

**BTS CIEL SESSION 2025 - IR**

**E6 – PROJET TECHNIQUE**

**Dossier Commun**

Lycée : Touchard-Washington		Ville : LE MANS	
Nom du projet :		Simulation de pilotage de drone	
N° du projet :		TW1	

Projet nouveau :	Oui <input checked="" type="checkbox"/> Non <input type="checkbox"/>	Projet interne :	Oui <input checked="" type="checkbox"/> Non <input type="checkbox"/>
Délai de réalisation :	150 heures	Statut des étudiants :	Formation initiale <input checked="" type="checkbox"/> Apprentissage <input type="checkbox"/>
Spécialité des étudiants :	ER <input type="checkbox"/> IR <input checked="" type="checkbox"/> Mixte <input type="checkbox"/>	Nombre d'étudiants :	4 étudiants
Professeurs responsables :	Didier BERNARD, Jilali KHAMLACH		



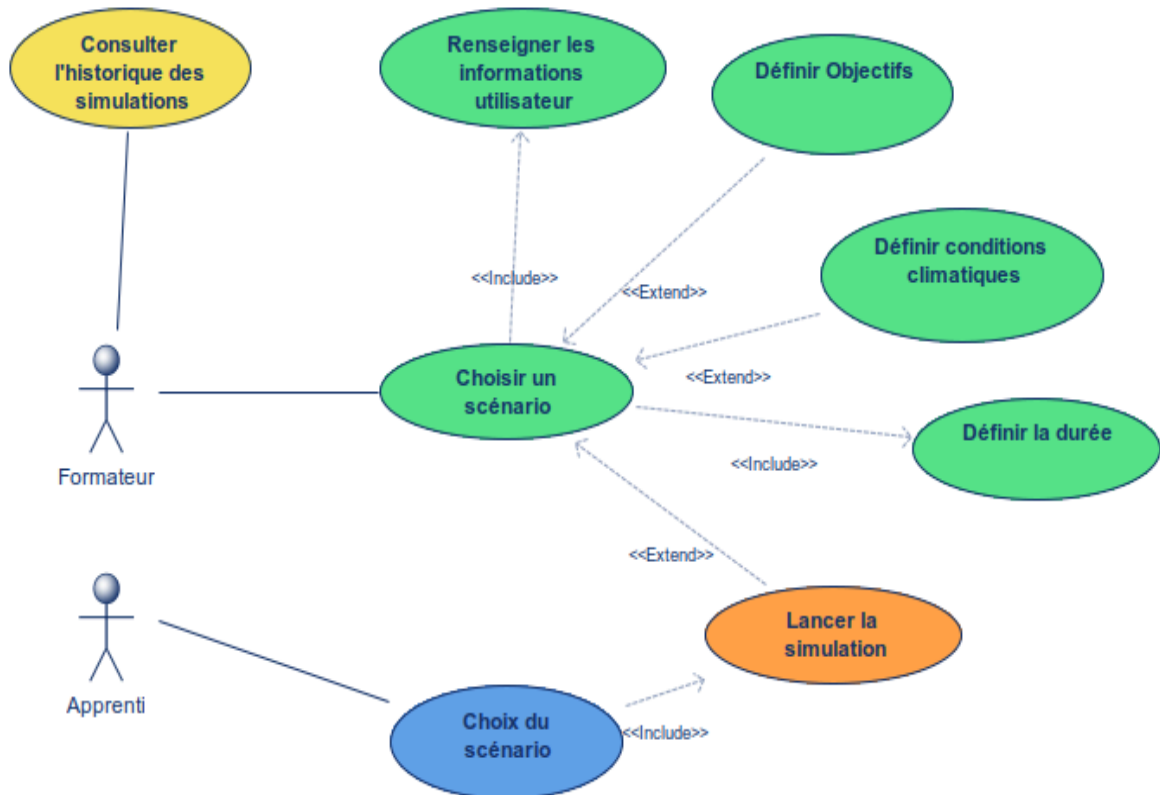
## ***Simulation de Pilotage de Drone VR***

CHARTIER Noé - BANDO Benjamin - VILLATTE Alexis - GERARD Lenny

**Sommaire :**

<b>1.1 Diagramme de Cas d'utilisation.....</b>	<b>3</b>
<b>1.1.1 Définir la durée :.....</b>	<b>4</b>
<b>1.1.2 Définir les conditions climatiques :.....</b>	<b>4</b>
<b>1.1.3 Renseigner les informations utilisateur :.....</b>	<b>5</b>
<b>1.1.4 Choisir un scénario :.....</b>	<b>6</b>
<b>1.1.5 Interface formateur.....</b>	<b>7</b>
<b>1.2 Choix technologiques.....</b>	<b>8</b>
<b>1.3 Représentation schématique de la base de données.....</b>	<b>9</b>
<b>1.4 Github du projet.....</b>	<b>10</b>
<b>1.5 Diagrammes de séquence (ajout des eleve dans la base de données).....</b>	<b>11</b>
<b>1.5.1 Diagramme de classes.....</b>	<b>12</b>
<b>1.6 Tests unitaires réalisés.....</b>	<b>13</b>
<b>1.7 Diagramme de GANTT.....</b>	<b>14</b>
<b>1.8 Conclusion.....</b>	<b>15</b>

