichier « `classement.jinja` » qui possède le contenu et la structure de la page HTML grâce à Jinja.

Ci-dessous, on observe que j'ai créé un tableau HTML avec la balise `<table>`. L'en-tête `<thead>` du tableau fait référence au nom de chaque colonne.

```

<!-- tableau classement -->





```

Ce n'est que dans la partie `<tbody>` que l'on utilise les variables `user` et `nbr_user`.

Grâce à Jinja, on peut utiliser des instructions `if` et des boucles `for` à un modèle. Ça offre un rendu conditionnel qui empêche de se répéter.

La condition `if` me permet de changer la couleur de la ligne ou non. La ligne du joueur connecté à son compte apparaît en rose lorsqu'il consulte le classement. Sinon, on affiche les utilisateurs en vert et leurs scores en bleu.

La position du joueur n'est pas stockée dans la base de données. Je me suis servi de l'index de la boucle `for` pour cela avec `{{ loop.index }}`.

Cette boucle affiche dans l'ordre déterminé plus tôt, chaque utilisateur et ses informations (photo de profil, nom d'utilisateur, score). Comme la variable renvoyée `user` est une liste la boucle `for` est parfaite pour cela.

```

<!-- corps du classement-->
<tbody>
    {% for user in user %}

        <!-- affichage de la ligne en rose si l'user qui consulte est connecté -->
        {% if user.id == current_user.id %}
            <tr class="active-row">
                <th># {{ loop.index }}</th>
                <td></td>
                <td class="username-classement-style">
                    <a href="{{url_for('main.profile', username=user.username)}}">{{user.username|e}}</a>
                </td>
                <td class="td-score">{{user.score|e}}</td>
            </tr>
        {% else %}

            <!-- les autres lignes sont affichées en vert et bleu -->
            <tr class="lambda-row">
                <th># {{ loop.index }}</th>
                <td>
                    
                </td>
                <td class="username-classement-style">
                    <a href="{{url_for('main.profile', username=user.username)}}">{{user.username|e}}</a>
                </td>
                <td class="td-score">{{user.score|e}}</td>
            </tr>
        {% endif %}
    {% endfor %}
</tbody>
</table>

```

## Profil utilisateur

Le profil utilisateur garantit au joueur de connaître sa position dans le classement son nombre de points et le nombre de challenges lancés.

À gauche, une liste des rooms auxquels le joueur a participé est présente, l'utilisateur peut aller les finir directement depuis son profil personnel.

À droite, on a de nouveau un classement puis un graphique représentant l'évolution des points du joueur sur 7 jours (nombre de points en ordonné et date en abscisse).

Pour tout cela, on créer une nouvelle route (voir le code en annexe).

Sur la figure suivante en dessous de Not Found (debut route), on voit que si je suis connecté avec le compte **stefen** et que je vais à l'URL : <http://127.0.0.1:5000/profile>.

Je serais renvoyé à l'URL : <http://127.0.0.1:5000/profile/stefen>.

En effet, si on veut accéder à l'URL de la page profil sans spécifier de nom d'utilisateur particulier. Alors on est renvoyé sur son propre profil à condition d'être connecté. Si on n'est pas connecté, on renvoie le code d'état **HTTP 401**. Ce code est utilisé pour un accès non autorisé.

Si un nom d'utilisateur existant est précisé dans l'URL, alors la variable `user` stockera les informations de l'utilisateur en question.

Sinon, un message d'erreur s'affichera avec un code d'état **HTTP 404** (voir figure Not found).



Figure 6.6.5: Erreur 404 lorsque lors d'un accès au profil d'un joueur non inscrit

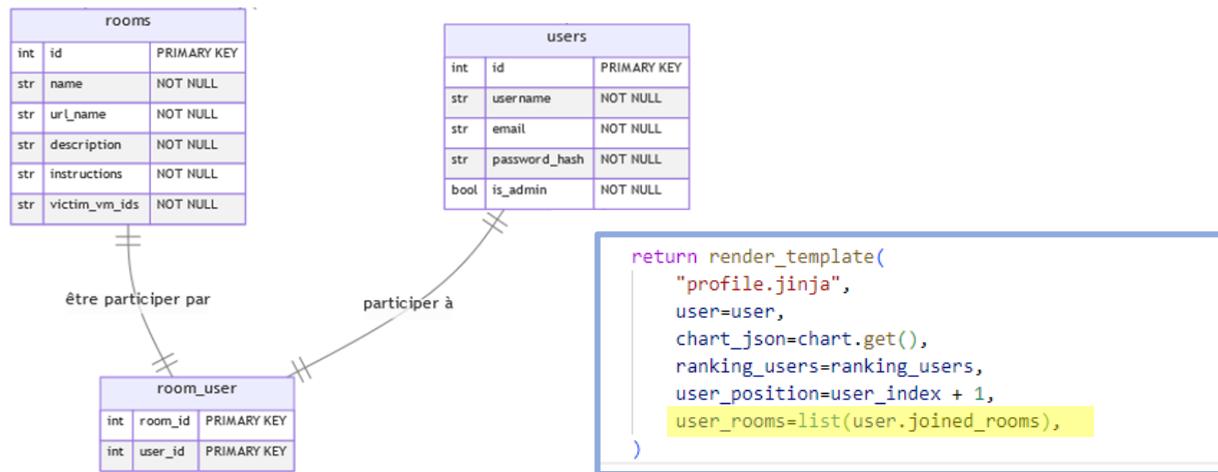
```
@main.route("/profile", defaults={"username": None})
@main.route("/profile/<username>")
def profile(username: str | None):
    """La page de profil de l'utilisateur"""

    # si on utilise l'URL : http://127.0.0.1:5000/profil
    # nom d'utilisateur pas spécifié
    if username is None:
        if current_user.is_authenticated:
            return redirect(url_for("main.profile", username=current_user.username))
        else:
            abort(401)
    else:
        user = User.query.filter_by(username=username).first_or_404(
            description="Cet utilisateur n'existe pas."
        )
```

Figure 6.6.6: première partie de la route profile

Le classement présent sur le profil utilisateur et le graphique ont été fait par Matthieu LAURENT, voir profile utilisateur de sa partie.

Lister les rooms auxquelles le joueur a participé se fait dans le `return` via la variable de type `list, user_rooms`. Cela se fait via la table de jointure `room_user` entre la table `rooms` et la table `users`.



Enfin, on affiche le nom de chaque rooms lancé avec une boucle `for` si l'utilisateur en a lancé. Sinon, on affiche le message « Qu'attendez-vous !? » :

```

    # liste des rooms lancées #
    <div>
        # si user_rooms n'est pas vide #
        {% if user_rooms %}
            # parcourir tout les rooms lancées #
            {% for r in user_rooms %}
                <a class="room-launched" href="{{url_for('main.room', room_url_name=r.url_name) }}">
                    <h3 class="room-title m-0">{{r.name}}</h3>
                    <div class="img-room">
                        
                    </div>
                    <div class="point-progression-room">
                        <span class="nbr-points-room">
                            {{ r.description | markdown | safe}}
                        </span>
                    </div>
                </a>
            {% endfor %}
            # si l'user n'a rejoint aucune room #
            {% else %}
                <p class="text-light fs-5 text-center">Qu'attendez-vous !?</p>
            {% endif %}
        </div>
    
```

Le résultat final ressemble à ceci :

**FlagQuest**

**admin** 🇫🇷

**Position** #1 **Points** 6 **Challenges** 2

**Rooms participées**

- Introduction à Nmap  
Apprenez les bases de Nmap, un outil d'exploration réseau.
- John The Ripper  
Apprenez à casser des mots de passe !

**Classement**

| Position | Avatar       | Utilisateur | Score |
|----------|--------------|-------------|-------|
| #1       | Admin Avatar | admin       | 6     |
| #2       | Pizza Avatar | pizza       | 0     |

Points

18/05/2023 19/05/2023 20/05/2023 21/05/2023 22/05/2023 23/05/2023 24/05/2023

Et à ça si l'utilisateur n'a participé à aucune room :

**FlagQuest**

**pizza** 🇫🇷

**Position** #2 **Points** 0 **Challenges** 0

**Rooms participées**

Qu'attendez-vous ?

**Classement**

| Position | Avatar       | Utilisateur | Score |
|----------|--------------|-------------|-------|
| #1       | Admin Avatar | admin       | 6     |
| #2       | Pizza Avatar | pizza       | 0     |

Points

18/05/2023 19/05/2023 20/05/2023 21/05/2023 22/05/2023 23/05/2023 24/05/2023

## Interface administrateur

Pour réaliser l'interface administrateur, j'ai utilisé **Flask-Admin**. Il facilite la tâche aux développeurs en évitant d'avoir à la créer à la main, en intégrant des fonctionnalités (CRUD) et des possibilités de personnalisation.



Pour cela j'utilise notre modèle de données déjà existant (voir partie Base de données de Matthieu LAURENT).

La capture qui suit initialise l'interface d'administration qui aura pour nom « **Interface Admin** » et qui supporte **Bootstrap 3**. L'`index_view` quant à lui renvoie à la classe `RestrictedAdminIndexView()` affiché dans une des figures suivante.

Cette ligne définit le thème à utiliser, il en existe une vingtaine sur : <https://bootswatch.com/3/>

```
app.config["FLASK_ADMIN_SWATCH"] = "cerulean" # thème couleurs clair
```

Lorsque les tables de la base de données sont disponibles, on les importe dans l'interface administrateur avec la méthode `add_view()`.

```
# flask-admin
from .classes import AdminModelView, RestrictedAdminIndexView

admin = Admin(
    name="Interface Admin",
    template_mode="bootstrap3",
    index_view=RestrictedAdminIndexView(),
)

app.config["FLASK_ADMIN_SWATCH"] = "cerulean" # thème couleurs clair
admin.init_app(app)

# création des tables
with app.app_context():
    from .models import Question, Room, SolvedQuestionData, User, VirtualMachine

    db.create_all()

# ajout des tables
admin.add_view(AdminModelView(User, db.session))
admin.add_view(AdminModelView(Room, db.session))
admin.add_view(AdminModelView(VirtualMachine, db.session))
admin.add_view(AdminModelView(Question, db.session))
admin.add_view(AdminModelView(SolvedQuestionData, db.session))
```

La classe `RestrictedAdminIndexView()` évoquée ultérieurement permet de limiter l'accès aux utilisateurs connectés avec un compte administrateur :

```
class RestrictedAdminIndexView(AdminIndexView):
    """
    Une vue d'index pour Flask-Admin.
    Accessible uniquement par les admins.
    """

    def is_accessible(self):
        """Vérifie que l'utilisateur a le droit d'accéder à la page"""
        return current_user.is_authenticated and current_user.is_admin
```

Lorsqu'on arrive sur la page du panel administrateur à l'URL : <http://127.0.0.1:5000/admin/>

Un message d'avertissement apparaît ainsi qu'un bouton retour redirigeant vers l'accueil du site au cas où quelqu'un réussirait à entrer (ce n'est pas censé être le cas) :

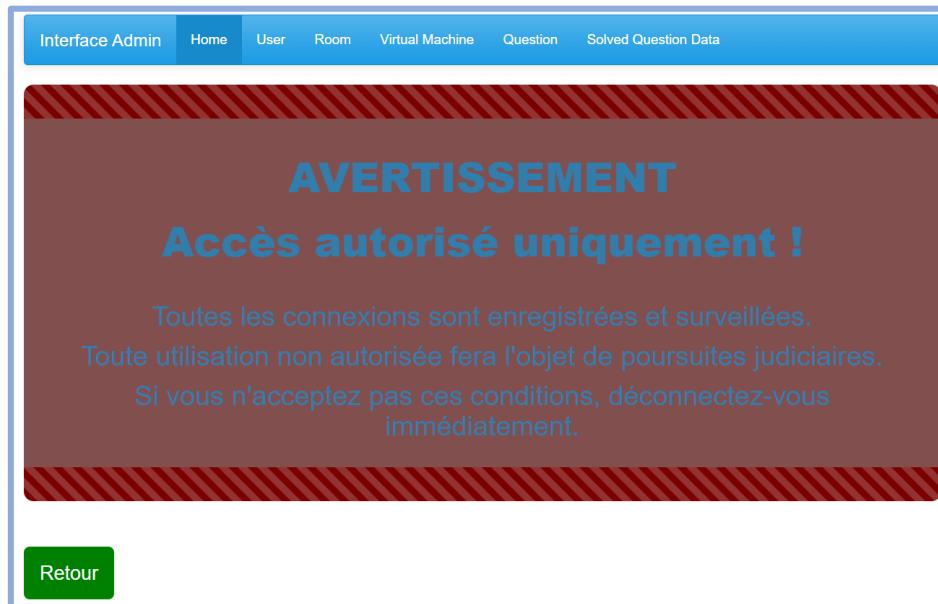


Figure 6.6.7: message d'avertissement à l'arrivée sur le panneau administrateur

Pour personnaliser la page **Home** du panel administrateur (vierge par défaut). J'ai créé un dossier **admin** dans le dossier **template** du site (voir le chapitre Structure du projet de Matthieu LAURENT) dans lequel j'ai créé le fichier « *classement.jinja* ».

Le code du fichier est le suivant :

```
{% extends 'admin/master.html' %}

{% block head %}
{{super()}}
<link rel="stylesheet" href="/static/style/auth.css">
{% endblock %}

{% block title %}
{{super()}} - Classement
{% endblock %}

{% block body %}
<div class="warning-container">
  <div class="warning-infos">
    <h2 class="warning-title">AVERTISSEMENT</h2>
    <h3 class="warning-title">Accès autorisé uniquement !</h3>
    <p></p>
    <h4 class="warning-content">Toutes les connexions sont enregistrées et surveillées.</h4>
    <h4 class="warning-content">Toute utilisation non autorisée fera l'objet de poursuites judiciaires.</h4>
    <h4 class="warning-content">Si vous n'acceptez pas ces conditions, déconnectez-vous immédiatement.</h4>
  </div>
</div>

<div class="button-container">
  <a href="/" class="button-retour">Retour</a>
</div>
{% endblock %}
```

Figure 6.6.8 : contenu du fichier flagquest/templates/admin/index.jinja

Sous la rubrique **User** (nom de la classe), on accède au contenu de la table **users**. Elle répertorie les joueurs inscrits sur la plateforme CTF.

| Interface Admin          |        |               |                   |   |               |                                     |
|--------------------------|--------|---------------|-------------------|---|---------------|-------------------------------------|
|                          | Home   | User          | Room              | Virtual Machine   | Question      | Solved Question Data                |
| List (5)                 | Create | With selected |                   |   |               |                                     |
|                          |        | Username      | Email             |   | Password Hash | Is Admin                            |
| <input type="checkbox"/> |        | admin         | feur@desu.wa      | pbkdf2:sha256:260000\$vbTIXTfM9CE1lOT3\$ee9716e060401bb20eeabf3fdb0904d6d0114a2a9f4b6e640d59814254e70974  |               | <input checked="" type="checkbox"/> |
| <input type="checkbox"/> |        | pizza         | pizza@mail.fr     | pbkdf2:sha256:260000\$4RRT26Blv6dtSR5a\$4fd3a136bcc38ecec794af57538558d1957b9870ec5f818c72b85eefaa78fa27e |               | <input checked="" type="checkbox"/> |
| <input type="checkbox"/> |        | leon          | leoon@mail.france | pbkdf2:sha256:260000\$5IJVmT5Tkgiiekc8B\$9393b16ab583124060054e50fa70b581da0045a211b60f14ccc9f6adad4e5d21 |               | <input checked="" type="checkbox"/> |
| <input type="checkbox"/> |        | annick        | annick@m.ck       | pbkdf2:sha256:260000\$vSm8XebjHApT45TS\$b6ca0a891a10a2ed664a6daed862548c7bdc4c43e58afb77e7e2a08856c1b9e5  |               | <input checked="" type="checkbox"/> |
| <input type="checkbox"/> |        | sarah         | sarah@m.fr        | pbkdf2:sha256:260000\$4ZtwkHZLveqHxz1I\$70402878af16ac501039c6e30e9667341ca4dcfc1c19091c06a345a2875307f   |               | <input checked="" type="checkbox"/> |

Même principe avec la rubrique **Room** faisant référence à la table **rooms**.

| Interface Admin          |        |                      |                   |  |   |                      |
|--------------------------|--------|----------------------|-------------------|--|---|----------------------|
|                          | Home   | User                 | Room              | Virtual Machine                                    | Question  | Solved Question Data |
| List (4)                 | Create | With selected        |                   |  |   |                      |
|                          |        | Name                 | Url Name          | Description  | Instructions  | Victim Vm Ids        |
| <input type="checkbox"/> |        | Introduction à Hydra | hydra             | Trouvez le mot de passe du serveur FTP!            | # Hydra [Hydra](https://github.com/vanhauser-thc/thc-hydra) est un logiciel libre permettant de craquer un mot de passe en ligne par **bruteforce**. ## Installation *Cet outil est déjà installé sur la VM d'attaque, cette explication est simplement à but éducatif.* ### Sous Linux On peut l'installer via **apt-get** ou **snap** : `` \$ sudo apt-get update \$ sudo apt-get install hydra -y `` ### Sous Windows Passez par [Cygwin](https://www.cygwin.com/) qui est une bibliothèque de logiciels libres permettant d'émuler un système Linux sous différentes versions de Windows. # Utilisation ## FTP Pour brute forcer un serveur FTP, on utilise la commande suivante : `` \$ hydra -l <username> -P </path/to/wordlist> ftp://<ftp-ip-adress> -V `` -l - spécifier le nom d'utilisateur du FTP -P - indiquer le chemin de la wordlist à utiliser -V - affiche le nom d'utilisateur et le mot de passe testés à chaque essai ## SSH Pour SSH, c'est légèrement différent : `` \$ hydra -l <username> -P </path/to/wordlist> <ssh-ip-adress> -t 4 ssh `` -l - spécifier le nom d'utilisateur -P - indiquer le chemin de la wordlist à utiliser -t - nombre de connexions en parallèle (16 par défaut) | 121                  |
| <input type="checkbox"/> |        | Introduction à Nmap  | introduction_nmap | Apprenez les bases de Nmap, un outil d'exploration | Après avoir dressé et borné la prestation de tests d'intrusion, vous devez commencer par la première étape de votre mission : Enumération Pour mener à bien cette phase, nous allons apprendre à utiliser l'application nmap. # NMAP Nmap (« Network Mapper ») est un outil open source d'exploration réseau et d'audit de sécurité. Les commandes principales sont : `nmap -v <ip>` Cette option scanne tous les ports réservés TCP sur la machine <ip>. L'option -v active le   | 121                  |

