# Feedback première itération

* Mettre des petits **points** pour bien **délimiter** les **tronçons**, car deux tronçons différents peuvent être colinéaires
* Le **clic** sur des **intersections** doit donner le **nom** **des rues** qui partent de cette intersection
* Le **clic** (ou survol à la souris) sur les **tronçons** donne le **nom de la rue**
* Changer les icônes pour ne pas qu’ils se chevauchent
* Moins d’espace pour l’affichage des livraisons
* **Synchroniser** l’affichage des **infos** dans le **bandeau** **latéral**, le clic des infos de livraison met en évidence la livraison sur la carte, et vice-versa
* **Accélérer** **l’affichage** progressif de la **tournée**
* On peut afficher toute la tournée d’un coup, au **clic** sur un les **infos** du **bandeau latéral** met en **gras** le tronçon traversé par le livreur
* **Dégradé de couleur** pour simulé l’avancé de la tournée
* Les flèches servent pour le départ de l’entrepôt, mais rendra la carte un peu illisible
* **Optimiser** le calcul de la **tournée**
* **Temps** trop **long**, **ne pas annuler l’opération**, mettre un timeout de 5 sec, lancer l’algo pour la durée prévue, quand le timeout est atteint **ne pas écrire Timeout atteint**. Le système avait trouvé une solution mais elle n’est **pas assurée d’être optimale**
* Mettre un **message** **discret** si la solution est **optimale**
* Mettre un **message** **voyant** si la solution est **non optimale**
* **Proposer** à l’user de **calculer** une **solution** **optimale** en lui proposant combien de temps il est prêt à atteindre
* **Plus agréable** : l’algo a en permanence une solution, lui mettre un timeout très court (1sec), mettre un **thread** faisant tourner **l’appli** et un **autre** faisant tourner **l’algo**. Au bout d’un sec le thread système récupère la solution mais indique qu’elle n’est pas optimale et l’autre thread continue en background pour chercher une meilleure solution. Mettre un bouton arrêter l’optimisation pour stopper le thread si la solution nous va.
* **Aspect humain débriefing**
* **Code débriefing :** qu’est ce qui a été fait à l’arrache ?

## Commencer itérations suivantes (4 séances)

**Combien** d’itérations ? A planifier ti suite. Quelle durée ?

Faire un **planning** pour chaque **itération**. Réfléchir à tout ce qui faut faire (User story…)

Est-ce que dans le fichier XML il peut avoir des aberrations ? Oui ! S’il n’est pas correct l’application ne doit **pas planter.** On veut un message décrivant l’erreur (vitesse négative, tronçon sans nom de rue etc.).

Analyse des cas d’utilisation à aborder et à implémenter.

**Découper** ces **cas** **d’utilisation** en petit morceau afin **d’implémenter** plus **facilement**.

Typiquement, juste dire implémenter le CU modifier tournée n’est pas assez **détaillé.**

Prévenir les profs pour planifier des démo (non noté).

L’appli doit fonctionner avec des plan 1000\*1000, et au maximum 45 livraisons !

* Mettre en place fonction de **déplacement** sur la **carte**
* Mettre en place fonction de **zoom**

**Evaluer** combien de **temps** chaque **tâche** va nous prendre.

**Evaluer** la **priorité** du **client**.

* Certaines demandes du client sont **ultra simples** à mettre en place, à faire impérativement
* Ne **pas** **penser** que colis (infirmière en tournée prise de sang, personne ne qui doit faire signer des gens)
* Performance de fou mais impossibilité de modifier la tournée -> **NON !**
* **Plages horaires**

Supprimer livraison : recalculer Dijkstra entre les 2 livraisons qui « entourait » la livraison a supprimé