## 1.1 Diagramme de Cas d'utilisation



*Figure 1 : Diagramme de Cas d'utilisation prévisionnel*

La partie verte du diagramme (noé) décrit les étapes que doit suivre un formateur pour configurer un scénario de simulation, incluant la définition de la durée, des conditions climatiques, des informations utilisateur et le choix final du scénario. Chaque action nécessite que le formateur soit connecté et que l'application soit fonctionnelle. À chaque étape, les données saisies (durée, météo, informations sur l'utilisateur) sont enregistrées dans une base de données et associées au scénario. Une confirmation visuelle valide chaque saisie réussie, tandis que des messages d'erreur apparaissent en cas de données invalides, de champs manquants ou de problèmes techniques (ex. : déconnexion ou panne du système). Une fois toutes les étapes complètes, le scénario est prêt à être lancé, avec un récapitulatif des paramètres affichés pour validation finale.

### 1.1.1 Définir la durée :

<b>Pré-conditions :</b>	<ul style="list-style-type: none"> <li>- Le formateur doit être connecté au système.</li> <li>- L'application doit être fonctionnelle et afficher une interface dédiée à la gestion de la durée.</li> </ul>
<b>Post-conditions :</b>	<ul style="list-style-type: none"> <li>- La durée est enregistrée et associée au scénario en cours pour la prochaine simulation.</li> <li>- Une confirmation visuelle qui correspond à un message de succès affiché sur l'interface formatrice via un modal.</li> </ul>
<b>Scénario principal :</b>	<ul style="list-style-type: none"> <li>- Il entre la durée souhaitée en heures ou minutes.</li> <li>- Le formateur valide les modifications.</li> <li>- Une notification de succès est affichée.</li> </ul>
<b>Alternatives :</b>	<ul style="list-style-type: none"> <li>- Alternative 1 : Si la durée entrée (5 min) n'est pas valide (exemple : un champ vide ou une durée négative ou une durée maximum à 5 min), un message d'erreur s'affiche et invite l'utilisateur à corriger la saisie via un message dans le label.</li> <li>- Alternative 2 : Si la connexion au serveur est interrompue, les données ne sont pas sauvegardées, et un message d'avertissement s'affiche pour signaler l'erreur.</li> </ul>

### 1.1.2 Définir les conditions climatiques :

<b>Pré-conditions :</b>	<ul style="list-style-type: none"> <li>- Le formateur doit être connecté au système et avoir sélectionné un scénario à configurer.</li> </ul>
<b>Post-conditions :</b>	<ul style="list-style-type: none"> <li>- Les conditions climatiques sont sauvegardées dans une base de données et les conditions sont prêtes à être appliquées lors de la simulation.</li> <li>- Un récapitulatif des conditions climatiques est affiché sur la page de configuration sous forme d'un tableau.</li> </ul>
<b>Scénario principal :</b>	<ul style="list-style-type: none"> <li>- Il définit les différents paramètres météorologiques (force du vent, intensité de la pluie, température) en sélectionnant les champs prérequis.</li> <li>- Le formateur valide les paramètres sélectionnés.</li> <li>- Une notification de succès s'affiche à l'écran via un modal.</li> </ul>
<b>Alternatives :</b>	<ul style="list-style-type: none"> <li>- Alternative 1 : Si certains paramètres météorologiques ne sont pas définis (ex. : température manquante), l'application utilise des valeurs par défaut et avertit le formateur.</li> <li>- Alternative 2 : Si la base de données rencontre une erreur ou si le système est hors ligne, les conditions ne sont pas sauvegardées. Un message d'erreur s'affiche pour demander une nouvelle tentative.</li> </ul>

---

### **1.1.3 Renseigner les informations utilisateur :**

<b>Pré-conditions :</b>	<ul style="list-style-type: none"><li>- Le formateur doit disposer des droits nécessaires pour accéder à la page dédiée aux informations de l'utilisateur.</li><li>- L'application doit être fonctionnelle, et les champs obligatoires doivent être disponibles pour la saisie.</li></ul>
<b>Post-conditions :</b>	<ul style="list-style-type: none"><li>- Les données utilisateur (nom, rôle, niveau, etc.) sont enregistrées dans la base de données et associées à la simulation.</li><li>- Une notification de succès ou un récapitulatif des informations enregistrées est affiché.</li></ul>
<b>Scénario principal :</b>	<ul style="list-style-type: none"><li>- Le formateur accède à la page ou au module "Informations utilisateur".</li><li>- Il entre les données nécessaires, comme le nom, le prénom, le niveau d'expérience ou d'autres caractéristiques liées à l'apprenti.</li><li>- Le formateur valide la saisie.</li><li>- Le système sauvegarde les informations de l'utilisateur et les associe au scénario configuré.</li><li>- Une confirmation visuelle ou un message de validation s'affiche à l'écran.</li></ul>
<b>Alternatives :</b>	<ul style="list-style-type: none"><li>- Alternative 1 : Si des informations obligatoires sont manquantes, un message d'erreur demande au formateur de compléter les champs avant de valider.</li><li>- Alternative 2 : Si la base de données rencontre une erreur (saturation ou panne), les informations ne sont pas enregistrées, et un message d'avertissement invite à réessayer plus tard.</li></ul>


