# Description du besoin

Afin d’améliorer la compétitivité des transports en commun, on voudrait visualiser la qualité des services fournis par la Société de transport de Montréal (STM) sur le territoire de l’ile de Montréal. Pour ce faire, nous allons créer une [carte thermique](https://fr.wikipedia.org/wiki/Carte_thermique) (« heatmap ») dont les couleurs indiqueront un score de « qualité du service » selon le lieu et l’heure. Par chance, [la STM rend disponible gratuitement sur son site web les données au format GTFS de l’ensemble de son réseau](https://www.stm.info/fr/a-propos/developpeurs) (arrêts, horaires, tarifs, etc.).

Pour simplifier la représentation de la carte thermique, nous allons subdiviser l’ile de Montréal en blocs de 100m par 100m (approximativement). De plus, puisque la qualité du service dépend de l’heure de la journée, nous allons subdiviser une journée de 24 heures en tranches de 15 minutes. Idéalement, ces paramètres devraient être modifiables dynamiquement dans le code du logiciel final, mais nous allons les fixer ici pour simplifier le développement du prototype.

Le calcul pour mesurer la qualité de l’offre de transport en commun au bloc dont le centre est aux coordonnées GPS [*x,y*] (en degrés) et à l’heure *t* (en minutes) est alors le suivant :

Calculer le nombre total de passages d’autobus différents[[1]](#footnote-1) pour tous les arrêts qui se trouvent dans le bloc centré en [*x,y*], au cours de la période allant de l’heure *t* à (*t + 15 minutes*).

Additionner le nombre total de passages de métro pour toute station qui se trouve dans le « super-bloc » centré en [*x,y*] (ie. tous les blocs entourant le bloc central de 100m x 100m, en plus de lui-même).

Pour générer la carte thermique à une heure *t*, il faudra donc effectuer ce calcul pour chaque bloc de 10000m2 sur l’ile de Montréal. Par la suite, nous voudrions voir cette carte changer dans le temps sur une période de 24 heures (par tranches de 15 minutes).

Le résultat final souhaité est une carte thermique animée dans le temps (par exemple, 15 minutes **→** 0.15 seconde) qui révèle visuellement la qualité du service de transport en commun sur l’ile de Montréal pour une journée donnée. Il n’est pas nécessaire que cette carte soit interactive (mettre en pause, « zoomer », etc.), mais ce serait un plus. Vous pouvez vous contenter de produire un simple fichier *.gif* ou *.png* (animé). Minimalement, une série d’images statiques correspondant à la carte générée pour chaque tranche de 15 minutes est également acceptable dans le cadre du prototype.

## Questions à poser au client :

* S’il manque des informations dans la description du besoin, inscrivez-les ici et j’y répondrai

# Technologies candidates

Ci-dessous, on veut lister les technologies qui pourraient convenir au besoin précédent, organisées selon leur catégorie.  
Vous pouvez indiquer « aucun » s’il n’est pas nécessaire de préciser le choix d’une technologie pour une catégorie donnée.

Pour chaque catégorie, vous devez remplir le tableau suivant en cherchant et documentant quelles technologies seraient des candidates potentielles afin de répondre au besoin du client.  
*Conseil : Mieux vaut proposer plus de technologies que moins, on fera le choix des technologies les plus adaptées plus tard, une fois toutes les options sur la table.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nom de la technologie | Description sommaire | Avantages | Limitations ou incertitudes | Références et exemples | Analyse (votre jugement) |
| Le nom de la technologie | Un court résumé de ce que ça fait (dans le cadre du besoin ci-haut). | Quelles sont les forces de cette technologie?   * Facilité d’apprentissage * Bonne documentation * Très populaire, établie depuis longtemps * Etc… | Quelles sont les faiblesses de cette technologie?   * Faible performance * Coût ou licence * Courbe d’apprentissage * Peu d’exemples connus * Etc… | Listez les sites, articles ou vidéos qui vous ont aidés à comprendre la technologie. Des exemples de projets similaires sont aussi très utiles. | Écrivez une ou deux phrases qui indiquent si vous pensez que cette technologie serait adéquate ou non au besoin, et pourquoi. |
| Ce tableau n’est qu’un exemple, n’inscrivez rien ici, utilisez plutôt les tableaux des catégories spécifiques dans les prochaines pages :  [Plateformes de déploiement (matériel, logiciels, services cloud, etc.) 2](#_Toc144118370)  [Sources de données (bases de données, API web ou locale, etc.) 2](#_Toc144118371)  [Langages, librairies et *frameworks* 2](#_Toc144118372)  [Environnement de développement 2](#_Toc144118373)  [Autres technologies 3](#_Toc144118374) | | | | | |

## Plateformes de déploiement (matériel, logiciels, services cloud, etc.)

Quelle sorte de matériel ou quelles alternatives (services web, virtualisation, etc.) sont nécessaires pour répondre au besoin?