Même chose pour la table **questions** qui comporte les questions (**prompt**), leurs réponses (**answer**) et le nombre de points qu'elles rapportent (**points**).

| Interface Admin          |        |  |      |                 |                 |                      |
|--------------------------|--------|--|------|-----------------|-----------------|----------------------|
|                          | Home   | User   | Room | Virtual Machine | Question        | Solved Question Data |
| List (14)                | Create | With selected  |      |                 |                 |                      |
|                          |        | Prompt   |      |                 | Answer          | Points               |
| <input type="checkbox"/> |        | Quel était le mot de passe du serveur FTP  |      |                 | hannah          | 5                    |
| <input type="checkbox"/> |        | Avec quel outils graphique (très populaire) peut-on se connecter au serveur FTP?   |      |                 | filezilla       | 3                    |
| <input type="checkbox"/> |        | Donnez le flag caché dans le fichier 'flag.txt' sur le serveur   |      |                 | r4ti0-au_M@x    | 5                    |
| <input type="checkbox"/> |        | Quel est l'argument permettant d'identifier l'OS **et** la version?  |      |                 | -A              | 2                    |
| <input type="checkbox"/> |        | Quels sont les machines qui ont un serveur FTP appartenant au réseau **10.10.12.128/25***? Réponse attendu sous la forme |      |                 | 10.10.12.128/25 | 2                    |

Comme évoqué précédemment, **Flask-Admin** assure également de réaliser les opérations de base pour la persistance des données, les **opérations CRUD** :

- **Create** : créer
- **Read** : lire
- **Update** : mettre à jour
- **Delete** : supprimer

Avec cette interface administrateur je suis donc capable de créer, lire, mettre à jour et de supprimer des ressources présentes dans la base de données.

Ci-dessous, il est démontré que je peux changer l'`username` de « *sarah* » en « *sarah B.* » et passer son compte de joueuse simple en compte administrateur.

The screenshot shows the Flask-Admin User edit interface. At the top, there are two user rows in a list view:

- sarah**: Username: `sarah@m.fr`, Password Hash: `pbkdf2:sha256:260000$4ZtwkHZLveqHxz1...`
- sarah B.**: Username: `sarah@m.fr`, Password Hash: `pbkdf2:sha256:260000$4ZtwkHZLveqHxz1...`

A large green curved arrow points from the first row to the second row, indicating the transition from reading to updating. A red arrow points to the 'Edit' button in the top right corner of the modal window.

The modal window is titled 'Edit' and contains the following fields:

- Username \***: `sarah B.`
- Email \***: `sarah@m.fr`
- Password Hash \***: `pbkdf2:sha256:260000$4ZtwkHZLveqHxz1I$7040287f8af16ac50103f9c6e30e966``
- Is Admin**: A checked checkbox.

At the bottom of the modal are four buttons: **Save** (blue), **Save and Add Another**, **Save and Continue Editing**, and **Cancel** (red).

Créer ou supprimer des utilisateurs est aussi très simple, on les sélectionne et on clique sur l'icône de la poubelle si on veut les supprimer par exemple :

The screenshot shows the Flask-Admin User list interface. The table displays four users:

|                                     | Username | Email             | Password Hash   |
|-------------------------------------|----------|-------------------|---|
| <input type="checkbox"/>            | admin    | feur@desu.wa      | <code>pbkdf2:sha256:260000\$vbTIXTfM9CE1lOT3\$ee9716e060401bb...</code> |
| <input checked="" type="checkbox"/> | leon     | leoon@mail.france | <code>pbkdf2:sha256:260000\$5IJVmT5Tkgielc8B\$9393b16ab8531240</code>   |
| <input checked="" type="checkbox"/> | annick   | annick@m.ck       | <code>pbkdf2:sha256:260000\$vSm8XebjHApT45TS\$b6ca0a891a10a2e...</code> |
| <input type="checkbox"/>            | sarah B. | sarah@m.fr        | <code>pbkdf2:sha256:260000\$4ZtwkHZLveqHxz1I\$7040287f8af16ac50</code>  |

A modal dialog box is open over the table, asking: "Are you sure you want to delete this record?". It has two buttons: **OK** (blue) and **Annuler** (white).

Le résultat :

| Record was successfully deleted. |          |              |  |
|----------------------------------|----------|--------------|--|
|                                  | Username | Email        | Password Hash  |
| <input type="checkbox"/>         | admin    | feur@desu.wa | pbkdf2:sha256:260000\$vbTIXTfM9CE1IOT3\$ee9716e060401bb20ee  |
| <input type="checkbox"/>         | annick   | annick@m.ck  | pbkdf2:sha256:260000\$vSm8XebjHApT45TS\$b6ca0a891a10a2ed66   |
| <input type="checkbox"/>         | sarah B. | sarah@m.fr   | pbkdf2:sha256:260000\$4ZtwkHZLveqHxz1\$7040287f8af16ac50103f |

### 7.1.5 Crédation VM Attaque

Le cahier des charges souligne que l'on doit utiliser Kali Linux pour la machine virtuelle d'attaque. On a donc téléchargé l'ISO suivante : <https://cdimage.kali.org/kali-2023.1/kali-linux-2023.1-installer-amd64.iso> et on l'a importée dans l'hyperviseur.

1

Stockage 'local' sur nœud 'rootme'

Résumé      Upload      Download from URL      Supprimer

Nom: backbox-8-desktop-amd64.iso

Images ISO

2

Upload

Fichier:  Select File

File name:

File size: -

MIME type: -

Hash algorithm: None

Checksum: aucune

Abandonner      Upload

3

Ouvrir

Nouveau dossier

Organiser

Ce PC      Bureau      Documents      Images      Musique

Téléchargement

Stefen

plus tôt dans le mois (1)

Le mois dernier (2)

Plus tôt cette année (5)

Nom      Modifié le

backbox-8-desktop-amd64.iso      02/05/2023 14:14

ubuntu-22.04.2-live-server-amd64.iso      27/04/2023 09:59

https%3a%2f%2fmirrors.163.com%2fcyg...      06/04/2023 15:36

ubuntu-22.04.2-desktop-amd64.iso      16/03/2023 14:18

kali-linux-2022.4-installer-amd64.iso      26/01/2023 14:05

site-web-connecté      12/01/2023 14:54

mywebsite      12/01/2023 08:05

noVNC-1.3.0      11/01/2023 14:42

Fichier d'image di... 4 047 360 Ko

Fichier d'image di... 1 929 660 Ko

Dossier de fichiers

Fichier d'image di... 2 784 718 Ko

Fichier d'image di... 3 708 272 Ko

Dossier de fichiers

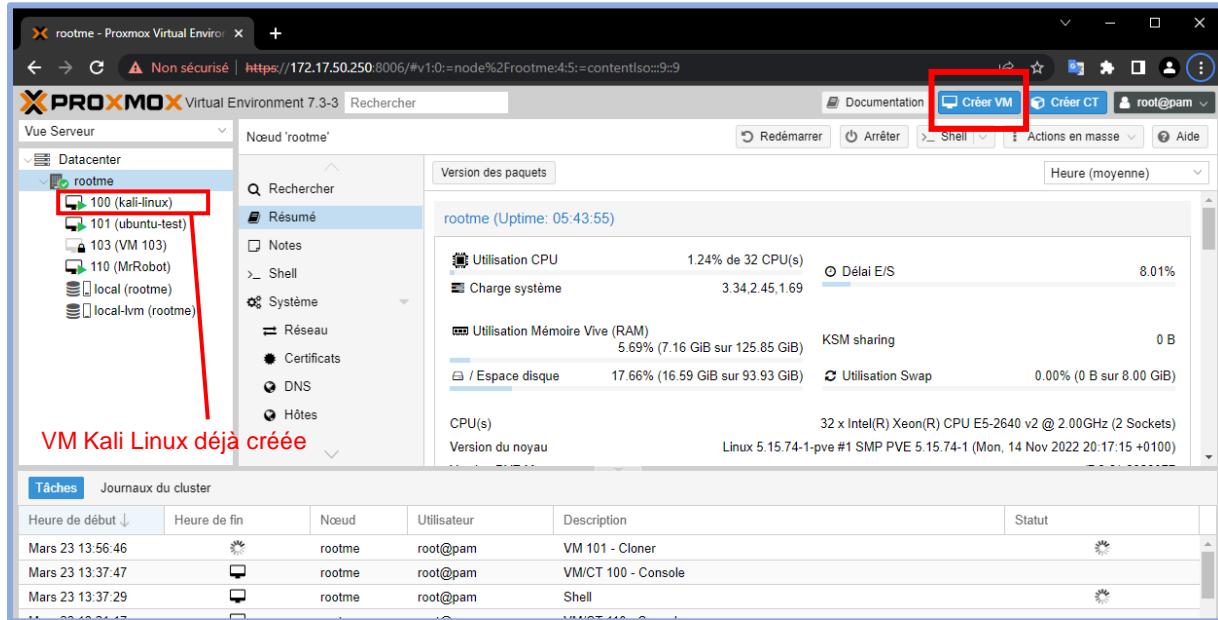
Dossier de fichiers

Nombre de fichier:  Fichiers personnalisés (\*.img; \*.iso)

Ouvrir      Annuler

Une fois l'ISO disponible dans l'hyperviseur. On créer notre VM avec les configurations système requise, pour Kali Linux il faut :

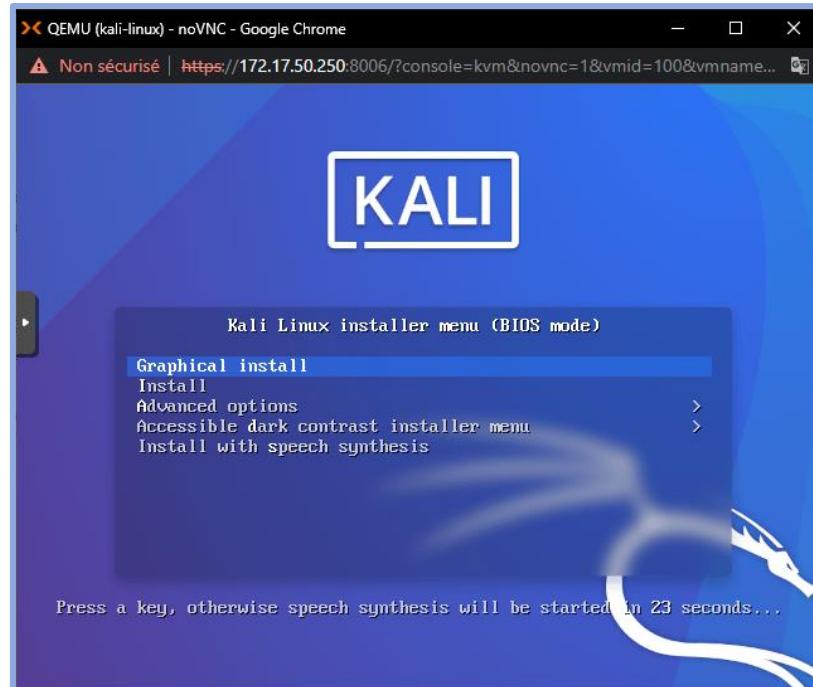
- 2 Go de RAM minimum
- 20 Go d'espace disque minimum



L'étape de création d'une VM est quasiment la même pour toutes à quelques différences près (prérequis système : RAM, espace disque, ...). Elle est détaillée voir partie Procédure de déploiement d'Adrien BRUAS.

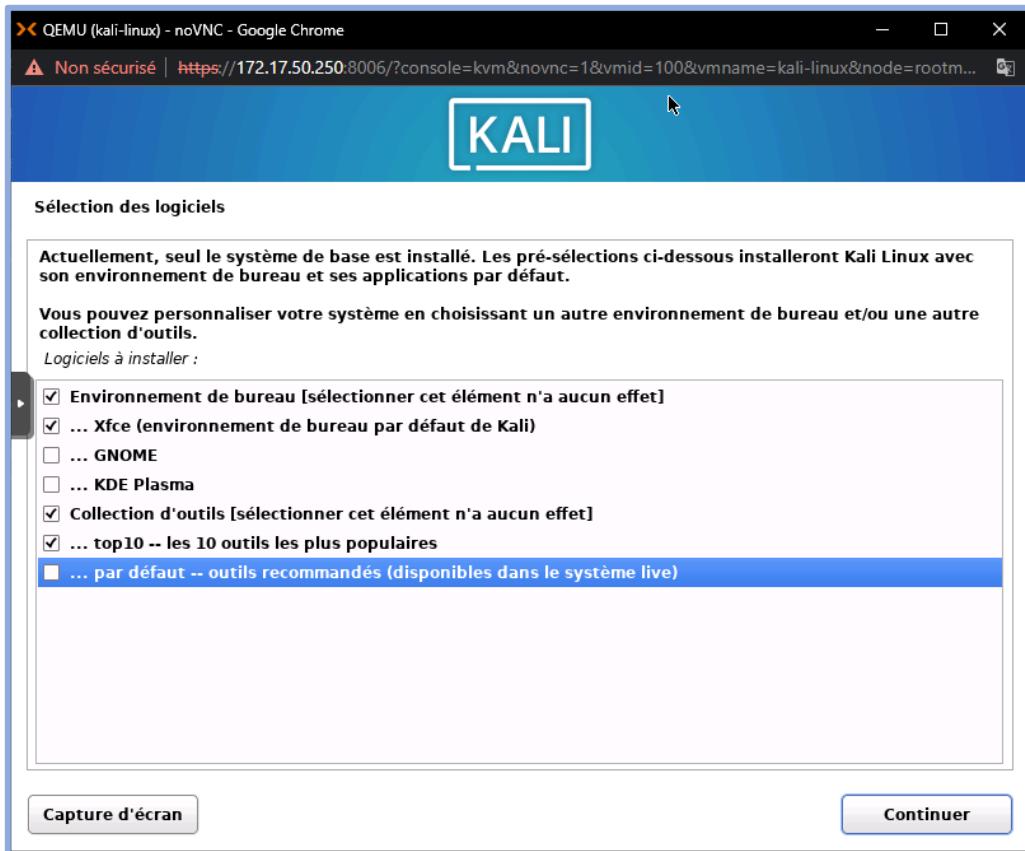
L'installation de Kali linux se fait de la manière la plus basique, on soit la procédure pour une installation graphique de la distribution.

Le nom de la machine sera **kali**, celui de l'utilisateur **hacker** et le mot de passe sera **kali**.



Cependant, on laisse la disposition du clavier en anglais (qwerty). **VNC Viewer** se chargera de convertir les frappes en azerty.

Aussi, pour que la VM soit la plus légère possible et qu'elle utilise le moins de ressources possible, nous avons décidé de n'installer que les 10 outils les plus populaires de la distribution (top 10).



Les 10 outils en question sont les suivants :

|             |           |                      |
|-------------|-----------|----------------------|
| aircrack-ng | burpsuite | crackmapexec         |
| hydra       | john      | metasploit-framework |
| nmap        | responder | sqlmap               |
| wireshark   |           |                      |

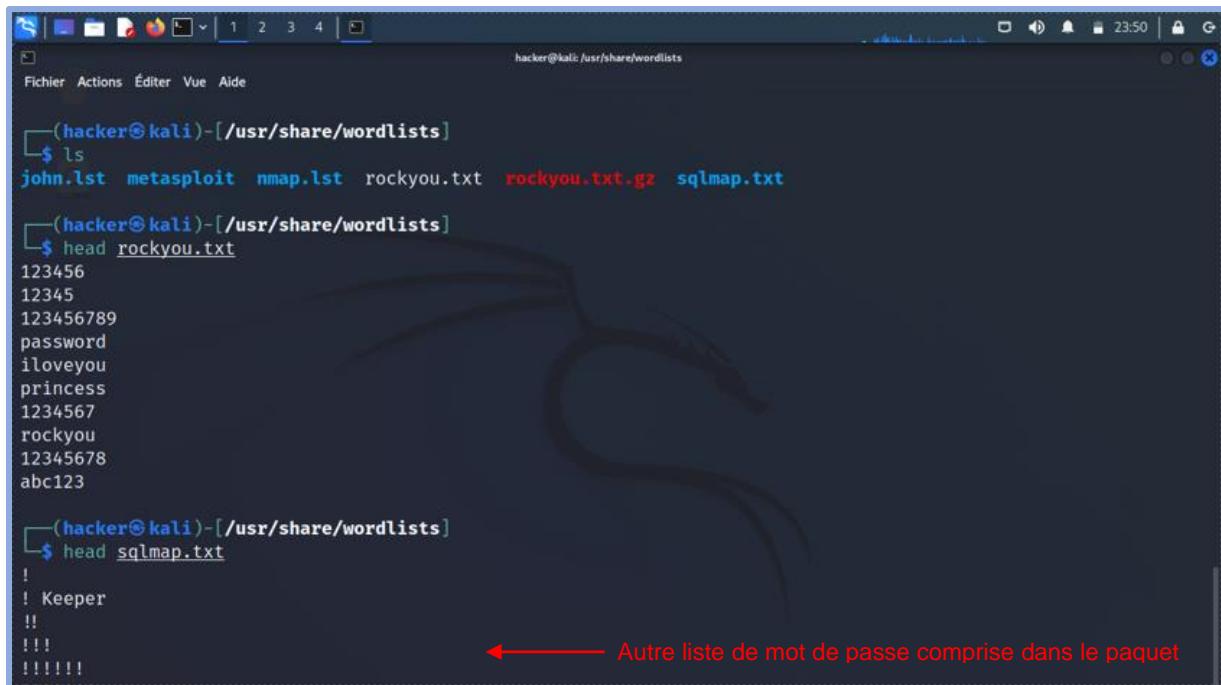
À ceux-là vient s'ajouter le paquet **wordlists**. Il contient notamment la liste de mots de passe **rockyou.txt** sous forme d'archive **Gzip** que j'ai extrait pour que les joueurs n'aient pas à le faire à chaque fois qu'ils克lonent la VM. Ce paquet contient également d'autres wordlists moins connues, toutes sont disponibles à l'emplacement **/usr/share/wordlists**.