### 1.1.4 Choisir un scénario :

<b>Pré-conditions :</b>	<ul style="list-style-type: none"> <li>- L'application doit être lancée et accessible par Le formateur.</li> <li>- Le formateur doit avoir configuré les paramètres nécessaires (durée, conditions climatiques, informations utilisateur).</li> </ul>
<b>Post-conditions :</b>	<ul style="list-style-type: none"> <li>- Le scénario est entièrement configuré et prêt à être exécuté.</li> <li>- Une vue récapitulative des paramètres du scénario est affichée pour validation finale.</li> </ul>
<b>Scénario principal :</b>	<ul style="list-style-type: none"> <li>- Le formateur accède à l'interface dédiée à la gestion des scénarios.</li> <li>- Il sélectionne un scénario préexistant ou en crée un nouveau.</li> <li>- Le formateur paramètre les options du scénario (durée, conditions climatiques, informations utilisateur).</li> <li>- Il valide les modifications ou la création du scénario.</li> <li>- Le système sauvegarde le scénario complet dans la base de données et affiche une confirmation.</li> </ul>
<b>Alternatives :</b>	<ul style="list-style-type: none"> <li>- Alternative 1 : Si une étape de paramétrage est incomplète (ex. : absence de durée ou d'informations utilisateur), un message d'erreur avertit le formateur et lui propose de corriger les erreurs.</li> <li>- Alternative 2 : Si le système rencontre une panne ou un problème de connexion, les modifications ne sont pas sauvegardées. Un message d'avertissement s'affiche, indiquant que la configuration du scénario doit être relancée une fois le problème résolu.</li> </ul>


### 1.1.5 Interface formateur

## Configuration de l'exercice

### 1. Sélectionner le lieu de l'exercice




Interieur




Exterieur


### 2. Choix du modele du drone



Assisté



Assisté



Non Assisté

### 3. Durée du vol

Objectifs



### 4. Conditions météo

Vent: 68km/h

Température: 9°C

Eleves

### 6. Pluie



Lancer simulation

*Figure 2 : Interface formateur prévisionnel*

Cette interface de configuration a été conçue pour offrir au formateur une grande flexibilité dans la préparation d'un exercice de simulation de vol de drone. Elle permet de définir avec précision les différents paramètres nécessaires avant le lancement de la simulation. Le formateur peut ainsi choisir le lieu de l'exercice, sélectionner le modèle de drone à utiliser, déterminer la durée de la session, ainsi que définir les objectifs à atteindre. Ces objectifs sont présentés sous forme de liste déroulante dynamique, avec la possibilité d'ajouter ou de retirer librement ceux qui sont jugés pertinents pour la séance. Il peut également configurer les conditions météorologiques, en ajustant des paramètres comme la force du vent, la température ambiante, et la présence ou non de pluie, afin de rendre la simulation plus réaliste. Enfin, le formateur a la possibilité d'assigner l'exercice à un élève spécifique en le recherchant par sa classe, garantissant ainsi un suivi personnalisé et structuré de la progression de chaque apprenant. L'ensemble de ces fonctionnalités vise à offrir une expérience complète, intuitive et adaptée aux besoins pédagogiques en matière de pilotage de drone.

## 1.2 Choix technologiques

### Outils utilisés :

- **Unreal Engine 5** : moteur de jeu 3D utilisé pour concevoir l'environnement de simulation, gérer les graphismes, la physique et l'interactivité.
- **C++** : langage de programmation principal utilisé avec Unreal Engine pour développer la logique de la simulation, les comportements des drones, et les interactions utilisateur.
- **MySQL** : système de gestion de base de données relationnelle pour stocker les données des utilisateurs, l'historique des simulations, les scénarios personnalisés, etc.