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nom de la technologie | Description sommaire | Avantages | Limitations ou incertitudes | Références et exemples | Analyse (votre jugement) |
| Ordinateur pouvant exécuter du code python | N’importe quelle plateforme supportant le développement en python est suffisante. | n/a | * Exécution locale seulement | n/a | Le besoin ne nécessite qu’un ordinateur pouvant exécuter des scripts python. |

## Sources de données (bases de données, API web ou locale, etc.)

Si l’application aura besoin de conserver ou d’obtenir des données d’une DB ou d’une API, quelles technologies ou services pourraient les fournir?

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nom de la technologie | Description sommaire | Avantages | Limitations ou incertitudes | Références et exemples | Analyse (votre jugement) |
| API temps réel de la STM | La STM fournit une API i3 donnant accès aux données GTFS augmentées avec les informations des passages en temps réel. | * Accès au données temps réel sur la position des autobus * Possibilité de faire des calculs comparant les passages prévus vs. réels * API gratuite à utiliser | * Inscription requise * Plus complexe à intégrer que les données statiques * Type d’informations et d’API propre à la STM (moins réutilisable) | <https://www.stm.info/fr/a-propos/developpeurs>  [GTFS-Realtime](https://portail.developpeurs.stm.info/apihub/?_gl=1*d61w1h*_ga*NDMwMDg4MzY2LjE2OTMyMDY1NjI.*_ga_37MDMXFX83*MTY5NDAwNDAwNS41LjEuMTY5NDAwNDQzOS42MC4wLjA.#/login) | Les données temps réel permettraient des analyses plus poussées que celles demandées dans le besoin, mais elles sont un peu plus complexe à intégrer. |
| Données GTFS statiques | Toutes les informations sur les arrêts et horaires planifiés des autobus sont contenues dans un fichier ZIP au format GTFS. | * Simplicité d’accès aux données * Aucune connexion internet ou clé d’API requise * Données dans un format standard | * Limité aux données statiques (planifiées) * Doit être mis à jour une fois par saison | <https://www.stm.info/fr/a-propos/developpeurs>  <https://www.stm.info/fr/a-propos/developpeurs/description-des-donnees-disponibles>  <https://developers.google.com/transit/gtfs?hl=fr> | Pour répondre au besoin énoncé, les données statiques sont suffisantes (mais elles limiteraient peut-être les améliorations futures). |
| Base de données | Une base de données n’est pas nécessaire à priori pour répondre au besoin énoncé. | n/a | n/a | n/a | Une version future de l’application pourrait mettre des données en cache ou conserver des données historiques à l’aide d’une base de données, mais ce n’est pas demandé dans le besoin énoncé. |

## Langages, librairies et *frameworks*

Quels langages de programmation et quelles librairies ou *frameworks* de ces langages pourraient répondre au besoin?  
*Conseil : Le choix des librairies ou frameworks est souvent plus important que le choix du langage lui-même, donc mieux vaut commencer par là.*