Installation du paquet **wordlists** et extraction du fichier **rockyou.txt.gz** :

```
$ sudo apt-get update
$ sudo apt-get install wordlists -y
$ sudo gzip -d /usr/share/wordlists/rockyou.txt.gz
```



Ci-après, on observe que le fichier a bien été extrait au format `.txt`. On affiche les premières lignes pour voir ce que ça donne avec la commande `head` :



```
(hacker㉿kali)-[~/usr/share/wordlists]
$ ls
john.lst metasploit nmap.lst rockyou.txt  rockyou.txt.gz  sqlmap.txt

(hacker㉿kali)-[~/usr/share/wordlists]
$ head rockyou.txt
123456
12345
123456789
password
iloveyou
princess
1234567
rockyou
12345678
abc123

(hacker㉿kali)-[~/usr/share/wordlists]
$ head sqlmap.txt
!
! Keeper
!!
!!!
!!!!!!
```

← Autre liste de mot de passe comprise dans le paquet

Pour compléter les rooms, on n'utilisera principalement la wordlist `rockyou` car elle possède un nombre important de mots de passe (14 344 392), tous tirés d'une fuite de données du site RockYou en décembre 2009.

La commande `wc -l` permet de compter le nombre total de lignes présent dans un fichier :

```
$ wc -l /usr/share/wordlists/rockyou.txt
14344392 /usr/share/wordlists/rockyou.txt
```

Également, on a ajouté les packages `arp-scan` et `netdiscover`. Ils permettent l'envoie de requêtes ARP à des cibles et d'afficher les réponses afin de pouvoir analyser le réseau local par exemple. Puis le package `filezilla` le **client FTP** en interface graphique, certainement plus intuitif à utiliser pour les étudiants de l'année prochaine.

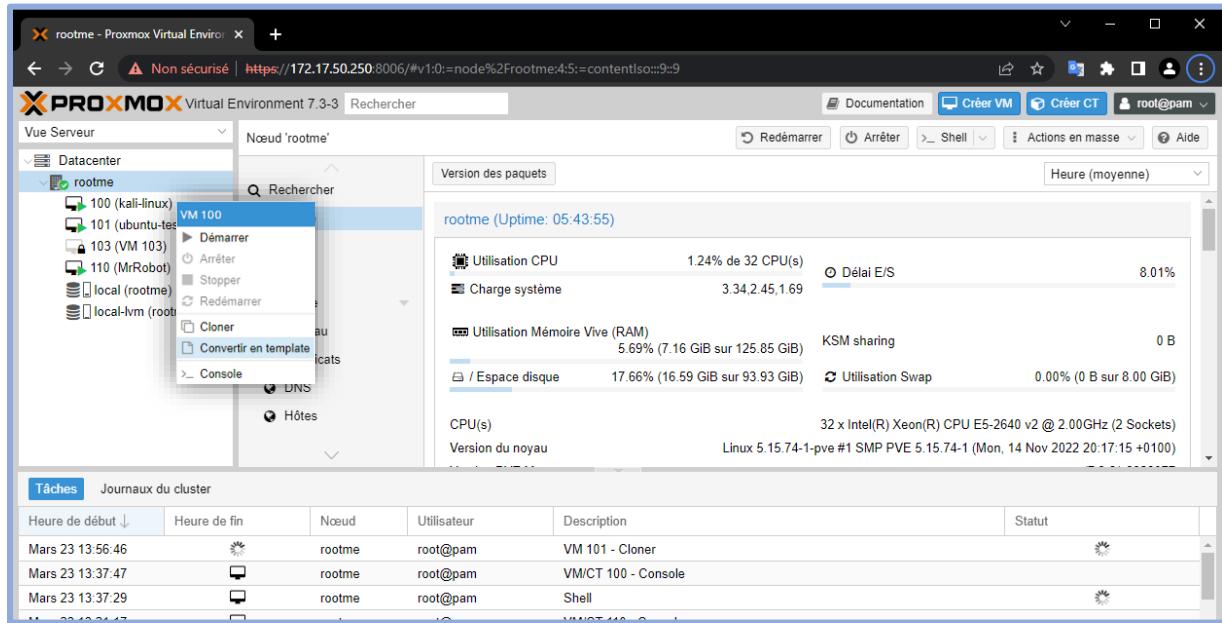
```
$ sudo apt-get install netdiscover arpscan filezilla -y
```



Pour finir, le professeur référant a demandé d'installer l'extension [Foxy Proxy Standard](#) pour une potentielle future room sur l'utilisation de `BurpSuite` :



Une fois la VM configurée et les outils additionnels téléchargés, je peux la convertir en template via Proxmox (elle doit être arrêtée avant) (se référer à la partie Procédure de déploiement d'Adrien BRUAS) :



### 7.1.6 Crédation des rooms

Les rooms disponibles respectent toutes la disposition schématisée :

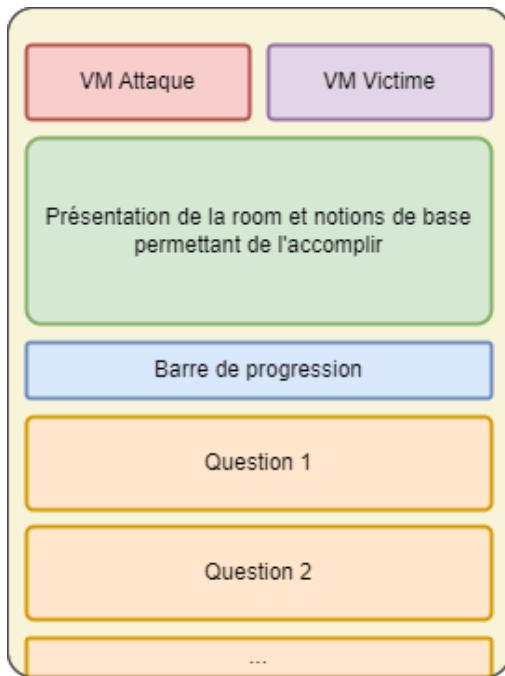


Figure 6.6.11: schématisation de la décomposition des différentes parties qui composent une room

#### VM attaque:

[Lancers la VM d'attaque](#)

Figure 6.6.9: bouton de création d'un clone de la VM d'attaque

#### VMs victimes:

[Lancers les VMs victimes](#)

Figure 6.6.10: bouton de création de clone(s) de(s) VM(s) victime

On a en haut de la page une partie VM Attaque et une partie VM Victime. Elles possèdent toutes deux un bouton permettant de les cloner à distance via le site.

Lorsque les VM seront clonés, ces parties renverront les informations les concernant.

La partie VM Attaque intègre un bouton « **Ouvrir avec RealVNC Viewer** » pour accéder à celle-ci.

Le clonage automatisé puis l'affichage des informations des VM a été réalisé par Matthieu, voir le chapitre Intéraction avec l'hyperviseur.

#### VM attaque:

IP Address: 172.17.50.250

VNC Port: 5901

Login: ubuntu

Mot de passe: ubuntu

[Ouvrir avec RealVNC Viewer](#)

#### VMs victimes:

10.0.0.2

S'ensuit la partie présentation de la room et des notions rudimentaires du/des outil(s) à utiliser pour répondre à toutes les questions.

Ci-dessous, un exemple avec la room **Hydra** :

## Hydra

[Hydra](#) est un logiciel libre permettant de craquer un mot de passe en ligne par **bruteforce**.

### Installation

*Cet outil est déjà installé sur la VM d'attaque, cette explication est simplement à but éducatif.*

### Sous Linux

On peut l'installer via **apt-get** ou **snap** :

```
$ sudo apt-get update
$ sudo apt-get install hydra -y
```

### Sous Windows

Passez par [Cygwin](#) qui est une bibliothèque de logiciels libres permettant d'émuler un système Linux sous différentes versions de Windows.

## Utilisation

### FTP

Pour brute forcer un serveur FTP, on utilise la commande suivante :

```
$ hydra -l <username> -P |/path/to/wordlist> ftp://<ftp-ip-adress> -V
```

-l - spécifier le nom d'utilisateur du FTP

-P - indiquer le chemin de la wordlist à utiliser

-V - affiche le nom d'utilisateur et le mot de passe testés à chaque essai

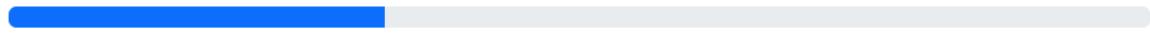
### SSH

Pour SSH, c'est légèrement différent :

```
hydra -l <username> -P |/path/to/wordlist> <ssh-ip-adress> -t 4 ssh
```

Une indication sur la progression de l'utilisateur dans la room est disponible. Elle évolue au fur et à mesure que l'utilisateur répond aux questions.

**Progression:** 1/3



Pour cela, la fonction `room()` vérifie que l'utilisateur soit connecté. Si c'est le cas, on calcule le nombre de questions auxquelles il a répondu dans la variable `nbr_question_solved`. Variable que la fonction retourne pour être utilisée avec Jinja dans le fichier « `room.jinja` » :

```
@main.route("/room/<room_url_name>")
def room(room_url_name: str):
    """Une room, avec affichage des questions"""
    from ...models import Room, VirtualMachine

    # cherche le nom la room entré dans l'url dans toute la bdd
    # si pas trouvé --> abandonne avec 404
    room: Room = Room.query.filter_by(url_name=room_url_name).first_or_404(
        description="Cette room n'existe pas."
    )

    # si l'user est connecté
    if current_user.is_authenticated:
        # calculer sa progression
        nbr_question_solved = sum(q.is_solved_by(current_user) for q in room.questions)

    return render_template(
        "room.jinja",
        room=room,
        nbr_question_solved=nbr_question_solved,
        user_existing_vms=user_existing_vms,
        user_attack_vm=user_attack_vm,
    )
```

Avec les variables `nbr_question_total` correspondant au nombre de questions que possède une room et `nbr_question_solved`, il devient possible de calculer la progression du joueur avec `progress_pourcentage`.

```
{% set nbr_question_total = room.questions | length %}
```

Figure 6.6.12: variable Jinja contenant le nombre de questions que la room possède

Côté front-end, on fait avancer la barre en fonction du nombre de questions répondues :

```
{% if nbr_question_total == 0 %}
{% set progress_percentage = 0 %}
{% else %}
{% set progress_percentage = (nbr_question_solved / nbr_question_total*100)|round|int %}
{% endif %}
<div class="bg-white rounded-3 p-4 justify-content-center border border-2 border-primary mb-0">
    <div class="row">
        <div class="col">
            <p><strong>Progression:</strong> {{nbr_question_solved}}/{{nbr_question_total}}</p>
            <div class="progress">
                <div class="progress-bar {{'bg-success progress-bar-striped progress-bar-animated' if nbr_question_solved == nbr_question_total else ''}}" aria-valuenow="{{nbr_question_solved}}" aria-valuemin="0" aria-valuemax="{{nbr_question_total}}"
                    style="width: {{ progress_percentage }}%">
                </div>
            </div>
        </div>
        <!-- button SUPPRIMER VM ---- call : deleteVms() -->
        <div class="col-md-auto delete-vms-button-container" style="display: none;">
            <button type="submit" name="submit_param" value="submit_value" onclick="deleteVms();">
                <span class="btn text-danger rounded-1 border-2 active fw-semibold m-2 align-self-end delete-vms-button"
                    role="button">Supprimer les VMS
            </span>
        </div>
    </div>
</div>
```

Enfin, nous avons les questions :

|  |                 |
|--|-----------------|
| Quel était le mot de passe du serveur FTP  | <i>5 points</i> |
| <input type="text" value="hannah"/>  |                 |
| <input type="button" value="Valider"/>   |                 |
| <hr/>  |                 |
| Avec quel outils graphique (très populaire) peut-on se connecter au serveur FTP? | <i>3 points</i> |
| <input type="text" value="Réponse"/>   |                 |
| <input type="button" value="Valider"/>   |                 |
| <hr/>  |                 |
| Donnez le flag caché dans le fichier <code>flag.txt</code> sur le serveur        | <i>5 points</i> |
| <input type="text" value="Réponse"/>   |                 |
| <input type="button" value="Valider"/>   |                 |

Afin de faciliter leur création, on crée un fichier en `.toml` pour chaque room. La fonction `add_room_from_toml()` s'occupe d'ajouter toutes les infos de la room, présentes dans le fichier, dans la base de données.

On utilise une syntaxe markdown pour gérer plus facilement les titres et sous-titres mais aussi l'affichage « stylisé » du code ou des lignes de commande lorsqu'il y en a dans la section **instructions**.

Avec cette syntaxe, pour afficher une ligne de commande par exemple on utilise le caractère « ` » ou « `` » si on a plusieurs lignes. Pour les titres « # », les sous-titres « ## », ...

La commande doit spécifier le paramètre `--single` ainsi que le format du hash :

```
john --single --format=<hash-format> <filename>
```

La commande doit spécifier le paramètre `--single` ainsi que le format du hash :

```
john --single --format=<hash-format> <filename>
```

Un fichier en `.toml` contient donc :

- le **titre** de la room
- sa **description**
- l'**URL** avec laquelle on accède à la room
- les **instructions** / l'objectif de la room avec une présentation de l'outil à utiliser
- l'**ID de la VM victime** à utiliser
- les **questions** (l'ordre a une importance)
  - intitulé de la question
  - la réponse attendue
  - le nombre de points que la question rapporte

Notons qu'il n'y a pas à spécifier l'ID pour la VM d'attaque dans le fichier étant donné que ce sera toujours la même pour toutes les rooms.

Voici un exemple :

```

name = "Introduction à Hydra"
description = "Trouvez le mot de passe du serveur FTP!"
url_name = "hydra"
instructions = """
# Hydra
[Hydra](https://github.com/vanhauer-thc/thc-hydra) est un logiciel libre
permettant de craquer un mot de passe en ligne par **bruteforce**.

## Installation

■ ■ ■
```
hydra -l <username> -P </path/to/wordlist> <ssh-ip-adress> -t 4 ssh
```
-l - spécifier le nom d'utilisateur
-P - indiquer le chemin de la wordlist à utiliser
-t - nombre de connexions en parallèle (16 par défaut)
```
victim_vm_ids = [121]

[[question]]
prompt = "Quel était le mot de passe du serveur FTP"
answer = "hannah"
points = 5

[[question]]
prompt = "Avec quel outils graphique (très populaire) peut-on se connecter au serveur FTP?"
answer = "filezilla"
points = 3

[[question]]
prompt = "Donnez le flag caché dans le fichier `flag.txt` sur le serveur"
answer = "r4ti0-au_M@x"
points = 5

```

Figure 6.6.13: fichier `hydra.toml` pour la création de la room Hydra

### 7.1.6.1 Room John The Ripper

La room John The Ripper n'utilise pas de machine virtuelle victime. J'ai jugé inutile de créer une VM qui ne servirait qu'à récupérer quelques fichiers. Par conséquent nous soumettons les hashs des mots de passe dans les questions.

Aussi, la valeur de la variable `victim_vm_ids` sera une liste vide, comme nous n'utilisons pas de machine virtuelle victime pour cette room.

```
Exemple :
`john --incremental=digits --format=raw-md
"""
victim_vm_ids = [ ]
```

[[question]]

Quel est le mot de passe correspondant au hash suivant (sha256):

2 points

`4cffb4ed84e2986f067c9e373ef87bf6d5eddc7866fb2cdd41eb48429743f50d`. Utilisez le mode simple.

Réponse

**Valider**

Modifier

Supprimer

La partie **instructions** de la room présente comment installer `john` sous Linux ou Windows, comment l'utiliser selon le type de hash que l'on a et le type d'attaque que l'on souhaite entreprendre.

L'utilisateur n'a alors qu'à créer un fichier texte via **vi**, **vim**, **nano** dans la machine virtuelle d'attaque Kali Linux, y insérer le hash en question et utiliser **John the Ripper** pour essayer de trouver ledit mot de passe.

Les mots de passe à trouver sont tous issues de la célèbre wordlist **rockyou** afin que l'utilisateur n'ai pas à attendre durant de longues heures avant de trouver la réponse.

Les hashs sont générés avec un terminal Linux de la manière suivante :

```
inces@SNIR-P146 ~
$ echo -n "francisco" | md5sum
117735823fadae51db091c7d63e60eb0 *-
```

La commande `echo` affiche une ligne de texte suivi par défaut d'un retour à la ligne. Nous ajoutons donc l'option `-n` pour ne pas afficher ce dernier.

Ce détail est important car il change la valeur du hash à cause du retour à la ligne.

