Projet : le jeu de société *Scotland Yard*

UCBL - Département Informatique de Lyon 1 – BDW1 - automne 2019

- Ce projet est évalué par deux notes : soutenance et réalisation.
- Cette UE est avant tout une introduction aux **bases de données**. La non-utilisation d'une BD dans le projet peut se voir attribuer la note 0.
- Le rendu du projet et la soutenance ont lieu lors de la dernière séance de TP. Tout rendu ne respectant pas les consignes peut se voir attribuer la note 0.
- Vu la quantité de travail demandé, il est fortement recommandé de réaliser le projet en binôme (pas de trinôme). Les étudiant.e.s travaillant seul.e sur le projet seront notés comme ceux travaillant en binôme. Un binôme signifie deux étudiant.e.s **du même groupe de TD.**
- Attention à la gestion de votre temps : concentrez-vous sur l'implémentation des fonctionnalités demandées, et pas plus. Cela ne sert à rien de gagner un point bonus pour quelques améliorations mineures mais de perdre plusieurs points pour des fonctionnalités non développées.

Le projet de cette année est inspiré du jeu de société *Scotland Yard*[1], dans lequel des détectives essaient d'attraper Mister X. Les protagonistes se déplacent sur les 199 stations du plateau de jeu (voir figure 1) en utilisant taxi, métro, bus ou bateau. La position de Mister X est secrète, mais les détectives connaissent les moyens de transport qu'il utilise à chaque tour. De plus, Mister X doit dévoiler sa position à certains tours. Plus de détails sur le jeu sur tric trac ou boardgamegeek. Il existe plusieurs éditions du jeu, mais le projet se concentre sur les règles de base adaptées.

1 Des spécifications au script SQL

Vous allez donc programmer une variante simplifiée du jeu *Scotland Yard*. La première différence, c'est que la traque se déroule à Lyon (cf le paragraphe *description du jeu de données fourni*). Ensuite, la partie n'est pas multi-joueurs : sur votre application, une seule personne (humaine) joue le rôle détective pendant une partie, et Mister X et les autres détectives sont déplacés par la machine. Vu les difficultés techniques, nous n'utiliserons pas de carte (comme sur le plateau) et les fonctionnalités seront limitées.

Description du jeu de données fourni : on considère que nous nous déplaçons entre quartiers (plutôt qu'entre stations). Une base de données (`dataset` dans PHPMyAdmin) vous est fournie (accès en lecture seulement) avec des informations sur les quartiers de Lyon et Villeurbanne (table `quartiers`), ainsi que les moyens de transports disponibles entre ces quartiers (table `routes`). Attention : les routes sont celles du jeu de société original, elles ne respectent donc pas forcément la réalité des quartiers de Lyon ! L'utilisation de ces données est **obligatoire**.

Spécifications : Les quartiers possèdent un identifiant, un nom, un code INSEE (unique), un type (H pour habitat, A pour activités, D pour divers, Z pour regroupement). Chaque quartier possède une géométrie, c'est à dire des coordonnées géographiques (latitude et longitude) ordonnées. Un quartier peut servir ou pas de point de départ pour Mister X ou les détectives. Un quartier est situé dans une commune, dont on connait le nom et le code postal. Chaque commune est rattachée à un seul département. Une route correspond à un trajet entre deux quartiers au moyen d'un type de transport. Les joueuses humaines sont stockées en base, avec leur nom

1. https://fr.wikipedia.org/wiki/Scotland_Yard_(jeu)