| Nom de la technologie | Description sommaire | Avantages | Limitations ou incertitudes | Références et exemples | Analyse (votre jugement) |
| --- | --- | --- | --- | --- | --- |
| Python | Langage de programmation orienté-objet et fonctionnel avec typage dynamique. | * Facile à apprendre * Énormément de librairies existantes pour le traitement des données * Très utilisé; beaucoup de tutoriels disponibles | * Performance limitée * Beaucoup de développeurs amateurs => code des librairies parfois douteux | <https://www.python.org/about/gettingstarted/>  <https://www.learnpython.org/> | Étant donné que le besoin ne nécessite pas de faire des calculs en temps réel, les avantages de python en termes de facilité d’accès compensent nettement sa faible performance de calcul. |
| GTFS-kit (librairie python) | Librairie pour lire et manipuler les données GTFS. | * Permet d’analyser les données GTFS automatiquement * Régulièrement mis à jour * Licence MIT | * Relativement peu d’exemples d’utilisation disponibles en ligne * « This project’s development status is Alpha. I use GTFS Kit for work and change it breakingly to suit my needs. » | <https://pypi.org/project/gtfs-kit/>  <https://medium.com/analytics-vidhya/the-hitchhikers-guide-to-gtfs-with-python-e9790090952a> | Cette librairie est seulement en version « alpha » donc elle n’est peut-être pas très robuste, mais il n’y a pas beaucoup d’autres options…  Note : Cette librairie remplace la précédente « GTFS-TK » disponible sur gtfs.org |
| GeoPy (librairie python) | Librairie pour manipuler des données géographiques. | * Excellente documentation * Relativement couramment utilisée * Compatible avec de nombreux fournisseurs de données | * Probablement pas nécessaire à moins de vouloir ajouter des données d’autres villes * Trop complexe pour nos besoins * L’accès aux données nécessite des clés d’API *third-party* | <https://geopy.readthedocs.io/en/stable/>  <https://stackoverflow.com/questions/19412462/getting-distance-between-two-points-based-on-latitude-longitude> | Pour le besoin énoncé, nous n’avons pas à choisir d’autres lieux à analyser, donc on ne devrait pas avoir besoin de cette librairie. |
| Pandas (librairie python) | Librairie pour manipuler des jeux de données en mémoire. | * Fonctionnement semblable à une base de données (mais sans SQL) * Très populaire et répandu * Excellente documentation * Licence BSD-3 (libre) | * Puisque tous les calculs sont faits en RAM, c’est rapide mais possiblement gourmand * Peut devenir assez complexe car la librairie est très large * Pas de support direct des données géographiques | <https://pandas.pydata.org/docs/>  <https://www.w3schools.com/python/pandas/default.asp>  <https://pbpython.com/pandas_dtypes.html> | Étant donné que la librairie GTFS-kit utilise déjà Pandas en arrière-plan, peut-être que nous n’en aurons pas directement besoin, mais mieux vaut prévoir s’en servir pour faire nos calculs « sur mesure ». |
| GeoPandas (librarie python) | Extension de la librairie Pandas qui ajoute le support des données géographiques. | * Pandas + type de données propres aux données géographiques * Permet de manipuler des données GTFS * Permet de faire des graphiques avec GeoPlot | * Nettement moins connu et supporté que Pandas tout court * N’est pas utilisée par GTFS-kit ni GeoPy, malgré leurs fonctionnalités similaires | <https://geopandas.org/en/stable/>  <https://geopandas.org/en/stable/gallery/index.html> | Cette librairie ajoute des fonctionnalités très utiles pour nos besoins, mais elle ne semble pas s’intégrer facilement aux autres librairies dont nous avons besoin. Cependant, avec GeoPlot, on pourrait peut-être tout faire avec seulement celle-ci (mais elle est moins flexible dans l’ensemble). |
| Numpy / Scipy (librairies python) | Librairie pour les calculs matriciels ou scientifiques. | * Pratiquement intégré à python tellement elles sont utilisées * Optimisé en C * Énormément de fonctionnalités * Compatible avec Pandas | * Pas de support direct des données GTFS ou GPS (trop généraliste) * Facile à utiliser, mais complexe à bien utiliser | <https://numpy.org/doc/stable/user/absolute_beginners.html> | Puisque nous avons plusieurs calculs à effectuer sur des tableaux de données, ces librairies pourraient être utiles, mais il serait préférable de s’en tenir à Pandas si possible pour réduire le nombre de librairies à apprendre. |
| MatPlotLib (librarie python) | Librairie pour générer des graphiques. | * Librairie très connue et utilisée * Relativement simple à apprendre * Support des graphiques de type « heatmap » * Directement intégré dans Pandas et GeoPandas | * Fonctionnement un peu archaïque * Graphiques principalement statiques (non-interactifs), mais il est au moins possible de « simuler » un effet d’animation * Graphiques pas très « jolis » * Documentation moyenne | <https://matplotlib.org/stable/gallery/images_contours_and_fields/image_annotated_heatmap.html>  <https://matplotlib.org/stable/gallery/animation/simple_anim.html>  <https://pandas.pydata.org/pandas-docs/version/0.13.1/visualization.html> | Cette librairie est assez vieille et un peu limitée (un peu comme les graphiques dans Excel), mais elle devrait pouvoir répondre aux exigences du besoin. |
| Seaborn (ibrairie python) | Librairie basée sur MatPlotLib pour générer des graphiques. | * Plus moderne et plus « jolie » que MatPlotLib pour les graphiques simples ou courants * Permet de générer des graphiques plus complexes * S’intègre facilement à Pandas (mais pas GeoPandas) | * Plus complexe (et donc plus difficile à apprendre) que MatPlotLib pour les graphiques « sur mesure » * Moins populaire que MatPlotLib (mais quand même assez connu) * Ne permet pas de faire des graphiques interactifs | <https://seaborn.pydata.org/generated/seaborn.heatmap.html>  <https://pythonbasics.org/seaborn-heatmap/>  <https://www.linkedin.com/pulse/matplotlib-vs-seaborn-which-python-data-visualization-iram-mehmood-1f> | Seaborn est une évolution de la librairie MatPlotLib mais n’ajoute pas de fonctionnalités dont on a besoin, au contraire même, donc le fait qu’elle soit plus complexe et moins aisément intégrée aux autres librairies la rend moins attirante. |
| Plotly (librairie python) | Librairie permettant de créer des graphiques interactifs avec python. | * Permet de créer des graphiques interactifs (avec contrôles programmables) * Peut aussi être utilisée en version web/JS * Graphiques modernes * S’intègre facilement à Pandas (mais pas GeoPandas) * Supporte les graphiques animés * Bonne documentation | * Nettement moins populaire que MatPlotLib * Sensiblement plus complexe que MatPlotLib (parce que plus puissante) * Moins facile à « personnaliser en profondeur » que MatPlotLib (moins flexible car moins « manuelle ») * Permet de créer des dashboards (avec Dash), mais c’est payant | <https://plotly.com/python/plotly-fundamentals/>  <https://plotly.com/python/heatmaps/>  <https://www.reddit.com/r/learnpython/comments/vpvco2/matplotlib_vs_plotly_vs_seaborn_data_science_in/> | Plotly est nettement plus puissante que MatPlotLib et permettrait dans une version future de créer des graphiques interactifs et même une version web, cependant elle n’est pas similaire à MatPlotLib et sensiblement plus complexe, donc un changement de librairie en cours de développement de serait difficile. |
| Bokeh (librairie python) | Librairie permettant de créer des graphiques interactifs avec python. | * Librairie avec des fonctionnalités presque identiques à celle de Plotly * Permet de créer des dashboards entièrement en python * Meilleure performance que Plotly avec de larges jeux de données | * Version JS/web moins développée que celle de Plotly * Un peu plus difficile à utiliser que Plotly (ressemble un peu plus à MatPlotLib) | <https://docs.bokeh.org/en/latest/>  <https://programminghistorian.org/en/lessons/visualizing-with-bokeh>  <https://buggyprogrammer.com/bokeh-vs-plotly-which-one-is-better-in-2022/> | Le choix entre Plotly et Bokeh est difficile car elles ont des avantages très similaires, mais la meilleure performance de Bokeh avec de grandes quantités de données pourrait être utile pour nous car le heatmap risque d’être assez complexe. |