```
inces@SNIR-P146 ~
$ echo "bonjour"
bonjour

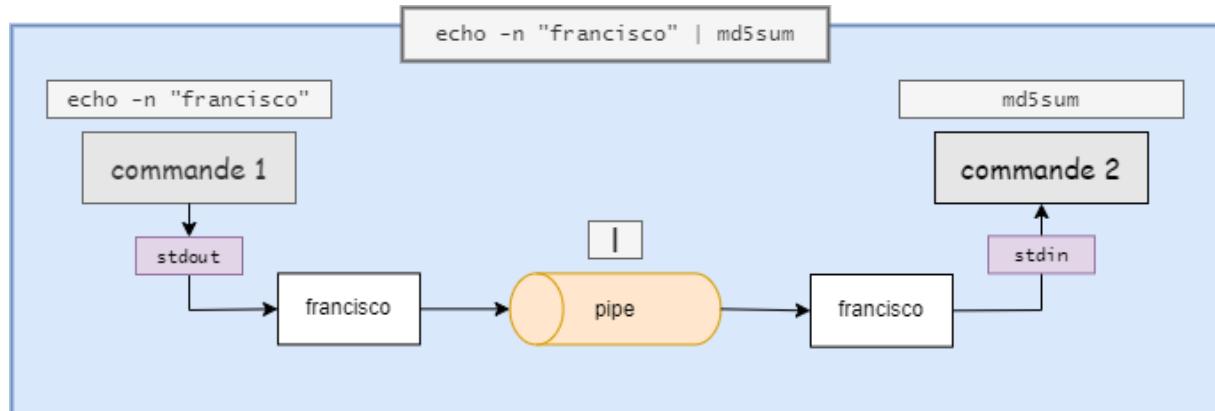
inces@SNIR-P146 ~
$ echo -n "bonjour"
bonjour
```

```
inces@SNIR-P146 ~
$ echo "francisco" | md5sum
5c9b1a945b58d625f8d9a8d2f2e57e1c *-
onces@SNIR-P146 ~
$ echo -n "francisco" | md5sum
117735823fadae51db091c7d63e60eb0 *-
```

La commande `md5sum` quant à elle permet d'obtenir la somme de contrôle de la chaîne de caractère envoyée à la commande `echo` grâce au `pipe`.

Le caractère « | » appelé `pipe`, se place entre les deux commandes. Il va injecter le résultat de la première dans la seconde.

Ci-après, une illustration du fonctionnement de la commande suivante :



Une fois le hash du mot de passe souhaité est généré, je n'ai plus qu'à l'insérer dans la question via le fichier `john.toml`.

```

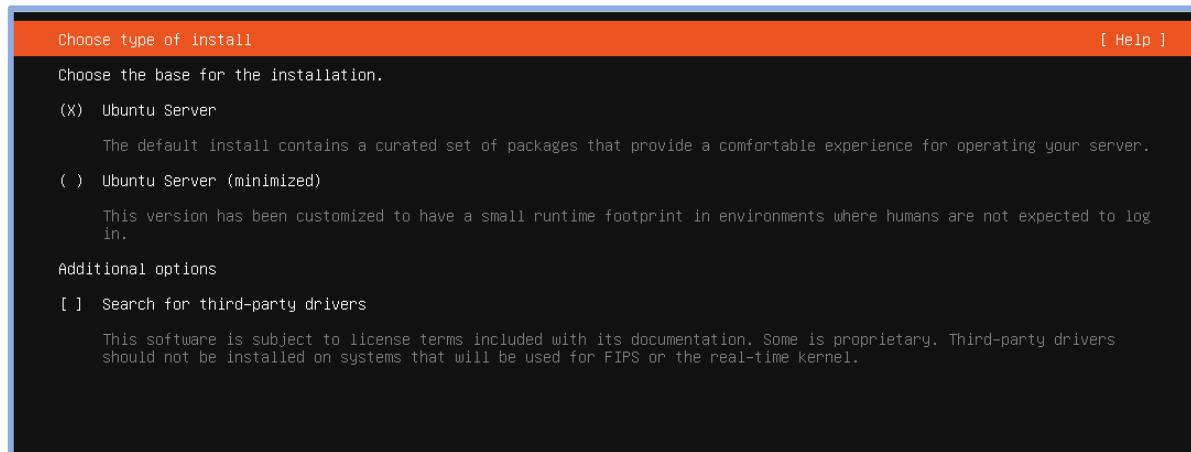
[[question]]
prompt = "Quel est le mot de passe correspondant au hash suivant (md5): *117735823fadae51db091c7d63e60eb0*. Utilisez l'attaque via la wordlist **rockyou**"
answer = "francisco"
points = 2
  
```

### 7.1.6.2 Room hydra

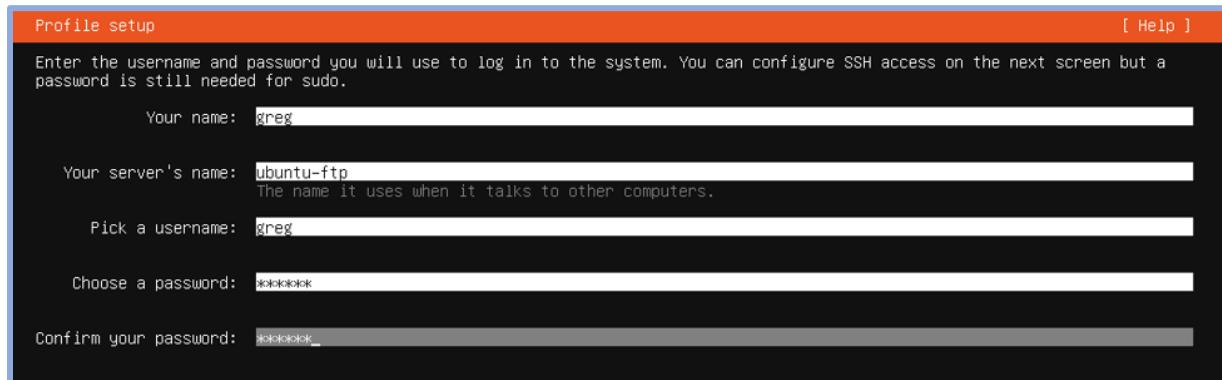
Concernant la room **hydra**, elle est composée d'un serveur FTP qui joue le rôle de la machine victime.

Pour le serveur, j'ai d'abord créé une VM [Ubuntu Server](#). Comme tout est en ligne de commande, c'est déjà plus léger que la VM Kali Linux.

On installe la version de base, le clavier peut être mis en disposition azerty étant donné que l'on ne passe pas par VNC Viewer mais par la console **noVNC** que met Proxmox à disposition.



On définit un utilisateur, ici j'ai mis **greg** pour le nom d'utilisateur, **ubuntu-ftp** pour le nom du serveur puis un mot de passe.



#### 7.1.6.3 Déploiement du serveur FTP

Une fois connecté à la session utilisateur d'Ubuntu Server, je commence par le mettre à jour.

Pour ça, je j'utilise la commande suivante :

```
greg@ubuntu-ftp:~$ sudo apt-get update && sudo apt-get upgrade -y
```

|                           |                                                                                                                                                                             |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>sudo</b>               | Utilisé pour exécuter la commande en mode super utilisateur (SuperUser DO = sudo).                                                                                          |
| <b>apt-get update</b>     | Télécharge les listes de paquets à partir des référentiels et les « met à jour » pour obtenir des informations sur les dernières versions des paquets et leurs dépendances. |
| <b>apt-get upgrade -y</b> | Récupère les nouvelles versions des paquets existant sur la machine.<br>APT les connaît parce qu'on a fait <b>apt-get update</b> juste avant.                               |
| <b>&amp;&amp;</b>         | Permet d'enchaîner les commandes, la seconde commande ne sera exécutée que si la première se passe bien.                                                                    |

```
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

greg@ubuntu-ftp:~$ whoami
greg
greg@ubuntu-ftp:~$ sudo apt-get update ←
[sudo] password for greg:
Atteint :1 http://fr.archive.ubuntu.com/ubuntu jammy InRelease
Atteint :2 http://fr.archive.ubuntu.com/ubuntu jammy-updates InRelease
Atteint :3 http://fr.archive.ubuntu.com/ubuntu jammy-backports InRelease
Atteint :4 http://fr.archive.ubuntu.com/ubuntu jammy-security InRelease
Réception de :5 http://fr.archive.ubuntu.com/ubuntu jammy/main Translation-fr [486 kB]
Réception de :6 http://fr.archive.ubuntu.com/ubuntu jammy/restricted Translation-fr [4 760 B]
Réception de :7 http://fr.archive.ubuntu.com/ubuntu jammy/universe Translation-fr [3 564 kB]
Réception de :8 http://fr.archive.ubuntu.com/ubuntu jammy/multiverse Translation-fr [93,5 kB]
4 149 Ko réceptionnés en 6s (690 Ko/s)
Reading package lists... Done
greg@ubuntu-ftp:~$ sudo apt-get upgrade -y ←
```

Figure 6.6.14: mise à jour d'Ubuntu Server via la ligne de commande

Une fois que tous les paquets sont à jour, je peux installer **VsFTPD** (Very Secure FTP Daemon). Il a été conçu pour avoir une sécurité maximale et personne n'a encore trouvé de faille majeure de sécurité contrairement aux autres serveurs FTP comme ProFTPD, PureFTPD, etc. Pour le coup, notre objectif est de le rendre vulnérable...

```
greg@ubuntu-ftp:~$ sudo apt-get install vsftpd -y
```

```
greg@ubuntu-ftp:~$ sudo apt-get install vim vsftpd nano -y
Lecture des listes de paquets... Fait
Construktion de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
nano est déjà la version la plus récente (6.2-1).
nano passé en « installé manuellement ».
vim est déjà la version la plus récente (2:8.2.3995-1ubuntu2.7).
vim passé en « installé manuellement ».
Les paquets supplémentaires suivants seront installés :
  ssl-cert
Les NOUVEAUX paquets suivants seront installés :
  ssl-cert vsftpd
0 mis à jour, 2 nouvellement installés, 0 à enlever et 1 non mis à jour.
Il est nécessaire de prendre 140 Ko dans les archives.
Après cette opération, 391 Ko d'espace disque supplémentaires seront utilisés.
Réception de :1 http://archive.ubuntu.com/ubuntu jammy/main amd64 ssl-cert all 1.1.2 [17,4 kB]
Réception de :2 http://fr.archive.ubuntu.com/ubuntu jammy/main amd64 vsftpd amd64 3.0.5-0ubuntu1 [123 kB]
140 Ko téléchargement en 1s (212 Ko/s)
```

Pour configurer VsFTPD, je commence par créer un nouvel utilisateur **userftp** qui utilisera l'accès FTP.

On lui définit aussi un mot de passe qui sera **hannah**. Ce mot de passe est présent au début de la wordlist **rockyou**, hydra n'aura donc pas à tester des centaines voire des milliers de mots de passe à essayer avant d'avoir accès au FTP. C'est aussi une question de confort pour le l'étudiant qui n'aura pas à attendre des heures devant son écran.

```
greg@ubuntu-ftp:~$ sudo adduser userftp
```

```
Adding user `userftp' ...
Adding new group `userftp' (1001) ...
Adding new user `userftp' (1001) with group `userftp (1001)' ...
Creating home directory `/home/userftp' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for userftp
Enter the new value, or press ENTER for the default
  Full Name []: userftp
  Room Number []:
  Work Phone []:
  Home Phone []:
  Other []:
Is the information correct? [Y/n] Y
Adding new user `userftp' to supplemental / extra groups `users' ...
Adding user `userftp' to group `users' ...
```

On redémarre le service :

```
greg@ubuntu-ftp:~$ sudo systemctl restart vsftpd
```

Après, je créer un dossier FTP dans le répertoire courant du nouvel utilisateur. Il contiendra les fichiers que l'utilisateur souhaite voir pour trouver le drapeau (aussi appelé flag).

```
greg@ubuntu-ftp:~$sudo mkdir /home/userftp/ftp
```

Bien que chaque joueur ait son propre environnement virtuel. Le participant ne doit pas pouvoir changer le contenu des fichiers présents dans le serveur FTP.

On attribue au dossier le propriétaire **nobody** et le groupe **nogroup** à l'aide de la commande **chown**.

Elle donne la possibilité de changer le propriétaire d'un fichier/dossier et le ou les groupes auxquels ce fichier est associé. Elle nécessite droit de l'utilisateur root.

Avec Linux, l'utilisateur **nobody** est un *pseudo utilisateur*. C'est une sorte de raccourci qui représente un utilisateur avec le moins d'autorisations sur la machine. Le terme **nogroup** utilise le même principe. Le dossier suivant ne sera dans aucun autre groupe que **nogroup**.

```
greg@ubuntu-ftp:~$sudo chown nobody:nogroup /home/userftp/ftp
```

La commande **chmod** permet de modifier les droits et permission sur un fichier ou un dossier. Le paramètre **a-w** supprime tous les droits en écriture du dossier **ftp** :

```
greg@ubuntu-ftp:~$sudo chmod a-w /home/userftp/ftp
```

Vérification en orange le résultat de la commande **chmod**, en vert le résultat de la commande **chown**, avant :

|                   |   |             |             |                   |            |
|-------------------|---|-------------|-------------|-------------------|------------|
| <b>drwxr-xr-x</b> | 2 | <b>root</b> | <b>root</b> | 4096 May 24 20:39 | <b>ftp</b> |
|-------------------|---|-------------|-------------|-------------------|------------|

Après :

|                   |   |               |                |                   |            |
|-------------------|---|---------------|----------------|-------------------|------------|
| <b>dr-xr-xr-x</b> | 2 | <b>nobody</b> | <b>nogroup</b> | 4096 May 24 20:39 | <b>ftp</b> |
|-------------------|---|---------------|----------------|-------------------|------------|

Je peux créer un dossier **files** dans le dossier **ftp** créé plus tôt et y mettre le contenu que je souhaite (fichiers texte, images, vidéos, ...) avec le drapeau à trouver bien sûr :

```
greg@ubuntu-ftp:~$ sudo mkdir /home/userftp/ftp/files
```

Comme ce dossier a été créé avec l'utilisateur **greg** en **sudo**, donc en **root**, il appartiendra à l'utilisateur et au groupe **root**. Je fais en sorte que le dossier appartienne à l'utilisateur **userftp** de nouveau avec la commande **chown** :

```
greg@ubuntu-ftp:~$ sudo chown userftp:userftp /home/userftp/ftp/files
```

Vérification :

|                   |   |             |             |                   |              |
|-------------------|---|-------------|-------------|-------------------|--------------|
| <b>drwxr-xr-x</b> | 2 | <b>root</b> | <b>root</b> | 4096 May 24 21:17 | <b>files</b> |
|-------------------|---|-------------|-------------|-------------------|--------------|

|                   |   |                |                |                   |              |
|-------------------|---|----------------|----------------|-------------------|--------------|
| <b>drwxr-xr-x</b> | 2 | <b>userftp</b> | <b>userftp</b> | 4096 May 24 21:17 | <b>files</b> |
|-------------------|---|----------------|----------------|-------------------|--------------|

On crée des fichiers vides dans le dossier **files** pour qu'il ait un peu de contenu :

```
greg@ubuntu-ftp:~$ sudo touch /home/userftp/ftp/files/voiture.png
greg@ubuntu-ftp:~$ sudo touch /home/userftp/ftp/files/souris.txt
greg@ubuntu-ftp:~$ sudo touch /home/userftp/ftp/files/chaise.mp4
```

Dans un dernier fichier on met le drapeau que les utilisateurs doivent trouver :

```
$ echo -n "r4ti0-au_M@x" > /home/userftp/ftp/files/flag.txt
```

Résultats :

```
$ sudo ls -la /home/userftp/ftp/files
```

```
-rw-r--r-- 1 root      root      0 May 24 21:31 chaise.mp4
-rw-r--r-- 1 root      root      0 May 24 21:36 flag.txt
-rw-r--r-- 1 root      root      0 May 24 21:31 souris.txt
-rw-r--r-- 1 root      root      0 May 24 21:32 voiture.png
```