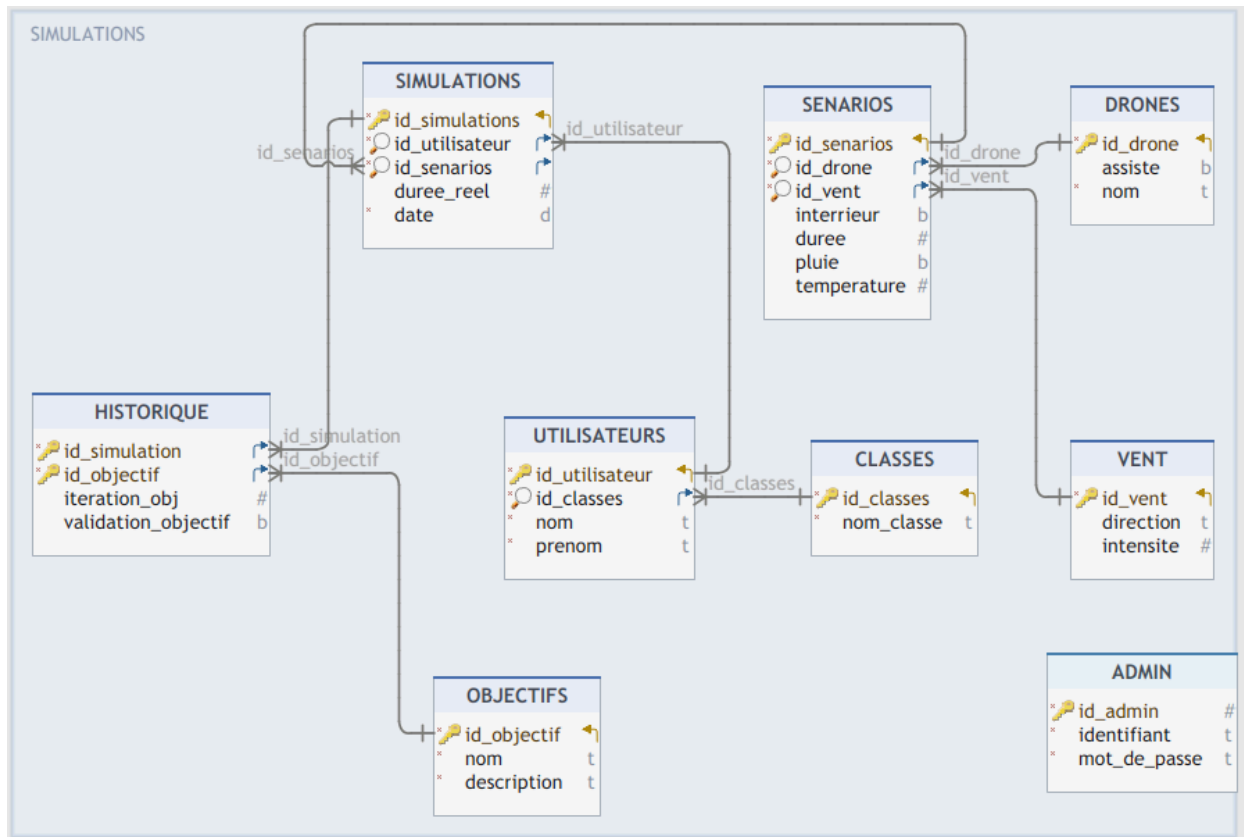
### Justification des choix :

- **Unreal Engine** a été choisi pour sa puissance graphique, sa prise en charge avancée de la physique en temps réel (idéale pour simuler le vol de drones), sa compatibilité avec la VR/AR et sa communauté active.
- **C++** s'intègre parfaitement à Unreal Engine et permet un contrôle bas niveau des performances, indispensable pour les calculs liés à la physique du vol et aux interactions complexes ainsi que l'ajout dans la base de données.
- **MySQL** a été sélectionné pour sa fiabilité, sa simplicité d'intégration avec d'autres technologies, et sa capacité à gérer efficacement des données structurées comme les profils utilisateurs, les paramètres des exercices et les résultats.

Pour la persistance des données, des bibliothèques C++ comme MySQL Connector/C++ ou des plugins spécifiques Unreal Engine sont utilisés pour établir la connexion avec la base de données MySQL. Cela permet à la simulation de récupérer dynamiquement les informations des utilisateurs, de charger des scénarios enregistrés, et de sauvegarder les résultats d'exercices ou les paramètres personnalisés en fin de session.



## 1.3 Représentation schématique de la base de données



*Figure 3 : Base de données prévisionnel*

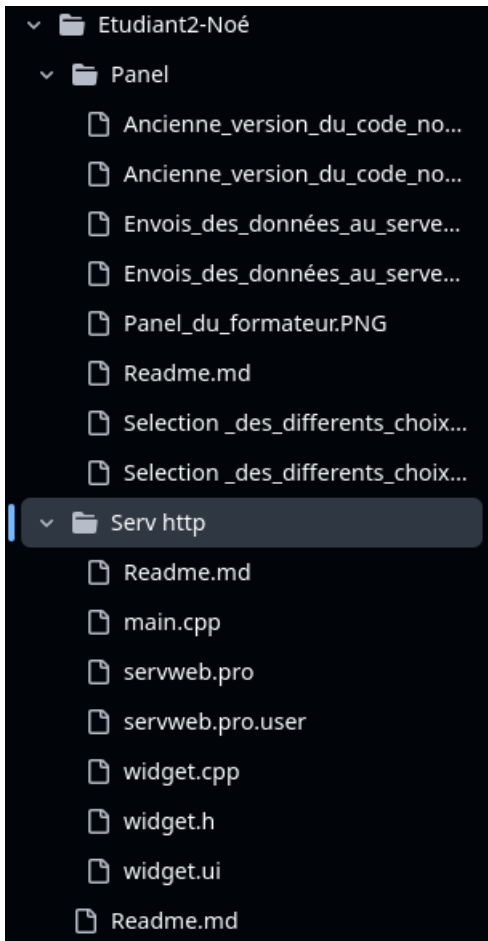
Ce schéma représente la structure complète de la base de données dédiée à la gestion des simulations de drones dans un cadre pédagogique. Chaque simulation est initiée par un apprenti, lui-même rattaché à une classe spécifique, ce qui permet de structurer les sessions d'entraînement par groupe et d'assurer un suivi collectif et individuel. Les simulations sont basées sur des scénarios prédéfinis, qui intègrent un ensemble de paramètres configurables tels que les conditions météorologiques (force du vent, intensité de la pluie, température ambiante), mais aussi le modèle précis du drone utilisé, ce qui permet de varier les contextes d'exercice et de renforcer la polyvalence des apprentis. Au cours de chaque simulation, les apprenants doivent accomplir un ou plusieurs objectifs bien définis, chacun étant associé à un type ou une catégorie, facilitant leur classification, leur analyse et leur réutilisation dans différents scénarios. Les performances de l'apprenti, ainsi que le degré de réussite des objectifs, sont systématiquement enregistrés dans un historique détaillé, permettant un suivi longitudinal des progrès. Ce modèle relationnel offre donc une vue complète et centralisée des activités de formation, favorisant à la fois l'individualisation des parcours et l'optimisation continue des scénarios de simulation en fonction des compétences observées.

