Liv'In Paris

Liv'in Paris propose des services de partages de repas entre voisins plus ou moins distants mais toujours dans Paris intramuros.

Pour cela, que l'on soit cuisinier ou client, il faut s'inscrire auprès de l'application en déclinant son identité. (nom, prénom, adresse, téléphone, adresse email).

Un cuisinier peut devenir client et inversement ou être les 2 à la fois de manière simultanée.

Un client peut être un particulier, ou une entreprise locale.

Dans le cas d'un particulier, son nom, prénom, adresse, numéro de téléphone et autres données si nécessaire permettent de le définir.

Dans le cas d'une entreprise locale, le nom de l'entreprise et possiblement le nom d'un référent seront nécessaires pour assurer une bonne communication.

En échange, chacun reçoit un identifiant unique et un mot de passe (personnalisé et/ou à personnaliser)

Pour cela, le cuisinier partage sur le site, le plat confectionné en précisant pour chacun des mets s'il s'agit

- D'une entrée
- Plat principal
- Dessert

D'autres informations sont indispensables

- Pour combien de personnes
- Date de fabrication et de péremption
- Prix par personne
- Nationalité de cuisine (chinoise, mexicaine …)
- Régime Alimentaire (Végétarien, Sans gluten, Viande Halal …)
- Les ingrédients principaux pour aider les clients à choisir.
- Une petite photo est la bienvenue !
- et autres données en fonction de vos envies !

Les plats peuvent être génériques et obéir à une recette déjà enregistrée ou bien être une déclinaison d'une recette déjà existante.
Le cuisinier se doit d'assurer la livraison de son plat, il peut combiner plusieurs livraisons si nécessaire en fonction des zones de livraisons.
La transaction peut être simple ou multiple en effet, il est possible qu'au sein d'une même commande, le client puisse avoir plusieurs lignes de commandes. Chaque ligne de commande peut avoir une date et un lieu de livraison différent. En revanche, la transaction financière est réalisée une fois pour toute avant la première livraison.

La base conserve l'historique de toutes les transactions réalisées entre un cuisinier et un client afin de pouvoir identifier les plus grands cuisiniers par mois, par an … et réorganiser au besoin les transactions en fonction des trajets effectués, des repas livrés, de la fréquence … et des retours des clients.
En fonction de ces retours, il est possible qu'un client ou un cuisinier puisse être radié de la plateforme.

Pour faciliter les transactions, l'application met à la disposition des cuisiniers une fonction qui va calculer le plus court chemin entre les 2 adresses celle du cuisinier et celle du client. Par soucis de simplification, bien sûr, nous ne traiterons que le cas des lignes de métro dans un premier temps.

Vous proposerez donc une application C# (en lieu et place d'un site internet) qui sous forme d'un menu principal permettra de traiter les clients, les cuisiniers, les commandes et les trajets

Module Client de l'interface

L'outil doit pouvoir permettre d'entrer, supprimer ou modifier un nouveau Client depuis la console ou depuis un fichier

Il faut à tout moment pouvoir afficher l'ensemble des Clients selon plusieurs critères : (successivement ou simultanément)

- Par ordre alphabétique
- Par rue

- Par montant des achats cumulés, ce qui permettra de connaître les meilleurs clients

Module Cuisinier de l'interface

De même un cuisinier peut être inséré, modifié ou supprimé depuis la Console ou depuis un fichier.
Il est demandé d'afficher pour un cuisinier a minima

- les clients qu'il a pu servir depuis son inscription à la plateforme ou depuis une tranche de temps proposée,
- le ou les plats qu'il a réalisé par fréquence
- le plat du jour proposé

Module Commande de l'interface

Pour faciliter la simulation, il vous est demandé de pouvoir partir d'une situation initiale où aucune commande n'existe et au fil de l'utilisation de l'application sauvegarder les commandes effectuées afin de créer une base qui s'autoenrichit.