```
$ cat /home/userftp/ftp/files/flag.txt
```

```
r4ti0-au_M@x
```

Le serveur est bien accessible avec **Filezilla**, les fichiers aussi :

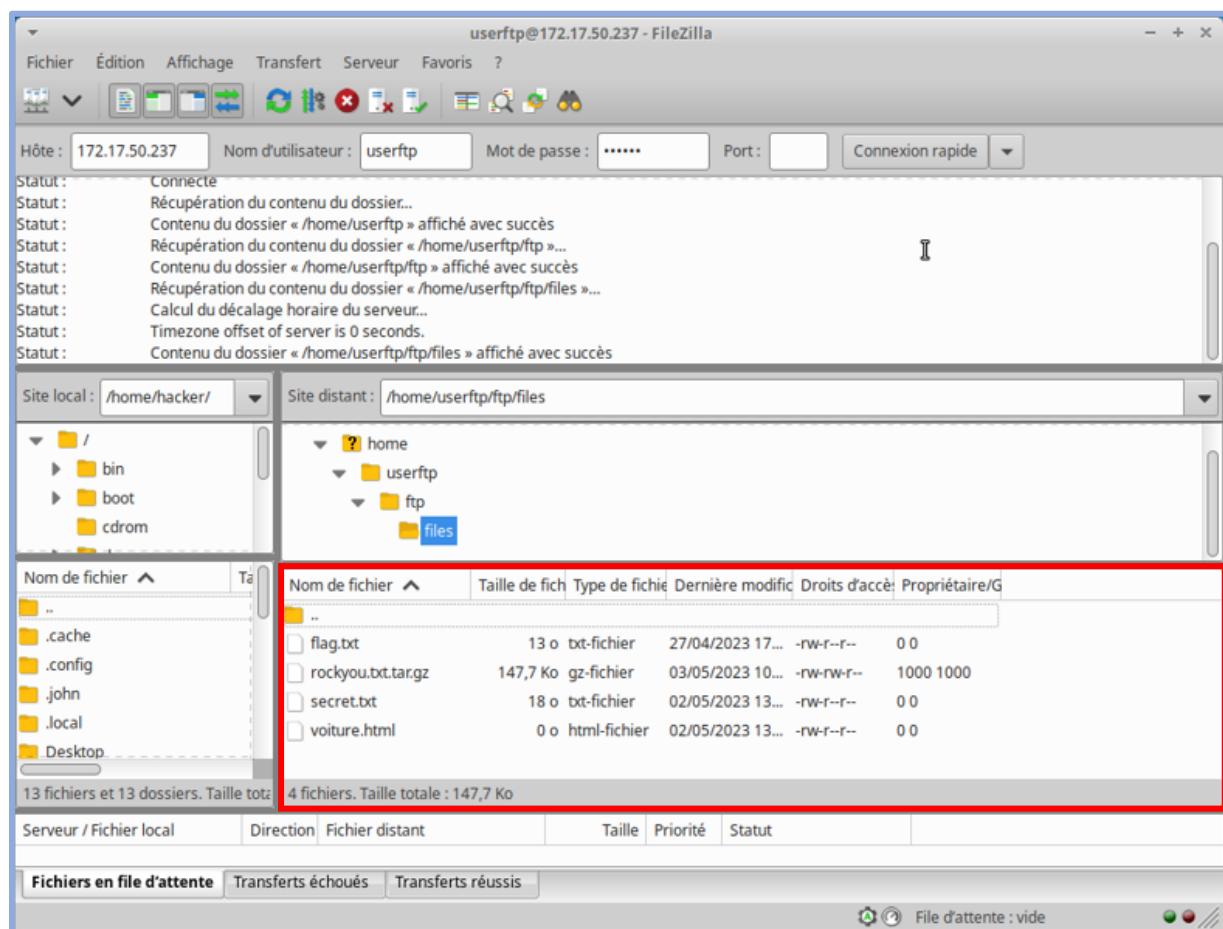


Figure 6.6.15: accès au serveur FTP avec Filezilla

L'attaque est réalisable avec **hydra** :

```
hacker@backbox-vm:~$ hydra -l userftp -P /usr/share/wordlists/rockyou.txt ftp://172.17.50.237 -V
Hydra v9.2 (c) 2021 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-05-03 11:30:38
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344398 login tries (l:1/p:14344398), ~896525 tries per task
[DATA] attacking ftp://172.17.50.237:21/
[ATTEMPT] target 172.17.50.237 - login "userftp" - pass "123456" - 1 of 14344398 [child 0] (0/0)
[ATTEMPT] target 172.17.50.237 - login "userftp" - pass "12345" - 2 of 14344398 [child 1] (0/0)
[ATTEMPT] target 172.17.50.237 - login "userftp" - pass "123456789" - 3 of 14344398 [child 2] (0/0)
[ATTEMPT] target 172.17.50.237 - login "userftp" - pass "password" - 4 of 14344398 [child 3] (0/0)
[ATTEMPT] target 172.17.50.237 - login "userftp" - pass "iloveyou" - 5 of 14344398 [child 4] (0/0)
```

Elle abouti, on trouve bien le mot de passe utilisé avec Filezilla pour accéder au serveur :

```
[ATTEMPT] target 172.17.50.237 - login "userftp" - pass "hottie" - 62 of 14344398 [child 13] (0/0)
[ATTEMPT] target 172.17.50.237 - login "userftp" - pass "tinkerbell" - 63 of 14344398 [child 14] (0/0)
[ATTEMPT] target 172.17.50.237 - login "userftp" - pass "charlie" - 64 of 14344398 [child 15] (0/0)
[21][ftp] host: 172.17.50.237 login: userftp password: hannah
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-05-03 11:31:00
hacker@backbox-vm:~$
```

### 7.1.7 Conclusion

Ce projet m'a permis d'avoir une idée de la manière dont une entreprise gère ses projets via la méthodologie SCRUM et ce qu'elle peut attendre des développeurs comme nous faisions comme si nous avions un client (M. DARTIGALONGUE au départ puis le BTS CIEL de l'année prochaine par la suite).

J'ai découvert le développement web en Python avec Flask et Jinja qui m'était totalement inconnu auparavant. Différents outils afin de maintenir une qualité de code convenable (PyLint, Black, isort, Flake8, ...). Puis une utilisation plus poussée de Bootstrap.

Le fait de travailler en groupe m'a permis de voir différentes façons de développer une solution, propre à chacune, me donnant de nouvelles idées pour aborder les choses.

Étant de nature réservée, ce projet de fin d'année m'a aussi fait sortir de ma zone, m'obligeant à communiquer avec mes coéquipiers.

Fièvre d'avoir contribué au projet qui devrait être utilisé l'année prochaine et donc avoir une utilisation concrète rendant aussi bien service aux enseignants du BTS CIEL qu'aux futurs étudiants de cette section.

## 8 Conclusion

Le projet a été mené à bien dans les délais prévus, ce qui représente une réussite significative pour notre équipe. Grâce à nos efforts, notre site propose désormais une formation complète grâce à des défis CTF (Capture The Flag). Cette plateforme de formation offre une occasion précieuse de s'entraîner et de se familiariser avec les principes fondamentaux de la sécurité informatique.

Au cours de ce projet, nous avons pu développer et améliorer nos compétences en programmation, en particulier dans le langage Python, ce qui a été particulièrement bénéfique pour Adrien BRUAS et Stefen INCE. Nous avons pu acquérir une expertise approfondie dans ce langage de programmation polyvalent et largement utilisé dans le domaine de la cybersécurité.

Par ailleurs, ce projet nous a permis d'apprendre comment structurer efficacement un projet, en mettant en œuvre les pratiques et les outils de DevOps. Grâce à l'utilisation de DevOps, nous avons pu optimiser nos processus de développement, de déploiement et de gestion du temps, ce qui a contribué à la réussite globale du projet.

En envisageant l'avenir, nous sommes conscients de l'importance de pouvoir continuer à enrichir notre plateforme. Il est donc possible d'ajouter de nouvelles salles de formation afin d'élargir les domaines de connaissances abordés sur notre site. Cette possibilité de diversification permettrait aux utilisateurs d'explorer davantage de sujets liés à la cybersécurité, mais également à d'autres domaines.

En résumé, grâce à notre travail acharné et à notre collaboration, nous avons réussi à mener à bien ce projet dans les délais impartis. Notre site offre maintenant une plateforme de défis CTF, renforçant ainsi nos compétences en Python et notre compréhension des principes de DevOps.

## 9 Annexes

### 9.1 Extrait de rapport de test HTML

#### report.html

Report generated on 23-May-2023 at 16:23:16 by [pytest-html v3.2.0](#)

#### Environment

|          |                                                          |
|----------|----------------------------------------------------------|
| Packages | {"pluggy": "1.0.0", "pytest": "7.2.1"}                   |
| Platform | Windows-10-10.0.19042-SP0                                |
| Plugins  | {"flask": "1.2.0", "html": "3.2.0", "metadata": "2.0.4"} |
| Python   | 3.10.10                                                  |

#### Summary

34 tests ran in 4.70 seconds.

(Un)check the boxes to filter the results.

34 passed,  0 skipped,  0 failed,  0 errors,  0 expected failures,  0 unexpected passes

#### Results

[Show all details](#) / [Hide all details](#)

| Result                | Test                                                                                                       | Duration | Links |
|-----------------------|------------------------------------------------------------------------------------------------------------|----------|-------|
| Passed (show details) | tests/functional/frontend/test_access.py::test_front_page_access                                           | 3.09     |       |
| Passed (show details) | tests/functional/frontend/test_access.py::test_route_access[main.acceuil-200-LoginLevel.NOT_LOGGED_IN]     | 0.00     |       |
| Passed (show details) | tests/functional/frontend/test_access.py::test_route_access[main.acceuil-200-LoginLevel.REGULAR_USER]      | 0.03     |       |
| Passed (show details) | tests/functional/frontend/test_access.py::test_route_access[main.liste_room-200-LoginLevel.NOT_LOGGED_IN]  | 0.00     |       |
| Passed (show details) | tests/functional/frontend/test_access.py::test_route_access[main.liste_room-200-LoginLevel.REGULAR_USER]   | 0.01     |       |
| Passed (show details) | tests/functional/frontend/test_access.py::test_route_access[main.connexion-200-LoginLevel.NOT_LOGGED_IN]   | 0.01     |       |
| Passed (show details) | tests/functional/frontend/test_access.py::test_route_access[main.deconnexion-401-LoginLevel.NOT_LOGGED_IN] | 0.00     |       |
| Passed (show details) | tests/functional/frontend/test_access.py::test_route_access[main.deconnexion-200-LoginLevel.REGULAR_USER]  | 0.00     |       |
| Passed (show details) | tests/functional/frontend/test_access.py::test_route_access[admin.index-403-LoginLevel.REGULAR_USER]       | 0.00     |       |
| Passed (show details) | tests/functional/frontend/test_access.py::test_route_access[admin.index-200>LoginLevel.ADMIN]              | 0.04     |       |
| Passed (show details) | tests/functional/frontend/test_access.py::test_route_access[api.doc-200>LoginLevel.NOT_LOGGED_IN]          | 0.01     |       |
| Passed (show details) | tests/functional/frontend/test_access.py::test_route_access[api.doc-200/LoginLevel.REGULAR_USER]           | 0.00     |       |
| Passed (show details) | tests/functional/frontend/test_access.py::test_route_access[api.doc-200>LoginLevel.ADMIN]                  | 0.00     |       |
| Passed (show details) | tests/functional/frontend/test_access.py::test_route_access[main.classement-200-LoginLevel.NOT_LOGGED_IN]  | 0.02     |       |
| Passed (show details) | tests/functional/frontend/test_access.py::test_route_access[main.classement-200-LoginLevel.REGULAR_USER]   | 0.01     |       |
| Passed (show details) | tests/functional/frontend/test_access.py::test_route_access[main.classement-200>LoginLevel.ADMIN]          | 0.01     |       |
| Passed (show details) | tests/functional/frontend/test_no_errors.py::test_routes_no_server_errors[None]                            | 0.04     |       |
| Passed (show details) | tests/functional/frontend/test_no_errors.py::test_routes_no_server_errors[regular_user]                    | 0.05     |       |

## 9.2 Classe abstraite VMManager

```

class VMManager(ABC):
    """
    Abstract Base Class d'un gestionnaire de VM
    """

    @abstractmethod
    def start_vm(self, vm_id: int, *, wait_until_on: bool = True):
        """Allume une machine virtuelle.

        Args:
            vm_id (int): L'id de la machine virtuelle.
            wait_until_on (bool): Bloque jusque à ce que la VM soit en ligne.
        """
        ...

    @abstractmethod
    def stop_vm(self, vm_id: int, *, wait_until_off: bool = True):
        """Eteind une machine virtuelle.

        Args:
            vm_id (int): L'id de la machine virtuelle.
            wait_until_off (bool): Bloque jusque à ce que la VM soit hors ligne.
        """
        ...

    @abstractmethod
    def setup(self, template_id: int, vm_name: str, vnc: bool = False) -> dict:
        """Met en place une machine virtuelle à partir d'un template, avec une adresse IP,
        optionnellement VNC.

        Args:
            template_id (int): L'ID du template à cloné.
            vnc (bool, optional): Si True, alors un display port VNC sera alloué à la VM. Defaults to False.

        Returns:
            VirtualMachine: Les informations de la machine virtuelle.
        """
        ...

    @abstractmethod
    def delete_vm(self, vm_id: int):
        """Supprime une VM sur proxmox.

        Args:
            vm_id (int): L'ID de la VM à supprimer.
        """
        ...

```

### 9.3 Fonction get\_n\_around

```

def get_n_around(lst: list[T], index: int, amount: int) -> list[T]:
    """Renvoie les n éléments les plus proches de l'index "index", incluant
    l'élément à la position index.

    Args:
        lst (list[T]): La liste contenant les éléments
        index (int): L'index servant de centre dans lst
        amount (int): Combien d'objets récupérer

    Raises:
        ValueError: amount est négatif, ou index n'est pas dans la liste

    Returns:
        list[T]: Une liste de "amount" éléments, ou de len(lst) si len(lst) < amount
    """

# Sanity checks
if amount < 0:
    raise ValueError("Amount can't be negative")
if not 0 <= index < len(lst):
    raise ValueError("Index not in list")

# * The list doesn't have enough elements, so we return it directly
# * since it's the max amount we'll be able to return
if len(lst) <= amount:
    return lst

def neg_pos_count() -> Iterator[int]:
    """Yields des int grandissant, d'abord négatif puis positif
    ex: 0, -1, -1, -2, 2, -3, ...
    """

    Yields:
        Iterator[int]: Un iterator de ints sans fin
    """

    yield 0
    for i in count(start=1):
        yield -i
        yield i

# We use a deque since we'll add both to the start and the end
output = deque()
index_iterator = neg_pos_count()
while len(output) < amount:
    target_index = index + next(index_iterator)
    # We don't want indexes outside the list
    if not 0 <= target_index < len(lst):
        continue

    # * We add to the start or the end, keeping the original list's order
    if target_index < index:
        output.appendleft(lst[target_index])
    else:
        output.append(lst[target_index])

return list(output)

```

## 9.4 Tests unitaires pour la fonction get\_n\_around

```

@pytest.mark.parametrize(
    "lst,index,amount,expected",
    [
        ([1, 2, 3, 4, 5], 0, 3, [1, 2, 3]),
        ([1, 2, 3, 4, 5], 4, 3, [3, 4, 5]),
        ([5, 4, 3, 2, 1], 2, 3, [4, 3, 2]),
        ([1, 2, 3], 1, 500, [1, 2, 3]),
        ([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], 1, 5, [0, 1, 2, 3, 4]),
        ([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], 2, 5, [0, 1, 2, 3, 4]),
        ([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], 6, 5, [4, 5, 6, 7, 8]),
    ],
)
def test_get_n_around(lst, index, amount, expected):
    """
    Teste que get_n_around renvoie bien les bonnes valeurs
    """
    assert get_n_around(lst=lst, index=index, amount=amount) == expected

def test_get_n_around_errors():
    """
    Teste que get_n_around n'accepte pas de mauvais paramètres
    """
    lst = [1, 2, 3, 4, 5]

    # On vérifie que des valeurs incorrectes fassent les erreurs appropriées
    with pytest.raises(ValueError):
        get_n_around(lst, 122, 2)

    with pytest.raises(ValueError):
        get_n_around(lst, -2, 2)

    with pytest.raises(ValueError):
        get_n_around(lst, 1, -5)

```

## 9.5 Utilisation d'un formulaire WTForm – Template Jinja

```

{% extends 'base.jinja' %}

{% block title %}
{{ super() }} - Connexion
{% endblock %}

{% block content %}
<main class="bg-black">
    
    <div class="modal modal-signin position-static d-block bg-black py-5" tabindex="-1" role="dialog" id="modalSignin">
        <div class="modal-dialog" role="document">
            <div class="modal-content rounded-2 bg-dark">
                <div class="modal-header px-4 pb-0 border-bottom-0">
                    <!-- <h1 class="modal-title fs-5" >Modal title</h1> -->
                    <h1 class="text-white">
                        
                        &nbsp;Se connecter
                    </h1>
                </div>
                <span class="login-description p-5 px-4 pt-0 fw-semibold ms-1 fs-7">Identifiants personnels</span>
            <div class="modal-body p-5 px-4 pt-0">

                <form method="POST" action="/connexion">
                    {% Token CSRF %}
                    {{ login_form.csrf_token }}

                    {% LOGIN %}
                    <div class="mb-3 col-12">
                        <label for="username"
                            class="form-label text-white fw-semibold ms-1 fs-5">{{login_form.login.label}}</label>
                        {{ login_form.login(class_="input-login-username form-control text-white bg-black rounded-2
                            p-1 ps-3 border border-1 border-info-subtle") }}
                        {% if login_form.login.errors %}
                            <ul class="errors">
                                {% for error in login_form.login.errors %}
                                    <li>{{ error }}</li>
                                {% endfor %}
                            </ul>
                        {% endif %}
                    </div>

                    {% MOT DE PASSE %}
                    <div class="mb-3 col-12">
                        <label for="password"
                            class="form-label text-white fw-semibold ms-1 fs-5">{{login_form.password.label}}</label>
                        {{login_form.password(class_="input-login-password form-control text-white bg-black
                            rounded-2 p-1 ps-3 border border-1 border-info-subtle")}}
                        {% if login_form.password.errors %}
                            <ul class="errors">
                                {% for error in login_form.password.errors %}
                                    <li>{{ error }}</li>
                                {% endfor %}
                            </ul>
                        {% endif %}
                    </div>

                    {% BOUTON VALIDER %}
                    <div class="d-grid gap-2 d-md-flex justify-content-md-end">
                        {{ login_form.submit(class_="btn-valider btn rounded-1 border-2 active text-info fs-5
                            fw-semibold px-4 py-2 mt-5") }}
                    </div>
                </form>
            </div>
        </div>
    </div>
</main>
{% endblock %}

```

## 9.6 Utilisation d'un formulaire WTForm – Code Flask

```
@main.route("/connexion", methods=["GET", "POST"])
def connexion():
    """Formulaire permettant la connexion."""
    form = LoginForm()
    # Si le formulaire a été correctement rempli
    if form.validate_on_submit():
        # On cherche l'utilisateur demandé
        user: User = User.query.filter_by(username=form.login.data).first()
        if user is None:
            form.login.errors.append("Cet utilisateur n'existe pas.")
            flash("user inexistant", "error")

        # Vérification du hash du mot de passe
        elif not user.verify_password(form.password.data):
            form.password.errors.append("Mot de passe invalide.")