## 1.4 Github du projet



Name	Last commit message	Last commit date
..		
Panel	Update Readme.md	17 minutes ago
Serv http	Update Readme.md	33 minutes ago
Readme.md	Update Readme.md	48 minutes ago

Figure 4 : Github prévisionnel

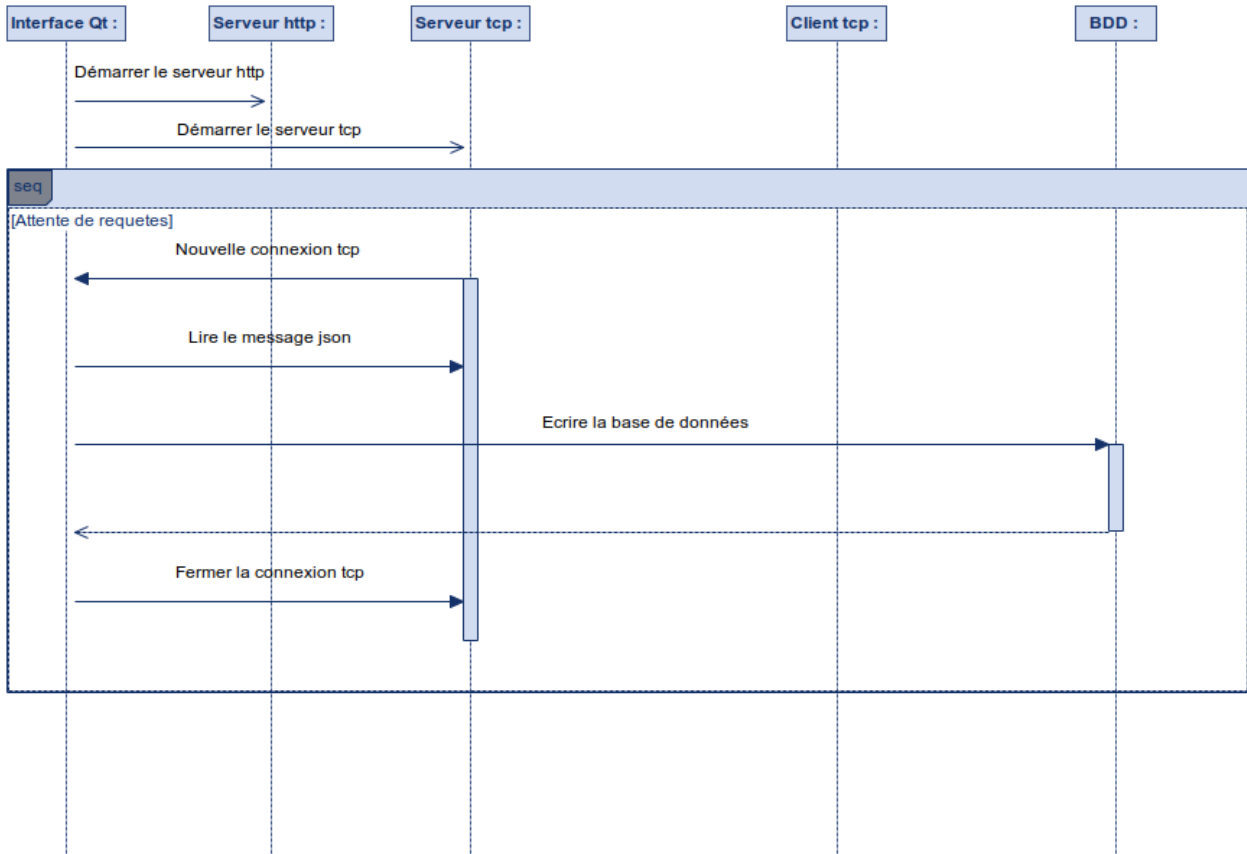


GitHub est une plateforme de développement collaboratif basée sur Git, un système de contrôle de version. Elle permet aux développeurs de stocker, gérer et suivre les modifications apportées à leur code source, tout en facilitant la collaboration entre membres d'une équipe. GitHub offre des fonctionnalités comme les *pull requests*, les *issues*, et les *branches*, qui aident à organiser le travail, à réviser le code et à intégrer des changements de manière structurée. Utiliser GitHub est essentiel pour tout projet logiciel, car il assure une traçabilité des modifications, une sauvegarde centralisée du code et une meilleure coordination entre les contributeurs.