Il faut donc pouvoir créer une nouvelle commande ou la modifier et simuler ses différentes étapes
Une commande ne peut être réalisée que si le client existe dans la base ou sinon il faut le créer. Une commande doit mettre en jeu un parcours entre une adresse de départ et une adresse d'arrivée et établir le **chemin le plus court** pour la date déterminée.
Il faut pouvoir à tout moment

- Calculer le prix d'une commande et l'afficher moyennant son numéro
- Déterminer le chemin que devra parcourir le cuisinier pour sa livraison.

Module Statistiques

Il faut également faire des bilans généraux

- Afficher par cuisinier le nombre de livraisons effectuées

- Afficher les commandes selon une période de temps
- Afficher la moyenne des prix des commandes
- Afficher la moyenne des comptes clients
- Afficher la liste des commandes pour un client selon la nationalité des plats, la période

Module Autre qui sera le résultat de votre créativité

- 5 suggestions sont attendues

Séquencement

Le projet peut se faire à 3 maximum avec des étudiants du même groupe

Il doit rendre compte du cahier des charges ci-dessus. Tout information non écrite peut être interprétée comme vous le souhaitez.

Dans tous les cas, il faut utiliser tous les concepts vus en Algo et POO, Algorithmique et graphes et Base de données

- POO (création de classes)
- Lecture de fichiers, utilisation de collections génériques différentes
- Accès aux bases de données
- Notions et propriétés de Graphe vu en cours d'Algorithmique et graphes
- Algorithme de recherche du chemin le plus court (Dijkstra, Bellman-Ford, ...)
- Coloration de graphes
- Couverture de graphes

Nous vous proposons de travailler par étape. Chaque étape donnant lieu à un livrable.

Etape 1 : (→ à réaliser au plus tard pour le 1 mars)

Du point de vue des graphes

Dans un premier temps vous allez vous entraîner sur un sujet plus simple avec peu de données pour mettre en application les concepts vus en Algorithmique et Graphes, pour cela vous avez en annexe le fichier Association.zip. Ce fichier met en relation des membres d'une association. Chaque membre est identifié par un numéro. Les relations entre les membres de l'association sont supposées réciproques.
A partir de cet exemple, vous consolidez en C# une architecture logicielle pour traiter ces données sous forme d'un graphe.

- Créer les classes Nœud, Lien, et Graphe pour traiter ce cas d'école
- A partir du fichier joint, instancier le graphe.
    - Utilisez 2 modes : Listes d'adjacence et Matrice d'adjacence
- Implémentez les deux algorithmes classiques de parcours d'un graphe à partir d'un sommet quelconque : Parcours en Largeur d'abord et Parcours en Profondeur d'abord. Affichez les sommets dans l'ordre de visite pour chaque méthode.
- Déterminez si le graphe est **connexe ?** Justifiez.
- Vérifiez si le graphe contient des **circuits** (cycles) ? Expliquez votre méthode et donnez des exemples si possible.

Bonus : Analysez et mentionnez des propriétés supplémentaires du graphe (ordre du graphe, taille du graphe, son type : orienté, pondéré…).

Vous proposerez ensuite un outil de visualisation du graphe en C#
Pour l'outil de visualisation, vous pouvez utiliser les librairies

- System.Drawing de Microsoft
- ou d'autres librairies comme SkiaSharp.(using SkiaSharp;)

Pour résoudre les problèmes de visualisation, vous pouvez vous aider des outils d'IA générative, vous devrez être en mesure d'expliquer le fonctionnement de l'un ou l'autre des modes et de fournir les prompts soumis.

Du point de vue BDD

A partir du cahier des charges, créer votre schéma Entité Association

A partir de ce schéma, vous développez en SQL les tables que vous allez peupler à partir des fichiers Clients.csv, Cuisiniers.csv et Commandes.csv donnés à titre d'exemple, que vous complétez en fonction de vos besoins.

Préparez quelques requêtes simples sur votre base de données ainsi créée.

Aucun code C# n'est utile dans cette première partie BDD

Livrables au 1er Mars

Vous déposerez un rapport dans lequel vous détaillerez les 2 parties

- Votre schéma Entité Association et le script SQL de la création de la base.
- Les prompts issus des IA génératives concernant la visualisation du graphe

Vous déposerez votre solution C# qui implémente la construction et le parcours du graphe à partir de l'exemple fourni.