## Environnement de développement

Pour développer le logiciel répondant au besoin, quels outils ou processus de développement seraient utiles ou nécessaires?  
Certaines plateformes nécessitent parfois l’utilisation de logiciels spécifiques pour le développement d’applications; Si c’est le cas, il faut les identifier ici.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nom de la technologie | Description sommaire | Avantages | Limitations ou incertitudes | Références et exemples | Analyse (votre jugement) |
| PyCharm (IDE) | PyCharm est une IDE pour Python créée par JetBrains. | * Interface similaire à IntelliJ, que nous connaissons déjà * Supporte le débogage interactif et plusieurs autres fonctionnalités utiles * Version « community edition » gratuite | * Version gratuite limitée à Python seulement (mais suffisante pour nos besoins) | <https://www.jetbrains.com/pycharm/>  <https://www.jetbrains.com/products/compare/?product=pycharm&product=pycharm-ce> | Plusieurs autres IDE ou outils de développement existent pour python, mais le fait d’être déjà familier avec IntelliJ (qui ressemble à PyCharm) rend le choix facile. |

## Autres technologies

Y a-t-il d’autres catégories de technologies qui sont nécessaires ou utiles pour répondre au besoin?  
Par exemple, des logiciels ou applications existantes qui couvrent déjà la majorité des fonctionnalités répondant au besoin.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nom de la technologie | Description sommaire | Avantages | Limitations ou incertitudes | Références et exemples | Analyse (votre jugement) |
| n/a |  |  |  |  |  |

# Choix et justification des technologies

Indiquez ci-dessous quelles technologies vous avez décidé d’utiliser pour réaliser le projet décrit en haut.  
*Note : Il est possible que vous changiez d’avis lors de la mise à l’essai, et c’est tout à fait correct; Les choix fait ici sont seulement provisoires.*

