Rapport Manager - Hackathon Junior 2025

Équipe :CY TeC

Thème de l'Univers : La purge

---

1. Présentation

1.1 L'Univers du Guide de Survie

Notre application "Guide de Survie" a été conçue pour aider les utilisateurs à naviguer dans un environnement [décrire votre thème : ex. hostile et radioactif]. L'objectif est de fournir des outils vitaux : localisation, itinéraire sécurisé et approvisionnement en ressources.

1.2 Choix Techniques

- **Langage :** Python.
    - *Justification :* Nous avons suivi la recommandation du sujet car c'est le langage le plus adapté pour une prise en main rapide par l'équipe et il dispose de bibliothèques robustes pour les calculs algorithmiques.
- **Architecture :** Programmation Orientée Objet (POO).
    - *Justification :* Imposée par le sujet, cette architecture permet de modéliser proprement nos entités (Lieux, Chemins) comme esquissé dans nos phases de conception.
- **Interface (Frontend) :** [Indiquer la lib utilisée, ex: Tkinter, Pygame ou Console].
    - *Justification :* Choix basé sur la rapidité de mise en place pour respecter la deadline de 18h00.

1.3 Bilan de Compétences

Au démarrage à 9h00, l'équipe possédait des niveaux hétérogènes. Grâce au "cours magistral" dispensé à 10h00 (voir planning), les développeurs ont acquis ou renforcé leurs compétences en :

- Structures de données complexes (Graphes).
- Logique algorithmique (Tri et Pathfinding).
- Architecture Backend/Frontend.

---

2. Calendriers

Conformément aux exigences, voici la gestion temporelle du projet.

2.1 Calendrier Prévisionnel (Basé sur le document de planification)

Ce planning a été établi et distribué à l'équipe dès l'arrivée des managers.

Heure	Phase	Tâches Principales
09h00	Lancement	Arrivée des managers, installation.
10h00	Formation	Introduction data, cours magistral, définition des structures de données.
11h00	Développement	Développement assisté, mise en place du "Full Backend".
12h00	Pause / Dev	Continuation du développement structurel.
14h00	UI/UX	Début de l'intégration de l'interface graphique.
15h00	Finalisation	Préparation du passage, débogage.
18h00	Rendu	Fin du Hackathon.

2.2 Calendrier de Suivi (Réel)

- **10h00 - 12h00 :** Respecté. La définition des classes (Lieux) et de l'énumération (Transports) a pris le temps prévu.
- **14h00 :** [Ajuster ici : ex. "Léger retard sur l'UI car l'algorithme de graphe a demandé plus de temps, mais rattrapé à 15h30"].
- **État final :** Le projet a été rendu dans les temps.

---

3. Schéma Technique (UML)

Voici les trois diagrammes modélisant notre application.

3.1 Diagramme de Classes

Notre structure de données principale repose sur les entités identifiées lors de la phase de réflexion :

- **Classe Lieux :** Possède les attributs Nom, Position et Liaisons (type de voie : ferrée ou route).
- **Classe Chemins :** Définit les arêtes du graphe avec Distance, Temps, Départ et Destination.
- **Enumération Transports :** Plutôt qu'une structure d'héritage complexe, nous avons opté pour une énumération (Enum) définissant les types de déplacements : VOITURE, TRAIN, PIED.
    - *Justification :* Cela simplifie le code en centralisant les constantes. Chaque valeur de l'énumération est associée à une vitesse spécifique utilisée par les algorithmes de calcul, sans avoir besoin d'instancier de multiples sous-classes.

3.2 Diagramme de Cas d'Utilisation (Use Case)

- **Acteur :** Survivant (Utilisateur).
- **Cas principaux :**
    - Consulter la carte.
    - Rechercher un itinéraire (avec ou sans étape).
    - Acheter un objet dans la boutique.
    - Recevoir une notification de danger.

3.3 Diagramme de Séquence (Calcul d'itinéraire)

1. L'utilisateur entre "Départ" et "Arrivée".
2. Le Système appelle le contrôleur de la Carte.
3. L'algorithme calcule le poids des arcs (Temps/Distance) selon le mode de transport (Enum).
4. Le Système retourne le chemin le plus court.

---

4. Fonctionnalités et Algorithmes

4.1 La Boutique (Algorithme de Tri)

- **Contexte :** Permettre l'achat de vivres et d'équipements triés par prix ou par nom.
- **Algorithme choisi :** Tri Fusion (Merge Sort) ou Tri Rapide (Quick Sort).
- **Justification :** Les tris par insertion/sélection étant interdits, nous avons opté pour un tri en $O(n \log n)$ pour garantir une fluidité même si le catalogue d'objets devient très grand (complexité optimale).

4.2 La Carte et Itinéraires (Algorithme de Graphe)

- **Contexte :** Une vingtaine de lieux interconnectés par des routes (voitures) ou rails (trains).

- **Algorithme choisi :** Algorithme de Dijkstra.
- **Justification :**
    - Nous avons modélisé la carte comme un graphe pondéré (Nœuds = Lieux, Arêtes = Chemins).
    - Dijkstra est idéal pour trouver le "plus court chemin" (en temps ou distance) dans un graphe aux poids positifs.
    - Il permet de gérer la contrainte du "lieu intermédiaire" en calculant deux segments : *Départ -> Intermédiaire* puis *Intermédiaire -> Arrivée*.

---

5. Formations

Pour assurer la réussite technique du projet, le manager a dispensé une formation accélérée (visible sur le planning à 10h00 "Cours magistrale") :

- **Sujets abordés :**
    - **Utilisation des Énumérations (Enum) :** Comment gérer efficacement les types de transports (Voiture, Train) et leurs constantes associées sans complexifier le code.
    - **Théorie des Graphes :** Explication des nœuds et des arêtes pour modéliser la carte.
    - **Versioning :** Utilisation de GitHub/Teams pour la collaboration.
- **Supports :** Schémas sur papier (voir annexe : brouillon UML) et démonstration de code au tableau (Live coding).