La solution doit proposer les commentaires ///, un projet de Tests Unitaires

ETAPE 2 : Généralisation de la solution et intégration des 2 parties (30 Mars)

Du point de vue des graphes

Vous allez modifier les classes de manière à les rendre génériques (créer des classes avec le type générique class Nœud<T> et Graphe<T> pour que ce soit possible d'utiliser ces classes pour n'importe quel type de nœuds. Et vous appliquerez vos modifications au plan du métro de Paris données en Annexe et sur le lien https://www.sncf-connect.com/ile-de-france/metro

Attention certaines liaisons entre 2 stations de métro ne sont pas à double sens.

Les stations sont identifiées par leurs longitude et latitude. Pour calculer la distance entre 2 points à partir de ces données, vous pourrez utiliser la formule suivante :

La formule de Haversine

La formule est la suivante :

$$d = 2 \cdot R \cdot \arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1) \cdot \cos(\phi_2) \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right)}\right)$$

Définition des termes

- d : Distance entre les deux points (en mètres ou kilomètres, selon R).
- R : Rayon de la Terre (en général, $R = 6371\,\text{km}$).
- ϕ_1, ϕ_2 : Latitudes des deux points (en radians).
- λ_1, λ_2 : Longitudes des deux points (en radians).
- $\Delta\phi = \phi_2 - \phi_1$: Différence de latitude.
- $\Delta\lambda = \lambda_2 - \lambda_1$: Différence de longitude.

Le fichier présente la liste exhaustive des lignes de métros avec leurs latitudes et longitudes ainsi qu'une proposition partielle du chemin à parcourir sur chacune des lignes. Le temps entre 2 stations est arbitraire. Il faudra également tenir compte d'un temps de changement dans certaines stations comme à Charles de Gaulle entre la ligne 1 et la ligne 2. Là aussi, vous pouvez partir sur des données arbitraires.

Une fois que votre instanciation de graphe est au point, nous vous proposons, à partir des algorithmes vus en cours d'Algorithmique et Graphes, de trouver le chemin le plus court entre les 2 adresses Client et Cuisinier qui pourront se résumer entre les 2 stations de métro les plus proches.

Les algorithmes à développer sont les suivants :

- **Dijkstra**
- **Bellman-Ford**
- **Floyd-Warshall** (référez-vous à la documentation comme Wikipédia pour implémenter cet algorithme).
- **Remarque :**
  - Déterminez le chemin le plus court entre **deux sommets** du graphe de votre choix.

  - De préférence, choisissez deux sommets situés aux **extrémités du graphe** et pour lesquels **plusieurs chemins** existent.
- **Comparez les résultats des trois algorithmes**
  - Évaluez leurs performances en termes de : complexité algorithmique, rapidité d'exécution et pertinence pour ce cas d'application.
  - Justifiez vos réponses.

Enfin, vous adapterez les outils graphiques pour être en mesure de visualiser le meilleur chemin à parcourir en affichant toutes les stations parcourues ainsi que toutes les données de temps utilisées.

Du point de vue BDD

Vous développez l'interface Console C# de votre application ainsi que l'intégration des appels de vos requêtes pour satisfaire chacun des modules donnés dans le cahier des charges.

Livrables :

Vous déposerez un rapport dans lequel vous détaillerez les 2 parties

- Votre schéma Entité Association et le script SQL de la création de la base. (si changements)
- Les prompts issus des IA génératives concernant la visualisation du graphe (si besoin)

Vous déposerez votre solution C# qui implémente la construction et le calcul du chemin le plus court.
La solution doit proposer les commentaires ///, un projet de Tests Unitaires
A ce stade, vous pouvez composer une solution graphique avec les bibliothèques Forms ou WPF (bonus considéré dans la partie BDD) et/ou implémenter un autre algorithme de recherche de chemin le plus court (bonus considéré dans la partie graphe)

Etape 3 : Gestion de couverture de graphe (=> 5 Mai) 25%)

Dans cette étape 3, une partie est imposée concernant les 2 modules l'autre fait la part belle à la créativité.