* Le langage choisi sera **Python** (v3) car il existe de nombreuses librairies dans ce langage qui nous permettront de simplifier les tâches d’analyse et de visualisation des données.
* Le choix des librairies est moins évident et devrait être particulièrement mis à l’essai dans un prototype pour évaluer leurs capacités réelles, mais basé sur cette analyse préliminaire, nous recommandons :
  + GTFS-kit pour importer et manipuler les données de la STM (à partir des données GTFS statiques dans le cadre du prototype, et éventuellement avec l’API temps réel si le développement de l’application se poursuit dans le futur).
    - La librairie GeoPandas est une alternative intéressante, mais étant donné sa compatibilité incertaine avec les autres librairies choisies, il y a un risque quelle ne soit pas suffisante à long terme (selon le développement futur après la fin du prototype).
  + Pour effectuer les calculs de la qualité de service, la librairie Pandas semble appropriée, en particulier grâce à son excellente compatibilité entre les librairies qui importe les données, et les librairies pour générer les graphiques.
    - Au besoin, Numpy pourra être ajoutée si les calculs le requièrent, mais ça ne devrait pas être nécessaire; le développement du prototype nous le dira.
  + Du côté de la carte thermique, il y a de nombreuses options possibles, mais basé sur les analyses, on recommanderait de les essayer dans l’ordre de priorité suivant :
    - Bokeh (pour sa performance et le potentiel de développement futur de l’application)
    - Plotly (alternative à Bokeh, très semblable mais légèrement moins intéressant à priori)
    - MatPlotLib (si les options précédentes ne sont pas assez flexibles ou trop complexes pour nos besoins, MatPlotLib se démarque comme le choix populaire et accessible)
* Finalement, on recommande PyCharm comme environnement de développement

# Proposition de prototype

Afin de tester si ces technologies sont appropriées pour répondre au besoin énoncé en haut, proposez un petit sous-ensemble du projet qui permettrait de les mettre à l’essai. Vous pouvez recommander de faire plus d’un prototype s’il y a plusieurs nouvelles technologies très différentes à intégrer.

*Pour vous aider, demandez-vous quels sont les aspects clés à expérimenter pour estimer si le choix de la technologie est adéquat. Le prototype ne devrait prendre que quelques semaines à développer tout au plus (ou moins, si certaines technologies sont déjà maitrisées).*

Afin de guider ou valider nos choix de technologies ci-dessus, nous allons développer un prototype (ou « proof of concept ») de l’application répondant au besoin énoncé en haut. Ce prototype doit réduire le plus possible la quantité de développement requis mais permettre d’expérimenter avec les technologies clés, c’est-à-dire celles dont on est le moins certain.

En ce basant sur les résultats de l’analyse préliminaire, on veut que le prototype permette au moins :

* De lire et filtrer des données au format GTFS (provenant du fichier zip avec les données statiques)
  + Que ces données proviennent de l’API en temps réel ou seulement des données statiques, l’important est d’expérimenter avec la librairie choisie pour manipuler ces données afin de déterminer ce qu’elle peut (ou ne peut pas) faire.
* De calculer la valeur de la métrique de qualité (possiblement simplifiée en retirant les métros par exemple) pour une zone fixée d’avance (plus petite que « l’entièreté de l’ile de Montréal)
  + Bien que réduire la zone ou les plages horaires qui seront analysées ne réduit pas vraiment la quantité de code qui sera requise, ça simplifiera au moins la production de tests pour valider que les calculs sont corrects.
* De produire une carte thermique animée à partir des valeurs de la métrique de qualité
  + Ici, plusieurs options s’offrent à nous, mais la seule exigence présente dans l’énoncé du besoin est que la carte soit animée (interactive est seulement un plus).
  + Ce serait donc acceptable de choisir l’option la plus simple (donc MatPlotLib), mais personnellement dans ce cas je profiterais du prototype pour expérimenter une librairie plus complexe mais avec plus de potentiel futur (Bokeh), justement pour déterminer si ça en vaut la peine ou non.
  + Ceci dit, même si on choisit une librairie plus puissante que nécessaire, il suffit de générer un heatmap animé pour atteindre nos objectifs du prototype.

1. Si plus d’un arrêt de la même ligne d’autobus (et allant dans la même direction) se trouvent dans un même bloc, il ne faut pas les compter comme deux passages différents d’autobus, puisque c’est le même autobus qui fera les deux arrêts. Cependant, avec des blocs de 100x100m2, cette situation devrait être très rare et vous pouvez l’ignorer dans votre prototype. [↑](#footnote-ref-1)