    # Pas d'erreur, on connecte l'utilisateur avec Flask-login (ajout des cookies de session)
    else:
        login_user(user)

        flash("Logged in successfully.", "success")

    next_url = request.args.get("next")
    # On redirige vers la page d'accueil
    return redirect(next_url or url_for("main.acceuil"))

    # Si on a pas submit une form valide, ou alors que l'on avais
    # un MDP invalide ou un login qui n'existaient pas
    return render_template("connexion.jinja", login_form=form)
```

## 9.7 Ressource Flask-restx pour les questions

```

@room_namespace.route("/question/<id>")
@room_namespace.response(200, "Succès")
@room_namespace.response(401, "L'utilisateur n'est pas connecté")
@room_namespace.response(403, "L'utilisateur n'a pas les priviléges requis")
@room_namespace.response(404, "La question n'existe pas")
class QuestionResource(Resource):
    """Informations lié à une question"""

    @room_namespace.marshal_with(question_model, as_list=False)
    def get(self, id):
        """Récupère les informations lié a une question."""
        question: models.User = models.Question.query.filter_by(id=id).first_or_404(
            description="Cet question n'existe pas."
        )
        return question_schema.dump(question)

    @room_namespace.marshal_with(question_model, as_list=False)
    def post(self, id):
        """Modifie les informations lié a une question."""
        if not current_user.is_authenticated:
            abort(401)
        if not current_user.is_admin:
            abort(403)

        question: models.User = models.Question.query.filter_by(id=id).first_or_404(
            description="Cet question n'existe pas."
        )

        data: dict = request.json
        if data.get("id"):
            del data["id"]

        for key, value in data.items():
            setattr(question, key, value)

        db.session.commit()
        return question_schema.dump(question)

    def delete(self, id):
        """Supprime une question"""
        if not current_user.is_authenticated:
            abort(401)
        if not current_user.is_admin:
            abort(403)

        question: models.User = models.Question.query.filter_by(id=id).first_or_404(
            description="Cet question n'existe pas."
        )
        db.session.delete(question)
        db.session.commit()

```

## 9.8 Classe modèle SQLAlchemy

```

class Question(db.Model):
    """Une question appartenant à une room"""

    __tablename__ = "questions"

    id: Mapped[int] = mapped_column(primary_key=True)
    """Identifiant unique de la question. N'est pas montré à l'utilisateur."""

    room_id: Mapped[int] = mapped_column(_current_foreign_key("rooms.id"))
    room: Mapped[Room] = _current_relationship(back_populates="questions")

    prompt: Mapped[str]
    """L'énoncé de la question affiché à l'utilisateur. Peut contenir du markdown."""
    answer: Mapped[str]
    """La réponse attendu à la question. Non-sensible à la case."""

    points: Mapped[int]
    """
    Le nombre de points que vaut la question.
    Utilisé pour calculer le score des utilisateurs.
    """

    # Les SolvedQuestionData sont supprimé automatiquement lorsque l'on supprime une question
    solved_questions_data: Mapped[list["SolvedQuestionData"]] = relationship(
        back_populates="question", cascade="all, delete, delete-orphan"
    )

    def is_solved_by(self, user: User) -> bool:
        """Vérifie si cette question a été résolu par un utilisateur.

        Args:
            user (User): L'utilisateur à vérifier.

        Returns:
            bool: True si la question a été résolue, sinon False.
        """
        user_question_data = SolvedQuestionData.query.filter_by(
            user_id=user.id, question_id=self.id
        ).first()

        return user_question_data is not None

    def solve(self, user: User):
        """Marque une question comme complété par un utilisateur.

        Args:
            user (User): L'utilisateur qui a résolu la question.
        """
        if not self.is_solved_by(user):
            user_question_data = SolvedQuestionData(
                user_id=user.id, question_id=self.id
            )
            db.session.add(user_question_data)
            db.session.commit()

```

## 9.9 Code complet du filtre markdown (fonction markdown\_filter)

```
def markdown_filter(text: str) -> str:  
    """Interprète le markdown vers de l'HTML  
  
    Args:  
        text (str): Texte contenant du markdown  
  
    Returns:  
        str: HTML de ce texte.  
    """  
  
    # Extensions utilisé:  
    # - fenced_code: permet l'utilisation de code block avec ```  
    # - nl2br: remplace \n par <br>  
    # - CodeHiliteExtension: permet le synthax highlighting dans les code blocks  
    # - tables: permet l'utilisation de tableaux  
    return md(  
        text,  
        extensions=[  
            "fenced_code",  
            "nl2br",  
            CodeHiliteExtension(guess_lang=True, linenums=None),  
            "tables",  
        ],  
    )
```

## 9.10 Code HTML complet de la page d'accueil

```

<% extends 'base.jinja' %>




{% block title %}
{{ super() }} - Accueil
{% endblock %}

{% block content %}

{% if config.ENABLE_ANIMATED_BACKGROUND %}


- 
- 
- 
- 
- 
- 
- 
- 


{% endif %}

<main class="container pt-5 pb-5 mb-4 ">
  <div class="container-index row">
    <!-- partie gauche avec logo + texte -->
    <div class="index-left-part col text-center pb-5">
      
      <h1 class="text-white fs-2 fw-normal">Une plateforme rapide, accessible et réaliste pour tester vos compétences en
      | hacking.
      </h1>
      <h3 class="text-info fs-3 fw-normal m-0">la root est longue mais la voie est Libre</h3>
    </div>

    <!-- partie droite avec formulaire d'inscription -->
    {% if not current_user.is_authenticated %}
      <div class="index-right-part modal modal-signin position-static d-block col" tabindex="-1" role="dialog"
        id="modalSignin">
        <div class="modal-dialog" role="document">
          <div class="modal-content rounded-2 bg-dark">
            <div class="modal-header px-4 pb-0 border-bottom-0">
              <h1 class="text-white">
                | &nbsp;S'inscrire
              </h1>
            </div>

            <!-- formulaire -->
            <div class="modal-body p-5 px-4 pt-0">
              <form method="POST" action="{{ url_for('main.inscription') }}">
                
                {{ signup_form.csrf_token }}

                <!-- NOM UTILISATEUR / PSEUDO -->
                <div class="mb-3 col-12">
                  <label for="username">
                    | class="form-label text-white fw-semibold ms-1 fs-5">{{signup_form.username.label}}</label>
                    {{ signup_form.username(class_="input-login-username form-control text-white bg-black rounded-2 p-1 ps-3
                    border border-1 border-info-subtle") }}
                  </div>
                </div>
              </form>
            </div>
          </div>
        </div>
      </div>
    {% endif %}
  </div>
</main>

```

```

<!-- Email -->
<div class="mb-3 col-12">
  <label for="email"
    | class="form-label text-white fw-semibold ms-1 fs-5">{{signup_form.email.label}}</label>
  {{ signup_form.email(class_="input-login-username form-control text-white bg-black rounded-2 p-1 ps-3 border border-1 border-info-subtle") }}
</div>

<!-- PASSWORD -->
<div class="mb-3 col-12">
  <label for="password"
    | class="form-label text-white fw-semibold ms-1 fs-5">{{signup_form.password.label}}</label>
  {{ signup_form.password(class_="input-login-password form-control text-white bg-black rounded-2 p-1 ps-3 border border-1 border-info-subtle") }}
</div>

<!-- PASSWORD CONFIRMATION -->
<div class="mb-3 col-12">
  <label for="password_confirmation"
    | class="form-label text-white fw-semibold ms-1 fs-5">{{signup_form.password_confirmation.label}}</label>
  {{ signup_form.password_confirmation(class_="input-login-password form-control text-white bg-black rounded-2 p-1 ps-3 border border-1 border-info-subtle") }}
</div>

{# bouton valider #}
<div class="d-grid gap-2 d-md-flex justify-content-md-end">
  {{ signup_form.submit(class_="btn-valider btn rounded-1 border-2 active text-info fs-5 fw-semibold px-4 py-2 mt-5") }}
</div>
</form>
</div>
</div>
</div> <!-- fin formulaire de connexion a cette div -->
</div>
{% endif %}
{% if current_user.is_authenticated %}
<div class="d-flex row align-items-md-stretch">

  {% for r in rooms %}
  {{room_widget(r.name, r.description, r.url_name)}}
  {% endfor %}

  <div class="col-md-6 p-3">
    <div class="{{config.BACKGROUND_BLOCK}} h-100 p-5 rounded-3 {{config.TEXT_COLOR_BLOCK}}">
      <h2>Liste Rooms</h2>
      <p>liste de toutes les rooms</p>
      <a href="/liste_rooms" class="btn rounded-1 border-2 active text-info fw-semibold px-4 py-2 mt-5"
        | role="button">Lancer</a>
    </div>
  </div>

  <div class="col-md-6 p-3">
    <div class="{{config.BACKGROUND_BLOCK}} h-100 p-5 rounded-3 {{config.TEXT_COLOR_BLOCK}}">
      <h2>Utilisation Générale</h2>
      <p>utilisation des rooms et des machines virtuelles</p>
      <a href="/utilisation" class="btn rounded-1 border-2 active text-info fw-semibold px-4 py-2 mt-5"
        | role="button">Lancer</a>
    </div>
  </div>

  {% endif %}
</div>
</main>

{% endblock %}

```

## 9.11 Code HTML du header partie utilisateur connecté

```

{% block header %}
<header class="text-dark bg-transparent">
    <div class="container-fluid m-0">
        <div class="d-flex align-items-center justify-content-between ">
            <a href="/" class="d-flex align-items-center mb-2 mb-lg-0 text-white text-decoration-none">
                
                <span class="site-name" itemprop="name">{{config.APP_NAME}}</span>
            </a>

            <div class="text-end d-flex align-items-center">
                {% if current_user.is_authenticated %}
                    <div class="nom-utilisateur">
                        <h4> {{current_user.username}} </h4>
                    </div>
                    <!-- creation menu / bouton qui permet le défilement - dropdown -->
                    <div class="dropdown text-end">
                        <a href="{{ url_for('main.profile') }}"
                            class="curseur-avatar d-block link-dark text-decoration-none dropdown-toggle"
                            data-bs-toggle="dropdown" aria-expanded="false">
                            
                        </a>
                        <ul class="dropdown-content dropdown-menu text-small" style="">
                            <!-- contenu du menu / ce qui défile -->
                            <li> # lien vers profil user #
                                <a class="dropdown-item" href="{{ url_for('main.profile') }}"
                                    ><span>Mon profil</span>
                                    
                                </a>
                            </li>
                            <li> # lien vers classement #
                                <a class="dropdown-item" href="{{ url_for('main.classement') }}"
                                    ><span>Classement</span>
                                    
                                </a>
                            </li>
                            <li> # lien vers liste des rooms #
                                <a class="dropdown-item" href="{{ url_for('main.liste_room') }}"
                                    ><span>Rooms</span>
                                    
                                </a>
                            </li>
                        </ul>
                    </div>
                {% si user connecté en admin --> liens vers panel admin %}
                {% if current_user.is_admin == true %}
                    <li> # lien vers panel administrateur #
                        <a class="dropdown-item" href="{{ url_for('admin.index') }}"
                            ><span>Panel admin</span>
                            
                        </a>
                    </li>
                {% endif%}

                <li> <!-- simple séparation estétique -->
                    <hr class="dropdown-divider">
                </li>
                <li> # bouton déconnexion #
                    <a class="dropdown-item" href="{{ url_for('main.deconnexion') }}"
                        ><span>Se déconnecter</span>
                        
                    </a>
                </li>
            </ul>
        </div>
    </div>
</header>

```

## 9.12 Code HTML du header partie l'utilisateur n'est pas connecté

```
{# si l'utilisateur n'est pas connecté - afficher header avec bouton connexion #}
{% else %}
<a href="{% url_for('main.connexion') %}" 
    class="btn rounded-1 border-2 active text-info fw-semibold px-2 py-1" role="button"
    aria-pressed="true">J'ai déjà un compte</a>
<a class="a-rootme-pro" href="http://www.lyceebranly.com/">
    
</a>
{% endif %}
</div>
</div>
</div>
<% endblock %>
```

### 9.13 Code HTML du header partie l'utilisateur n'est pas connecté

```

{% block footer %}
<footer class="page-footer container-fluid pt-3 align-items-center">
    <!-- logo rond blanc -->
    <div class="d-flex justify-content-center">
        
    </div>
    <!-- informations à propos du site -->
    <ul class="nav justify-content-center">
        <li class="nav-item"><a href="{{config.CONFIDENTIALITE_LINK}}">
            class="nav-link px-2 text-muted text-secondary-emphasis fs-6">Confidentialité</a></li>
        <li class="nav-item"><a href="{{config.MENTIONS_LEGALES_LINK}}">
            class="nav-link px-2 text-secondary-emphasis fs-6">Mentions Légales</a></li>
        <li class="nav-item">
            <a href="{{config.CONDITION_GENERALES_D_UTILISATION_LINK}}">
                class="nav-link px-2 text-secondary-emphasis fs-6">Conditions Générales d'Utilisation</a>
            </li>
        </ul>
    <!-- phrase accolche + copyright -->
    <p class="text-center p-0 m-0 text-white fs-6">{{config.SENTENCE_FOOTER_END}}</p>
    <p class="text-center border-bottom border-secondary pb-3 mb-0 text-white fs-6">{{config.COPYRIGHT}}</p>

    <!-- liens extérieurs qui concernent le site-->
    <ul class="nav justify-content-center bg-black pt-3">
        {% es logos se cachent bien si l'URL n'existe pas %}
        {% if config.TWITTER_LINK %}
            <li class="nav-item px-3">
                <a href="{{config.TWITTER_LINK}}" class="nav-link px-2">
                    
                </a>
            </li>
        {% endif %}

        {% if config.LINKEDIN_LINK %}
            <li class="nav-item px-3">
                <a href="{{config.LINKEDIN_LINK}}" class="nav-link px-2">
                    
                </a>
            </li>
        {% endif %}

        {% if config.RSS_FEED_LINK %}
            <li class="nav-item px-3">
                <a href="{{config.RSS_FEED_LINK}}" class="nav-link px-2">
                    
                </a>
            </li>
        {% endif %}

        {% if config.DISCORD_LINK %}
            <li class="nav-item px-3">
                <a href="{{config.DISCORD_LINK}}" class="nav-link px-2">
                    
                </a>
            </li>
        {% endif %}
    </ul>
</footer>
{% endblock %}

```

## 9.14 Liens utilisés dans le footer dans app\_config.py

```
# ===== footer =====
CONFIDENTIALITE_LINK: str = "/confidentialite"
"""L'URL vers la page de confidentialité"""
MENTIONS_LEGALES_LINK: str = "/mention_legales"
"""L'URL vers la page des mentions légales"""
CONDITION_GENERALES_D_UTILISATION_LINK = "/Conditions_generales_d_utilisation"
"""L'URL vers la page des conditions générales d'utilisation"""

SENTENCE_FOOTER_END: str = "FlagQuest : plateforme d'apprentissage dédiée au hacking et à la sécurité de l'information"
"""La phrase en haut du footer"""
COPYRIGHT: str = "© 2023"
"""Le copyright à afficher dans le footer"""

# ===== Liens Footer =====
# Laissez un lien vide pour le retirer
TWITTER_LINK: str = "https://twitter.com/intent/follow?original_referer=https%3A%2F%2Fwww.root-me.org%2F&region=follow_link&screen_name=rootme_org&tw_p=followbutton"
"""L'URL du compte Twitter (laissé vide pour cacher)"""
LINKEDIN_LINK: str = "https://www.linkedin.com/in/flag-quest-825419276/"
"""L'URL du compte Linkedin (laissé vide pour cacher)"""
RSS_FEED_LINK: str = ""
"""L'URL du flux RSS (laissé vide pour cacher)"""
DISCORD_LINK: str = ""
"""L'URL du serveur Discord (laissé vide pour cacher)"""


```

## 9.15 Code HTML tableau classement

```
<!-- tableau classement -->





```

## 9.16 Code HTML création de la route /profile

```

@main.route("/profile", defaults={"username": None})
@main.route("/profile/<username>")
def profile(username: str | None):
    """La page de profil de l'utilisateur"""

    # si on utilise l'URL : http://127.0.0.1:5000/profil
    # nom d'utilisateur pas spécifié
    if username is None:
        if current_user.is_authenticated:
            return redirect(url_for("main.profile", username=current_user.username))
        else:
            abort(401)
    else:
        user = User.query.filter_by(username=username).first_or_404(
            description="Cet utilisateur n'existe pas."
        )