Les relations clients et cuisiniers sont établies par les commandes effectuées entre les 2 parties dans l'historique des commandes stockées dans la base de données.
A partir de vos données, que vous allez incrémenter au fur et à mesure de vos expérimentations, vous appliquerez votre outil de visualisation en l'adaptant aux consignes suivantes

---

Du point de vue des graphes

Coloration de graphe

Appliquez la **coloration de graphe** sur le graphe généré à partir de vos données en utilisant l'**algorithme de Welsh-Powell** pour répondre aux problématiques suivantes :

- **Nombre minimal de couleurs :** Déterminez le nombre minimal de couleurs nécessaires pour colorier le graphe.
- **Propriétés du graphe** : À partir des résultats de la coloration obtenus, répondez aux questions suivantes :
  - Le graphe est-il biparti ? Justifiez.
  - Le graphe est-il planaire ? Justifiez.
- **Groupes indépendants :** Identifiez les **groupes indépendants** à partir des résultats de coloration (par exemple, des clients ne partageant aucun cuisinier commun).

Analyse des résultats de la coloration :

i. Examinez les résultats de la coloration pour identifier **des caractéristiques pertinentes** de votre graphe/application.

Du point de vue BDD

Vous profiterez de l'exploitation de ces données pour les exporter dans les formats Json et XML.

Etape 3 : Créativité

Cette partie n'est pas consolidée dans sa totalité car elle dépend de vos appétences à découvrir tout secteur (découverte d'autres algorithmes d'étude de graphe, de visualisation, d'approfondissement de SQL …) Une idée dans chaque domaine est proposée :

- **Visualisation** : Utilisation de OpenStreetMap : https://www.openstreetmap.org/

  OpenStreetMap (OSM) offre une base de données géographique libre et collaborative où les relations entre stations et lignes sont modélisées.

  En utilisant des outils comme Overpass Turbo, https://overpass-turbo.eu/ il est possible d'extraire ces relations pour obtenir des détails sur les correspondances entre stations.

- **Couverture de graphe :** Dans le cadre du cours d'Algorithmique et Graphes, vous avez étudié les algorithmes permettant de déterminer un arbre couvrant de poids minimum dans un graphe non orienté (comme les algorithmes de Kruskal ou Prim). Cependant, dans cette application, nous travaillons avec un graphe orienté, où le sens des trajets est essentiel.

  Pour ce cas particulier, nous vous proposons de découvrir et de développer l'algorithme de **Chu-Liu/Edmonds**, adapté aux graphes orientés. Cet algorithme permet de trouver une arborescence couvrante de poids minimum dans un graphe orienté.
  Pour vous guider dans cette démarche, nous vous recommandons de consulter les ressources suivantes :
  i. Référence 1 : Chu-Liu/Edmonds Algorithm on Wikipedia
  ii. Réference 2 : [lien], voir page 128.

  Instructions :

  a. Familiarisez-vous avec les étapes principales de l'algorithme et comprenez son fonctionnement.
  b. Implémentez l'algorithme pour le graphe orienté généré à partir de vos données.
  c. Déterminez une arborescence couvrante minimale à partir d'un sommet racine représentant un point de départ (par exemple, un cuisinier central), connectant tous les clients avec un coût total minimal tout en respectant la direction des trajets.

- **SQL** : Utilisation de transactions pour assurer l'intégrité des données en toutes circonstances

Proposition de Découpage en TDs

TD1 : BDD - Prise en main de l'outil Looping par exemple, de GitHub, et du modèle Entité Association.

Graphe - Revoir ce qu'est un Graphe, et la manière de l'implémenter en C#

TD2 : BDD - Création de tables et de données Validation du modèle

Graphe : Implémentation du graphe simple en C# et utilisation des outils graphiques

TD3 : Implémentation depuis une application C#

Graphe : Recherche du meilleur chemin

TD4 : Finaliser l'intégration de tous les modules

TD5 : Revue de code