L'image montre les fichiers et dossiers organisés en deux sections principales : *Panel* et *Serv http*. Les fichiers impliquent un serveur HTTP (*Serv http*) et une interface formateur (*Panel*), avec des fichiers typiques comme des sources C++ (.cpp, .h), des configurations de projet (.pro), et des ressources d'interface (.ui).

Figure 5 : Github

## 1.5 Diagrammes de séquence (ajout des eleve dans la base de données)



*Figure 6 : Diagrammes de séquence prévisionnel*

Ce diagramme de séquence décrit le processus d'ajout d'élèves dans une base de données. Tout commence lorsque l'interface Qt démarre le serveur HTTP, qui, à son tour, lance le serveur TCP. Une fois en marche, le serveur HTTP entre dans une phase d'attente de requêtes. Lorsqu'un client HTTP établit une nouvelle connexion, le serveur HTTP lit le message reçu, qui est au format JSON. Ce message contient les informations des élèves à enregistrer. Le serveur traite ensuite ces données et les écrit dans la base de données. Une fois l'opération terminée, il ferme la connexion HTTP avec le client. Ce mécanisme permet ainsi d'ajouter des élèves dans la base de manière structurée et automatisée.

### 1.5.1 Diagramme de classes

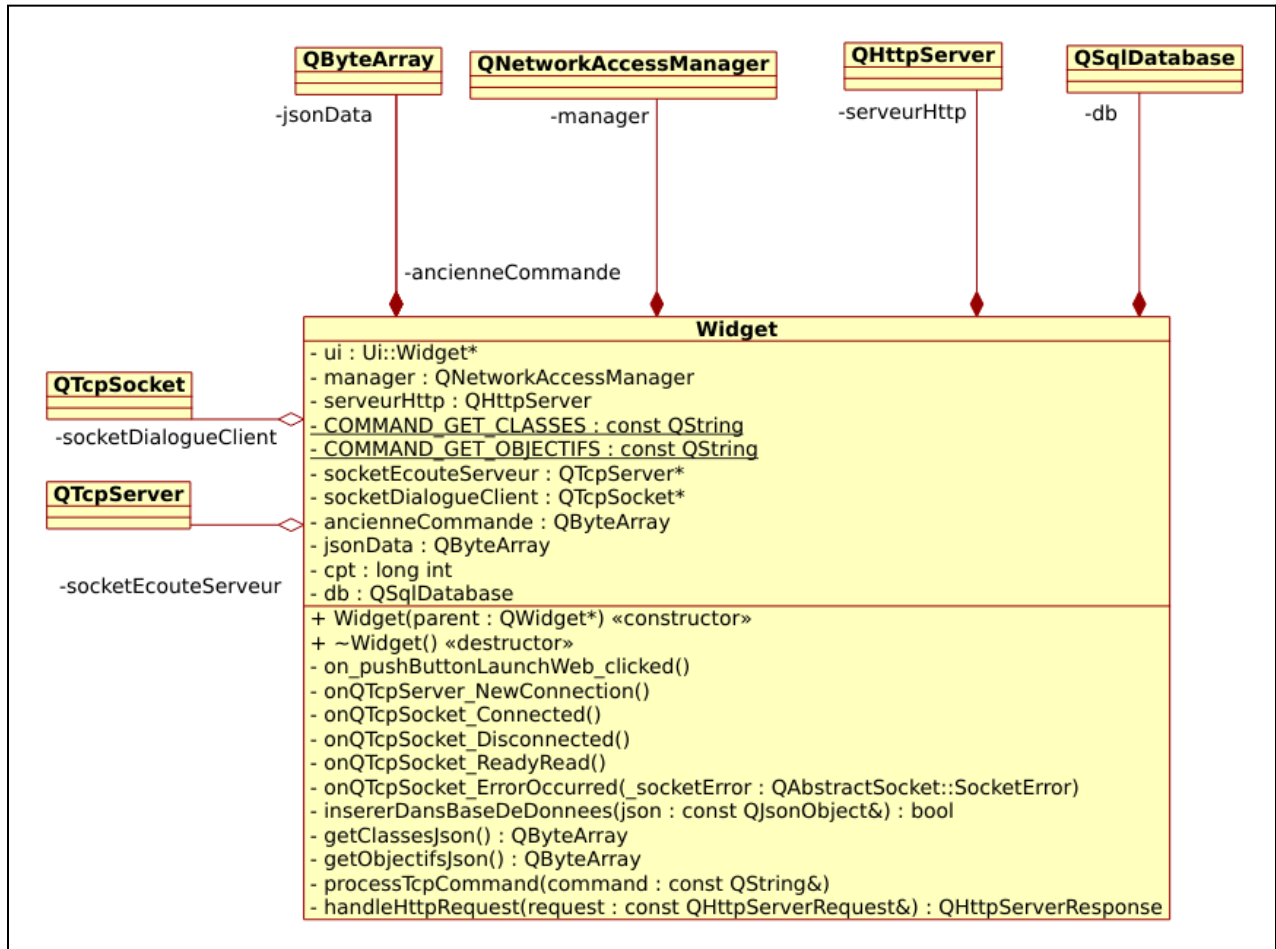


Figure 7 : Diagramme de classes prévisionnel

Ce diagramme met en évidence la manière dont l'application insère des données dans la base de données à travers la classe principale **Widget**. Plus précisément, la méthode **insererDansBaseDeDonnees(json : const QJsonObject&) : bool** est chargée de cette opération. Elle reçoit un objet JSON, contenant les informations à enregistrer (par exemple des données d'élèves), puis les traite pour effectuer l'insertion dans la base via l'attribut **db** de type **QSqlDatabase**. Cette méthode est probablement appelée après la réception et la lecture d'un message JSON à travers un socket TCP, ce qui lie directement la communication réseau à la mise à jour de la base. Elle constitue donc un point clé de l'architecture, assurant la persistance des données reçues dans le système.

## 1.6 Tests unitaires réalisés

Fiche de tests			
Nature :	Fonctionnel	Référence :	tw1 – etu2 – 001
Module :	Interface_formateur	Auteur :	Noé Chartier