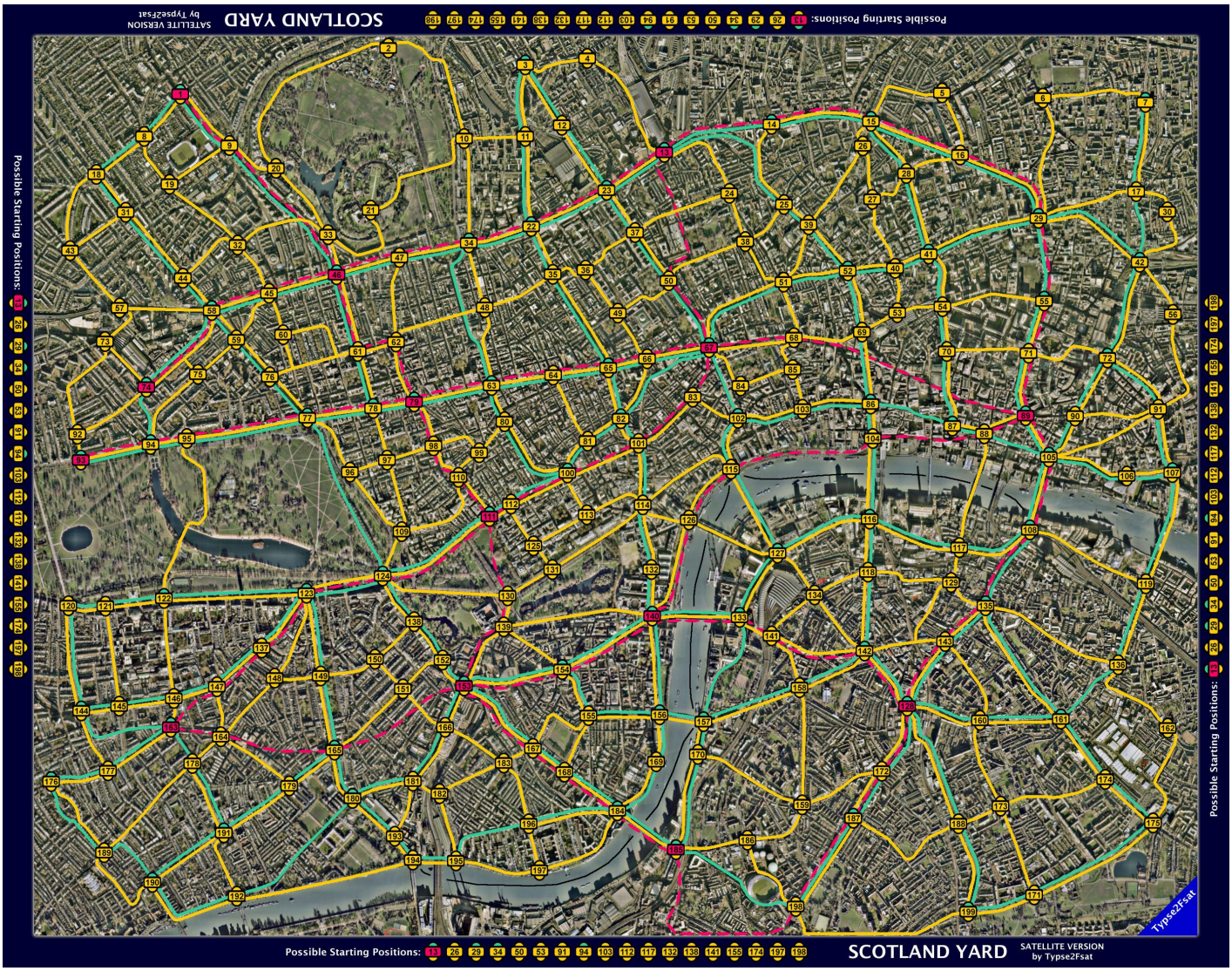

FIG. 1: Aperçu d'une version du plateau de jeu de Scotland Yard : on se déplace entre stations en taxi (lignes jaunes), en bus (lignes vertes, arrêt uniquement aux stations avec un "chapeau" vert), en métro (lignes rouges, arrêt uniquement aux stations rouges) et en bateau (lignes noires sur la Tamise).

et leur email. Une joueuse participe comme détective à une ou plusieurs parties. Si les détectives gagnent la partie, on enregistre la victoire de la joueuse. Une partie possède une date de démarrage, le nombre de détectives et une configuration. La configuration inclut un nom (e.g., *défaut*), une date, une stratégie (basique, économe ou pistage) et un ensemble d'images pour représenter les tickets de transport. Chaque image a un nom et un chemin (vers le fichier image sur le serveur). Seuls les tours de Mister X sont stockés en base : le numéro de tour sert d'identifiant, et on enregistre la route utilisée (quartier de départ, celui d'arrivée et le moyen de transport).

Dans un premier temps, nous allons concevoir la base de données à partir des spécifications. Votre modélisation doit respecter au mieux ces spécifications, mais votre site web n'utilisera pas tous les concepts décrits.