    # --- graphique en bas à droite
    chart = UserWeekPoints()

    day_to_points: dict[str, int] = {}
    today = date.today()
    # For days from 6 day ago to today
    for offset in range(6, -1, -1):
        day = today - timedelta(days=offset)
        day_to_points[day.strftime("%d/%m/%Y")] = user.points_at_date(day)

    chart.labels.grouped = list(day_to_points.keys())
    chart.data.data = list(day_to_points.values())

    # partie classement
    all_users = list(sorted(User.query.all(), key=lambda u: u.score, reverse=True))
    user_index = all_users.index(user)

    ranking_users = [
        (all_users.index(u) + 1, u)
        for u in get_n_around(lst=all_users, index=user_index, amount=5)
    ]

    return render_template(
        "profile.jinja",
        user=user,
        chart_json=chart.get(),
        ranking_users=ranking_users,
        user_position=user_index + 1,
        user_rooms=list(user.joined_rooms),
    )

```

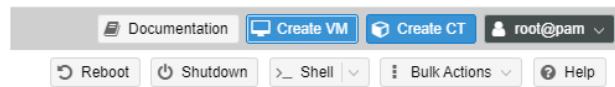
## 9.17 Procédure pour convertir une machine virtuelle en template

### Procédure pour convertir une machine virtuelle en template

adrien brus mardi

#### Création d'un machine virtuelles

On clique sur "Create VM"



Ensuite on nomme et on donne un ID à la machine virtuelles.

|        |        |                |
|--------|--------|----------------|
| Node:  | rootme | Resource Pool: |
| VM ID: | 102    |                |
| Name:  |        |                |

Advanced  Back Next

Ensuite on attribue un ISO à la machine

| Name                                    | For... | Size    |
|-----------------------------------------|--------|---------|
| backbox-8-desktop-amd64.iso             | iso    | 4.14 GB |
| kali-linux-2022.4-installer-amd64.iso   | iso    | 3.80 GB |
| lubuntu-22.04.2-desktop-amd64.iso       | iso    | 2.85 GB |
| <b>ubuntu-22.04.2-desktop-amd64.iso</b> | iso    | 4.93 GB |
| ubuntu-22.04.2-live-server-amd64.iso    | iso    | 1.98 GB |

Advanced  Back Next

Pour ajouter une ISO il faut aller dans "local" et choisir entre installer une ISO depuis un lien ou depuis son ordinateur.

Server View

Datacenter

rootme

- 100 (automatic-2b0ea64e053f43bf8a39)
- 103 (automatic-12499e8717ef41fab80b)
- 105 (VM 105)
- 101 (ubuntu-test)
- 104 (ubuntu-test-attaque)
- 120 (kali-linux-clone)
- 121 (ubuntu-server-ftp)
- 130 (attack-vm-backbox)
- local (rootme)**
- local-lvm (rootme)

Storage 'local' on node 'rootme'

ISO Images

backbox-8-desktop-amd64.iso  
kali-linux-2022.4-installer-amd64.iso  
lubuntu-22.04.2-desktop-amd64.iso  
ubuntu-22.04.2-desktop-amd64.iso  
ubuntu-22.04.2-live-server-amd64.iso

Upload Download from URL Remove

Ensuite pour finir l'installation de la machines il faut simplement la configurer comme on le souhaite.

il ne faut pas oublier de mettre la bonne interface virtuelles

Create: Virtual Machine

General OS System Disks CPU Memory Network Confirm

No network device

Bridge: **vmbr1** Model: **VirtIO (paravirtualized)**

VLAN Tag: **no VLAN** MAC address: **auto**

Firewall:

Help Advanced Back Next

## 9.18 Procédure de création d'un réseau virtuel

### Création d'un réseau virtuel

On clique sur "Create"

On donne le nom de l'interface virtuelle

Create: Linux Bridge

|                 |                                    |               |                                     |
|-----------------|------------------------------------|---------------|-------------------------------------|
| Name:           | <input type="text" value="vmbr2"/> | Autostart:    | <input checked="" type="checkbox"/> |
| IPv4/CIDR:      | <input type="text"/>               | VLAN aware:   | <input type="checkbox"/>            |
| Gateway (IPv4): | <input type="text"/>               | Bridge ports: | <input type="text"/>                |
| IPv6/CIDR:      | <input type="text"/>               | Comment:      | <input type="text"/>                |
| Gateway (IPv6): | <input type="text"/>               |               |                                     |

[Help](#) Advanced  [Create](#)

Enfin on donne une adresse IP du réseau

Create: Linux Bridge

|                 |                                          |               |                                     |
|-----------------|------------------------------------------|---------------|-------------------------------------|
| Name:           | <input type="text" value="vmbr1"/>       | Autostart:    | <input checked="" type="checkbox"/> |
| IPv4/CIDR:      | <input type="text" value="10.0.0.0/16"/> | VLAN aware:   | <input type="checkbox"/>            |
| Gateway (IPv4): | <input type="text"/>                     | Bridge ports: | <input type="text"/>                |
| IPv6/CIDR:      | <input type="text"/>                     | Comment:      | <input type="text"/>                |
| Gateway (IPv6): | <input type="text"/>                     |               |                                     |

[Help](#) Advanced  [Create](#)

## 9.19 Procédure d'installation des requirement

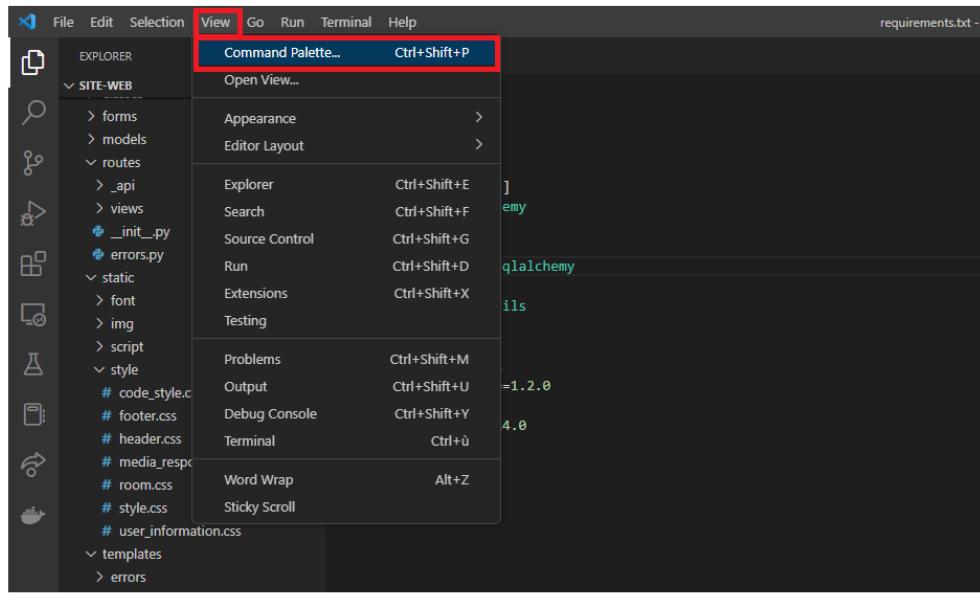
### installation requirement(développement).txt

adrien brus mardi

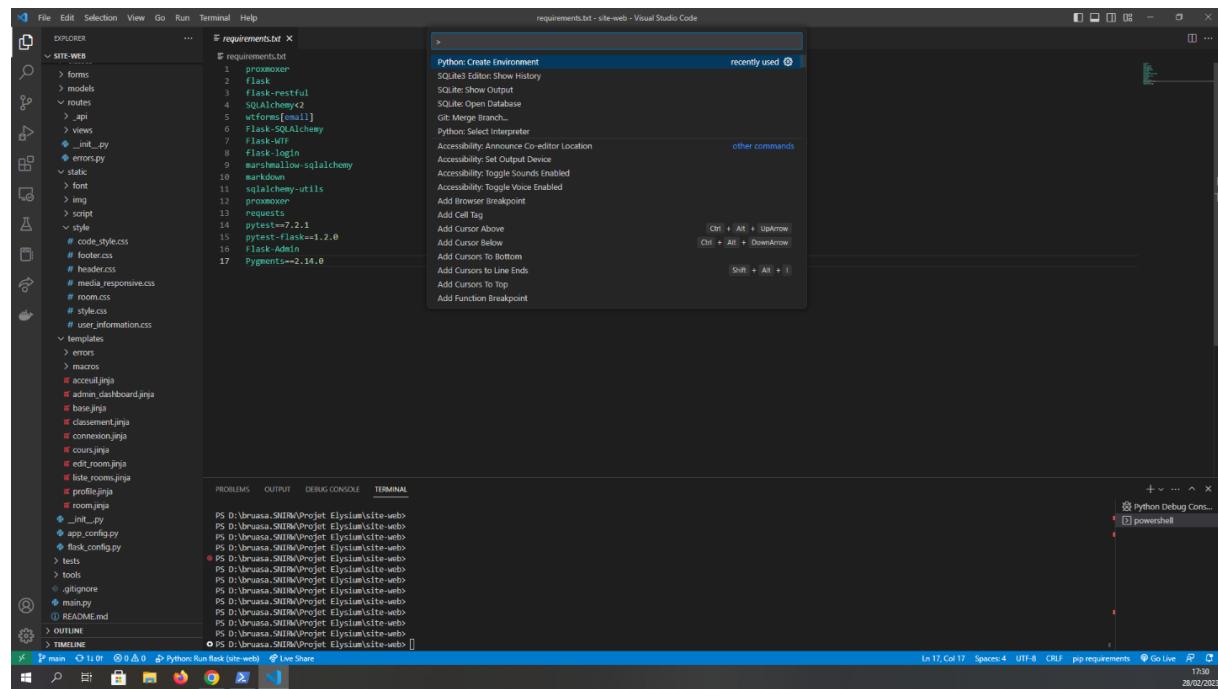
#### Installation des requirement du projet

pour installer les requirement dans Visual studio pour intégré la solution dans son propre environnement il faut tous d'abord installer certaines ressources pour y parvenir.

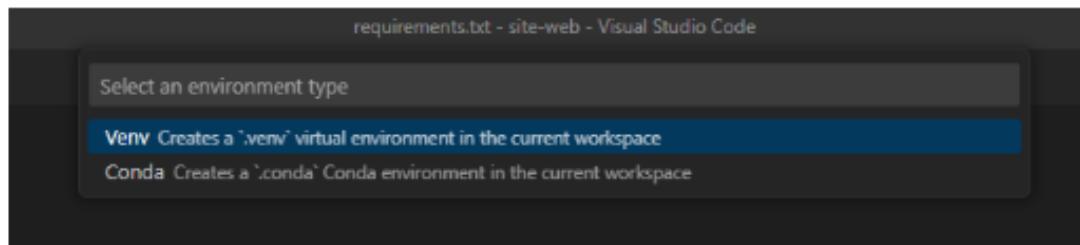
Pour cela il faut aller dans "View" puis dans "Command Palette".



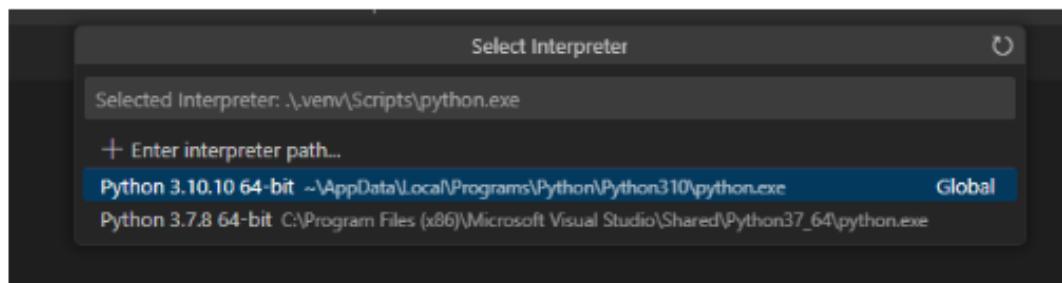
Ensute ou choisi "Python: Create Environment".



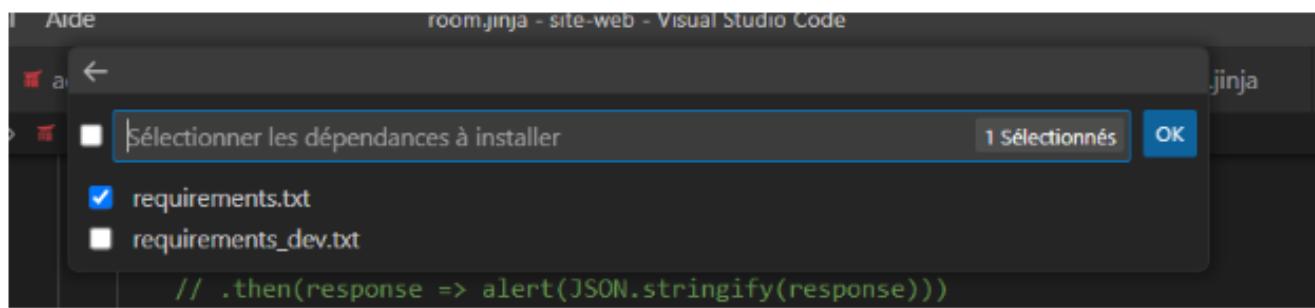
Puis on prend l'environnement "Venv".



Et on choisit la version de python.



Pour finir on choisit le fichier d'installation.



## 9.20 Procédure installation Python

### Installation python

adrien bruas · 3 mars

#### Etape 1: On récupère le fichier d'installation depuis un navigateur internet:

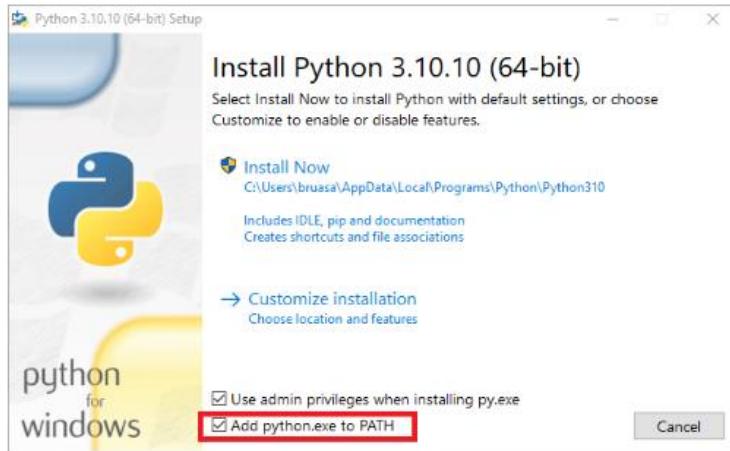
Dans un premier temps on installe une version ( éviter les version en beta test, dans notre cas nous avons choisi la version 3.10.10 ).<https://www.python.org/downloads/>

| Version                             | Operating System | Description              | MD5 Sum                          | File Size | GPG | Sigstore |
|-------------------------------------|------------------|--------------------------|----------------------------------|-----------|-----|----------|
| Gzipped source tarball              | Source release   |                          | 6dbe644dd1a520d9853cf6648084c346 | 26071329  | SIG | CRT SIG  |
| XZ compressed source tarball        | Source release   |                          | 7bf85df71bbe7f95e5370b983e6ae684 | 19627028  | SIG | CRT SIG  |
| macOS 64-bit universal2 installer   | macOS            | for macOS 10.9 and later | 892634724ab799569b512082c8f48c83 | 41005648  | SIG | CRT SIG  |
| Windows embeddable package (32-bit) | Windows          |                          | a681a7f9b242fe35b4d96d79e15e57d6 | 7663448   | SIG | CRT SIG  |
| Windows embeddable package (64-bit) | Windows          |                          | f38a9e7e02a992daa62569b758d0a388 | 8625602   | SIG | CRT SIG  |
| Windows help file                   | Windows          |                          | 448fb401ade49a7e2156d02512f2f9bf | 9391521   | SIG | CRT SIG  |
| Windows installer (32-bit)          | Windows          |                          | a81b81687bc2575c05a30f4b31d6ea00 | 27859200  | SIG | CRT SIG  |
| Windows installer (64-bit)          | Windows          | Recommended              | 9735797853cba809b13c8396c91354a0 | 29010904  | SIG | CRT SIG  |

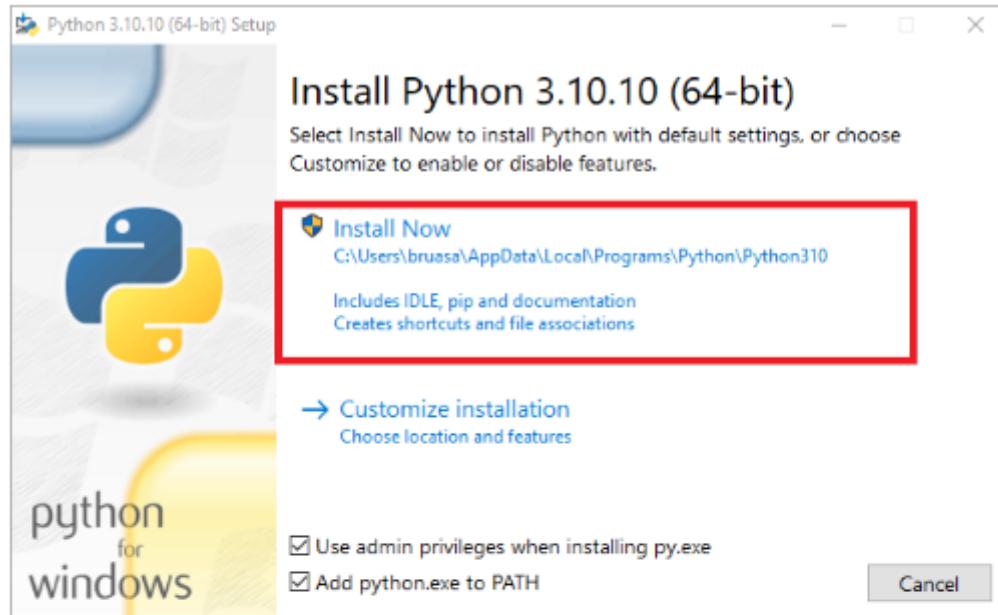
#### Etape 2: installation Python:

Dans un second temps on lance le fichier que l'on a téléchargé.

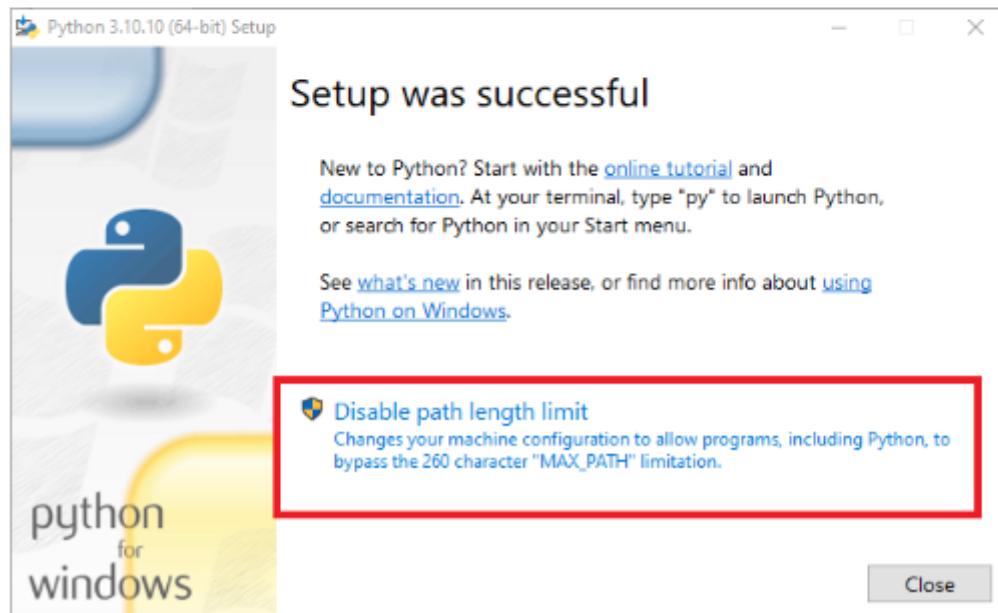
Ensuite on clique sur << Add python.exe to PATH >>



Puis on lance l'installation.



Enfin on enlève la limite de longueur du path, et on ferme la page et python est installer.



## 9.21 Procédure de déploiement du serveur ProxMox

Par la suite, nous utiliserons les identifiants suivants:  
 utilisateur : root  
 mot de passe : passw0rd

### Etape 1: Boot sur la clef USB

Afin d'installer Proxmox sur la machine, on doit booter sur l'ISO de Proxmox.

Afin d'aller sur le menu boot de votre machine, allez voir dans le manuel de celui-ci.

### Etape 2: Configuration de Proxmox

- Langage

Dans cette étape il suffit d'en suivre les indications données sur l'application tout en renseignant certaines informations telles que la localisation où vous vous trouvez, la langue du clavier.

- Administration

Dans un deuxième temps, il faut attribuer un mot de passe et une adresse mail pour pouvoir administrer Proxmox dans le futur.

- Configuration Réseau

Pour la configuration réseau de Proxmox il faut simplement mettre un Hostname, une IP, une Gateway, et un serveur DNS

### Etape 3: FIN

Une fois la configuration réseau fait l'installation de Proxmox va se lancer.

A la fin de l'installation, on retire clé USB puis on redémarre l'hyperviseur

## 9.22 Procédure de mise en place de dnsmasq (ProxMox)

### Prérequis:

- Les machines virtuelles seront connecté à proxmox sur l'interface bridge virtuelle **vmbr1**

### Etape 1: Installation

Sur un terminal proxmox