Date de création / mise à jour :		11/03/2025	
Objectif :	Vérifier que les informations sélectionné par le formateur via l'interface formateur sont correctement envoyer en json sur l'adresse 172.18.58.93:8080 et dans la base de données.		
Condition du test			
État initial du module :		Environnement du test :	
Ordinateur	PC	OS Windows 10	
Programme	Projet_Drone_2025	Unreal engine 5.4.4	
Base de données (si utile pour le test)	172.18.58.7 Projet_Drone_2025 snir snir	La base de données contient des tables qui sont les suivante : objectif ; classe	
Procédure de test			
Lancement de l'application :		Lancement de la simulation, affichage de l'interface formateur	
Repère	Opérations	Résultats attendus	
1	Cliquer sur le simulateur	Ouverture de l'application	
2	Cliquer sur les option pour commencer la simulation avec : map extérieur drone avec assistance pas de pluie 15 km/h de vent 20°C de température 5 min de vol Jean BTS ciel Entrer dans le cercle 2 fois Cliquer sur valider	{ "Scenario" : "exterieur", "Drone" : "1", "Pluie" : "0", "Vent" : "15", "Celsius" : "20", "Duree" : "5", "Utilisateur" : "Jean", "Classe" : "BTS_ciel", "Objectif" : "cercle", "iteration_obj" : "2", }	
3	Cliquer sur les option pour commencer la simulation avec la pluie en plus : map intérieur drone sans assistance avec de la pluie 55 km/h de vent -3°C de température 5 min de vol Tom BTS snir Entrer dans le cercle 8 fois	{ "Scenario" : "intérieur", "Drone" : "0", "Pluie" : "1", "Vent" : "55", "Celsius" : "-3", "Duree" : "5", "Utilisateur" : "Tom", "Classe" : "BTS snir", "Objectif" : "cercle", "iteration_obj" : "8", }	

Figure 8 : Tests unitaires prévisionnel

Le test unitaire présenté vise à vérifier que les informations saisies par le formateur via l'interface sont correctement envoyées en JSON à un serveur distant et insérées dans la base de données. Deux scénarios ont été testés, l'un avec des conditions de simulation extérieures modérées, l'autre avec des conditions intérieures extrêmes, chacun générant un objet JSON bien structuré. Toutefois, une anomalie a été relevée : l'absence de message d'erreur en cas de saisie invalide. De plus, aucun contrôle n'est prévu pour confirmer que les données ont bien été insérées en base, ce qui affaiblit la validité du test.

