Mon retour d’expérience sur ce projet de Java, en montrant mon implication dans celui-ci, les difficultées et les problèmes que j’ai rencontrés (pour n’en citer que quelques-uns) :

J'ai commencé par mettre en place l'environnement de travail :

- Projet maven : J'ai créé le projet et le fichier pom.xml pour intégrer les dépendances. J'ai notamment ajouté Jackson (obligatoire pour lire le JSON de configuration) et JUnit pour les tests.
- Organisation des packages : structuration initiale du code en séparant les domaines (model, config, strategy, simulation, etc.) pour avoir une architecture claire, essentielle pour un « gros » projet. (je suis très perfectionniste donc il fallait que les choses soit organisées dès le départ.)

J’ai ensuite fait les prémices des classes principales :

- Le modèle de la tour : J'ai défini les classes Building et Floor avec par exemple s’assurer qu’il n’y ai aucun résident dans le hall d’accueil.
- Les acteurs et leurs habitudes : J'ai créé Resident et ResidentTripPlan. C’est ce qu’il fallait faire pour intégrer les "habitudes de vie" (travail, amis, etc.) demandées par la consigne. J'ai aussi implémenté le facteur probabiliste (les fourchettes horaires aléatoires) pour rendre le trafic réaliste.
- Les ascenseurs : J'ai posé les bases des classes Elevator et ElevatorRequest, intégrant les caractéristiques physiques comme la vitesse, l'accélération et le temps d'ouverture des portes, qui sont toutes paramétrables via la configuration.

On a garanti le chargement des paramètres :

- Mapping : J'ai créé les classes SimulationConfig pour que Jackson puisse facilement mapper le JSON vers des objets Java.
- Construction du Monde (ModelFactory) : Cette classe fait le pont. Elle prend la configuration lue et construit toutes les instances : la Building, les Resident et tous les Elevator. C'est elle qui garantit que le modèle est initialisé correctement avant de lancer le moteur.

Mon travail a soulevé des difficultés surtout au niveau de l'architecture et de la robustesse.

- Séparation stricte des couches : Le plus dur, c'était de bien séparer la logique de parsing JSON (config) de la logique métier (model). J'ai dû faire attention à ce que nos objets du modèle restent propres et ne dépendent jamais directement de Jackson,c'est ce qui fait la qualité du modèle objet exigé.

- Validation de la Config : Pour éviter les crashs, j'ai dû ajouter des contrôles dans le ModelFactory. C'est l'étape où on vérifie si les valeurs lues dans le JSON sont valides (par exemple, vérifier que l'étage de destination n'est pas hors bornes ou que la vitesseest positive). C'était nécessaire pour rendre la simulation fiable.

- Problèmes d'I/O au démarrage : J'ai eu des problèmes avec le chemin des ressources. Il fallait s'assurer que le fichier config/demo-config.json soit bien trouvé par le ConfigLoader une fois le projet packagé par Maven. Un problème banal parmi tant d'autres qui sont juste des détails qui bloque pendant une petite heure.

Au-delà de tout cela, ce projet était très enrichissant et abordait pas mal de contraintes qui ont nécessité beaucoup de questionnement et des tests pas toujours concluants. Un projet qui nous a fait évoluer autant sur le plan technique avec des connaissances acquises en java mais aussi sur le plan humain avec l'élaboration d'un travail à plusieurs, répartir celui-ci, nous mettre d'accord sur des solutions, des typos, et autres qui font partie du jeu de travail en équipe. J'ai apprécié cette expérience, même si concevoir tout ceci avec un agenda serré (des DS, projets, TP entremêlé pendant la période du projet) n'a pas garanti la mise au point de forcément tout ce que nous avions en tête aussi loufoque ai été l'idée (un UI avec visualisation des habitants et de l'immeuble pourquoi pas).