```
apt-get update
apt-get install dnsmasq
```

### Etape 2: Configuration

Dans le fichier de configuration `/etc/dnsmasq.conf`, on place les options suivantes:

```
# On répond seulement aux requêtes sur l'interface des VMs
interface=vmbr1

# On lis les IP statique de /etc/ethers
read-ethers

# On attribue les IPs du réseau 10.0.0.0/16, mais seulement celles spécifiées dans /etc/ethers
dhcp-range=10.0.0.0,static

# On désactive le serveur DNS de dnsmasq
port=0
```

### Etape 3: Vérification

On vérifie que la config est valide

```
dnsmasq --test
```

### Etape 4: Ajout de /etc/ethers

On modifie le fichier `ethers`, qui fait l'association entre les adresses MAC réservé et les adresses IP.

scp est particulièrement adapté pour cette tâche.

### Etape 5: Activation

On start le service et on l'active pour les prochain redémarrages

```
systemctl enable dnsmasq --now
```

### Etape 6: Vérification du bon fonctionnement

On crée une VM sur proxmox avec une des adresses mac de `/etc/ethers`.

On met **vmbr1** comme interface de la VM, et on la configure en DHCP. La méthode pour faire cela varie en fonction du système d'exploitation.  
 Enfin, on vérifie que l'IP associé à cette adresse mac nous est bien attribué.

## 9.23 Template base Ninja

```
<!doctype html>
<html lang="fr">

<head>
    {% block head %}
        <meta charset="utf-8">
        <meta name="viewport" content=
"width=device-width, initial-scale=1">
        <title>{% block title %}{{config.APP_NAME}}{% endblock %}
    </title>
    <!-- logo -->
    <link rel="icon" href="{{config.APP_FAVICO}}>
    <!-- css personnel -->
    <link rel="stylesheet" href="/static/style/style.css">
    <link rel="stylesheet" href="/static/style/code_style.css">
    <link rel="stylesheet" href=
"/static/style/media_responsive.css">
    <link rel="stylesheet" href="/static/style/header.css">
    <link rel="stylesheet" href=
"/static/style/user_information.css">
    <link rel="stylesheet" href="/static/style/room.css">
    <link rel="stylesheet" href="/static/style/footer.css">
    <link rel="stylesheet" href=
"/static/style/block_background.css">

    <link href=
"https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
        rel="stylesheet"
        integrity=
"sha384-GLhLTQ8iRABdZLl603oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA
6j6gD"
        crossorigin="anonymous">
```

```
{# script js bootstrap (utilisé pour le dropdown de l'avatar utilisateur) #}
<script src=
"https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js"
        integrity=
"sha384-w76AqPfDkMBDXo30jS1Sgez6pr3x5MlQ1ZAGC+nuZB+EYdgRZgiwxhTBTkF
7CXvN"
    crossorigin="anonymous">
</script>

<!-- Chart.JS -->
<script src=
"https://cdn.jsdelivr.net/npm/chart.js@4.2.1/dist/chart.umd.min.js"
></script>

<!-- JQuery -->
<script src="https://code.jquery.com/jquery-3.6.4.min.js"
        integrity=
"sha256-oP6HI9z1XaZNBrJURtCoUT5SUnxFr8s3BzRl+cbzUq8=" crossorigin=
"anonymous"></script>

<!-- Toastify -->
<link rel="stylesheet" href=
"https://cdn.jsdelivr.net/npm/toastify-js/src/toastify.min.css">
<script src="https://cdn.jsdelivr.net/npm/toastify-js"
></script>

<!-- Confetti -->
<script src=
"https://cdn.jsdelivr.net/npm/canvas-confetti@1.5.1/dist/confetti.browser.min.js"
></script>

<!-- Our own scripts -->
<script src=
"{{url_for('static', filename='script/global.js')}}"></script>
{% endblock %}
</head>
```

```

<body {% block body_args %}class="{{config.BACKGROUND}}" {% endblock
body_args %}>
    {% block header %}
        <header class="text-dark bg-transparent">
            <div class="container-fluid m-0">
                <div class=
"d-flex align-items-center justify-content-between ">
                    <a href="/" class=
"d-flex align-items-center mb-2 mb-lg-0 text-white text-decoration-none"
>
                        
                        <span class="site-name" itemprop="name">
{{config.APP_NAME}}</span>
                    </a>

                    <div class="text-end d-flex align-items-center">
                        {% if current_user.is_authenticated %}
                            <div class="nom-utilisateur">
                                <h4> {{current_user.username}} </h4>
                            </div>
                        {% endif %}
                    </div>
                </div>
            </div>
        </header>
    {% endblock header %}

    <# <a href="{{ url_for('deconnexion') }}" type="button"
           class="btn btn-outline-danger rounded-1 border-2
fw-semibold px-2 py-1">Déconnexion</a> #>
    <!-- creation menu / bouton qui permet le défilement -->
        <div class="dropdown text-end">
            <a href="{{ url_for('main.profile') }}" class=
"curseur-avatar d-block link-dark text-decoration-none dropdown-toggle"
data-bs-toggle="dropdown" aria-expanded=
"false">
                <img src=
"{{url_for('main.profile_picture', username=current_user.username) }}"
alt="mdo" width="36" height="36" class=
"avatar-user rounded-3">
            </a>
            <ul class=
"dropdown-content dropdown-menu text-small" style="">
                <!-- contenu du menu / ce qui défile -->
                <# lien vers profil user #>
                <li>
                    <a class="dropdown-item" href=
"{{ url_for('main.profile') }}">
                        <span>Mon profil</span>
                        
                    </a>
                </li>
            </ul>
        </div>
    </div>

```

```

        {# lien vers liste des rooms #}
        <li>
            <a class="dropdown-item" href=
"{{ url_for('main.liste_room') }}">
                <span>Rooms</span>
                
            </a>
        </li>

{# si user connecté en admin --> liens vers panel admin + edit room#}
    {% if current_user.is_admin == true %}
        {# lien vers panel administrateur #}
        <li>
            <a class="dropdown-item" href=
"{{ url_for('admin.index') }}">
                <span>Panel admin</span>
                
            </a>
        </li>
    {% endif%}

        <li>
            <hr class="dropdown-divider">
<!-- simple séparation estétique -->
        </li>
        <li>
            <a class="dropdown-item" href=
"{{ url_for('main.deconnexion') }}">
                <span>Se déconnecter</span>
                
            </a>
        </li>
    </ul>
</div>
{% else %}

<!-- <button type="button" class="btn btn-outline-light me-2"><a href="c
onexion.html" text-decoration-none>J'ai déjà un compte</a></button> -->
    <a href="{{ url_for('main.connexion') }}" class=
"btn rounded-1 border-2 active text-info fw-semibold px-2 py-1" role=
"button"
        aria-pressed="true">J'ai déjà un compte</a>
    <a class="a-rootme-pro" href=
"http://www.lyceebranly.com/">
        <img class="logo-rootme-pro" src=
"/static/img/university.svg" alt="Logo d'une université"
            title="Lycée Edouard Branly">
    </a>
    {% endif %}
</div>
</div>
</div>
</header>
{% endblock %}

```

```
<!-- ----- layout des flash : succès, error, ... ----- -->
<!-- icons SVG flashing -->
<svg xmlns="http://www.w3.org/2000/svg" style="display: none;">
    <!-- icon validée -->
    <symbol id="check-circle-fill" fill="currentColor" viewBox="0 0 16 16">
        <path
            d=
            "M16 8A8 8 0 1 1 0 8a8 8 0 0 1 16 0zm-3.97-3.03a.75.75 0 0 0-1.0
            8.022L7.477 9.417 5.384 7.323a.75.75 0 0 0-1.06 1.06L6.97 11.03a.
            75.75 0 0 0 1.079-.0213.992-4.99a.75.75 0 0 0-.01-1.05z"
        />
        </symbol>
        <!-- icon infos -->
        <symbol id="info-fill" fill="currentColor" viewBox="0 0 16 16">
            <path
                d=
                "M8 16A8 8 0 1 0 8 0a8 8 0 0 0 0 16zm.93-9.412-1 4.705c-.07.34.02
                9.533.304.533.194 0 .487-.07.686-.246l-.088.416c-.287.346-.92.598
                -1.465.598-.703 0-1.002-.422-.808-1.319l.738-3.468c.064-.293.006
                -.399-.287-.471-.451-.081.082-.381 2.29-.287zM8 5.5a1 1 0 1 1 0-2
                1 1 0 0 1 0 2z"
            />
            </symbol>
            <!-- icon triangle exclamation -->
            <symbol id="exclamation-triangle-fill" fill="currentColor"
viewBox="0 0 16 16">
                <path
                    d=
                    "M8.982 1.566a1.13 1.13 0 0 0-1.96 0L.165 13.233c-.457.778.091 1.
                    767.98 1.767h13.713c.889 0 1.438-.99.98-1.767L8.982 1.566zM8 5c.5
                    35 0 .954.462.9.995l-.35 3.507a.552.552 0 0 1-1.1 0L7.1 5.995A.90
                    5.905 0 0 1 8 5zm.002 6a1 1 0 1 1 0 2 1 1 0 0 1 0-2z"
                />
                </symbol>
            </svg>
```

```

{% block footer %}
<footer class=
"page-footer container-fluid pt-3 align-items-center">
    <div class="d-flex justify-content-center">
        
    </div>
    <ul class="nav justify-content-center">
        <li class="nav-item"><a href=
 "{{config.CONFIDENTIALITE_LINK}}"
            class=
"nav-link px-2 text-muted text-secondary-emphasis fs-6">
Confidentialité</a></li>
        <!-- <li class="nav-item ">&ampnbsp|</li> -->
        <li class="nav-item"><a href=
 "{{config.MENTIONS_LEGALES_LINK}}"
            class="nav-link px-2 text-secondary-emphasis fs-6">
Mentions Légales</a></li>
        <li class="nav-item"><a href=
 "{{config.CONDITION_GENERALES_D_UTILISATION_LINK}}"
            class="nav-link px-2 text-secondary-emphasis fs-6">
Conditions Générales
            d'Utilisation</a></li>
    </ul>
    <p class="text-center p-0 m-0 text-white fs-6">
{{config.SENTENCE_FOOTER_END}}</p>
    <p class=
"text-center border-bottom border-secondary pb-3 mb-0 text-white f
s-6"
>{{config.COPYRIGHT}}</p>

    <ul class="nav justify-content-center bg-black pt-3">

{# TODO: Faire un test que les logos se cachent bien si l'URL n'ex
iste pas #}
    {% if config.TWITTER_LINK %}
        <li class="nav-item px-3">
            <a href="{{config.TWITTER_LINK}}" class=
"nav-link px-2">
                
            </a>
        </li>
    {% endif %}

```

```
{% if config.LINKEDIN_LINK %}  
    <li class="nav-item px-3">  
        <a href="{{config.LINKEDIN_LINK}}" class=  
"nav-link px-2">  
              
        </a>  
    </li>  
    {% endif %}  
  
{% if config.RSS_FEED_LINK %}  
    <li class="nav-item px-3">  
        <a href="{{config.RSS_FEED_LINK}}" class=  
"nav-link px-2">  
              
        </a>  
    </li>  
    {% endif %}  
  
{% if config.DISCORD_LINK %}  
    <li class="nav-item px-3">  
        <a href="{{config.DISCORD_LINK}}" class=  
"nav-link px-2">  
              
        </a>  
    </li>  
    {% endif %}  
</ul>  
  
</footer>  
{% endblock %}  
  
</html>
```

```

<!-- ROUGE alert warning -->
{% with messages = get_flashed_messages(with_categories=true
) %}
  {% if messages %}
    {% for category, message in messages %}
      {%# VERT - success #}
      {% if category == 'success' %}
        <div class=
"flashingtest alert alert-success alert-dismissible fade show"
role="alert">
          <svg class="bi flex-shrink-0 me-2" width="24" height="24"
role="img" aria-label="Success:">
            <use xlink:href="#check-circle-fill" />
          </svg>
          {{ message }}
          <button type="button" class="btn-close" data-bs-dismiss=
"alert" aria-label="Close"></button>
        </div>
      {% endif %}
      {%# ROUGE - warning #}
      {% if category == 'error' %}
        <div class=
"flashingtest alert alert-danger alert-dismissible fade show"
role="alert">
          <svg class="bi flex-shrink-0 me-2" width="24" height="24"
role="img" aria-label="Danger:">
            <use xlink:href="#exclamation-triangle-fill" />
          </svg>
          {{ message }}
          <button type="button" class="btn-close" data-bs-dismiss=
"alert" aria-label="Close"></button>
        </div>
      {% endif %}
      {%# BLEU - info #}
      {% if category == 'info' %}
        <div class=
"flashingtest alert alert-info alert-dismissible fade show" role=
"alert">
          <svg class="bi flex-shrink-0 me-2" width="24" height="24"
role="img" aria-label="Info:">
            <use xlink:href="#info-fill" />
          </svg>
          {{ message }}
          <button type="button" class="btn-close" data-bs-dismiss=
"alert" aria-label="Close"></button>
        </div>
      {% endif %}
      {% endfor %}
      {% endif %}
      {% endwith %}

      {% block content %}<h1>THIS IS DEFAULT TEXT, CHANGE ME</h1>
    {% endblock %}
  </body>

```