## 1.7 Diagramme de GANTT

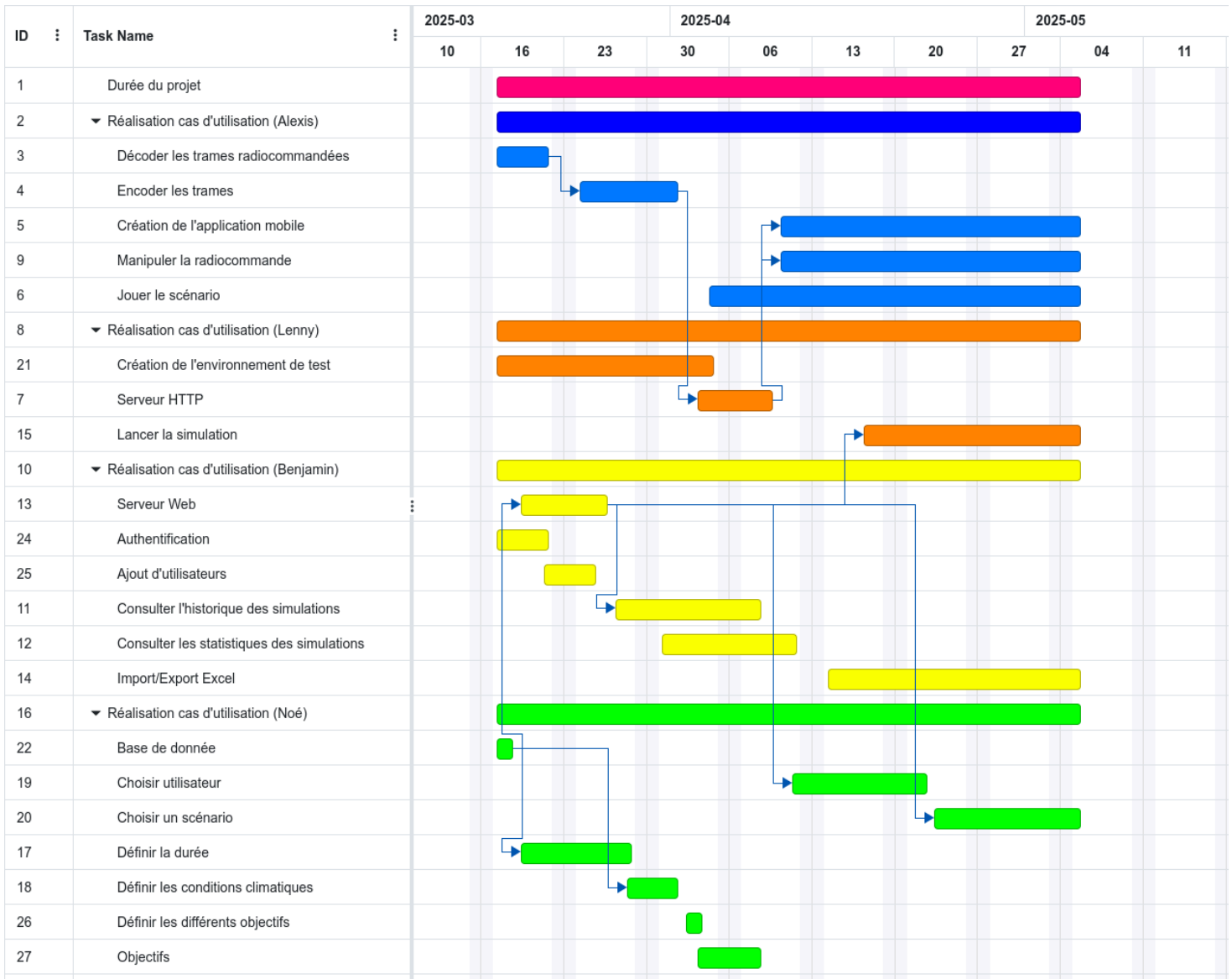


Figure 9 : Diagramme de GANTT prévisionnel

Voici le diagramme de GANTT prévisionnel de notre projet.

Certaines tâches ont pris du retard, comme par exemple avec le serveur HTTP, ou encore avec la création du menu.

Ce qui me fait dire que notre projet a pris du retard et donc que ce diagramme de GANTT n'a pas pu être suivi correctement.

## **1.8 Conclusion**

- **Bilan sur la planification personnelle :**

La planification personnelle a été dans l'ensemble respectée, avec un bon suivi des différentes phases du projet. Certaines tâches ont toutefois nécessité plus de temps que prévu, notamment les objectifs ainsi que les élèves choisis. L'organisation initiale m'a permis de garder une vision claire de l'avancement du projet.

- **Ce qui a été réalisé par rapport au planning initial :**

Les objectifs principaux ont été atteints, notamment le développement du menu formateur, la mise en place de la communication réseau, et l'envoi des données vers la base de données.

- **Éventuels retards ou réajustements et leur cause :**

Des retards ont été rencontrés durant l'ajout des objectifs et des élèves, principalement à cause d'une complexité imprévue. Ces retards ont été compensés par le report de certaines tâches secondaires.

- **Apport personnel du projet :**

Ce projet m'a permis de m'impliquer pleinement dans une démarche de développement complet, de la conception à la mise en production. J'ai également appris à travailler de manière autonome et à m'adapter face aux imprévus. J'ai particulièrement apprécié le développement du menu formateur pour son aspect blueprint.

- **Compétences acquises :**

Au cours de ce projet, j'ai développé mes compétences en C++, dans la manipulation de base de données, les différents protocoles réseau, la gestion d'un projet, ainsi que mes capacités à résoudre des problèmes, à organiser un projet, à communiquer efficacement en équipe.

- **Difficultés rencontrées et solutions apportées :**

Les principales difficultés ont concerné la validation des données, la communication client-serveur. Pour y remédier, j'ai mis en place des tests ciblés, des outils de debug, des validations côté serveur. Cela m'a permis de gagner en rigueur et en efficacité.

- **Perspectives d'amélioration :**

Pour améliorer ce projet, il serait intéressant d'ajouter un menu formateur mobile. Un approfondissement sur l'ajout des informations dans la base de données pourrait également apporter une plus-value.