1. Produisez un diagramme entité / association pour ces spécifications (par exemple avec AnalyseSI, MoCoDo ou JMerise).
2. Produisez le schéma relationnel dérivé de votre diagramme E/A. Si vous générez ce schéma avec un outil de modélisation, il est recommandé de le vérifier et éventuellement de le corriger / compléter.
3. Produisez le script SQL permettant la création de la base de données. Si vous générez ce script avec un outil de modélisation, il sera nécessaire de le vérifier et de le corriger / compléter (ces modifications devraient être stockées dans un autre fichier que celui généré par l'outil de modélisation).

2 Design du site et pages statiques

Vous êtes libres d'organiser votre site comme bon vous semble. Mais chacune de vos pages doit avoir les cinq zones suivantes :

- Un entête (`<header>`, avec un logo (au choix, mais cliquable pour revenir à l'accueil) et un nom de site;
- Un menu (`<nav>`), dont les libellés seront explicites;
- Des statistiques sur la base de données : nombre d'instances pour les tables importantes, nombre de parties totales/gagnées/perdues par joueuse, top-3 des meilleures détectives, etc.;
- Le contenu de la page, qui correspond aux fonctionnalités développées dans les sections suivantes. Prévoyez également de créer une page d'accueil qui décrit les objectifs de votre site;
- Un pied de page (`<footer>`), avec un lien vers le site de l'UCBL, un autre vers la page du BDW1, l'année courante et des remerciements (e.g., site OpenClipart pour les images).

La mise en page et mise en forme se feront évidemment avec des styles CSS. L'esthétique est prise en compte lors de la notation, aussi soignez votre site. Le projet sera évalué avec le navigateur **Mozilla Firefox**, donc vérifiez le rendu de votre site avec ce navigateur !

3 Fonctionnalité 1 : intégrer et afficher les données

Cette fonctionnalité consiste à migrer les données existantes (quartiers et routes) vers votre schéma. En effet, les deux tables fournies (base de données `dataset`) ne sont pas bien modélisées, et il est donc préférable de stocker leur contenu dans votre base. C'est un processus fréquent appelé *intégration de données*. Si besoin, **utilisez PHPMyAdmin pour insérer des instances** pour certaines de vos tables (e.g., les 4 types de transport).

Vous allez donc écrire un script PHP qui interroge la base `dataset` pour retrouver certaines données, les transforme éventuellement, puis insère dans votre base les données transformées. En théorie, ce script ne s'exécute qu'une seule fois. Mais en pratique, vous allez le tester à plusieurs reprises . N'oubliez pas de vider votre base de données avant de le ré-exécuter !

On peut décomposer l'étape d'intégration en trois étapes :

- Réfléchir aux correspondances entre les données existantes et votre schéma (e.g., idQ correspond à l'identifiant du quartier, que j'ai nommé dans mon schéma *id_quartier*). Vérifier si les types de données sont cohérents et si des transformations sont nécessaires. Le jeu de données fourni n'est pas bien modélisé;
- Écrire une requête `select` pour récupérer les informations pertinentes. Pour requêter, il faut indiquer le nom de la BD avant la table : `SELECT * FROM dataset.Routes;`
- Pour chaque tuple résultat, appliquer éventuellement des transformations sur les valeurs, puis écrivez une requête `insert` pour peupler vos tables. N'oubliez pas de peupler les tables intermédiaires !

Avant la soutenance, il est possible que l'on supprime la base de données `dataset`. Si votre intégration est réussie, vous disposez normalement des données de quartiers et routes dans votre base (*199* quartiers et *936* routes).

Pour simplifier l'affichage, on utilisera un tableau (e.g., 20 lignes de 10 colonnes) qui contiendra dans chaque cellule : l'identifiant du quartier, son nom, sa ville, et le numéro des quartiers vers lesquels on peut se rendre (avec le type de transport). Normalement, vous savez déjà faire cette étape d'affichage (similaire au TP4).

Si vous avez le temps, vous pouvez améliorer cette fonctionnalité en proposant une meilleure façon de présenter les données. L'affichage sous forme d'un graphe (e.g., avec d3.js) ou d'une carte (e.g., avec Leaflet ou OpenLayers) est pertinent, mais difficile à implémenter (langage JavaScript, gestion d'événements, etc.).

4 Fonctionnalité 2 : programmer une partie (stratégie basique)

Cette fonctionnalité permet de programmer une partie et ses 20 tours de jeu. Si besoin, utilisez PHPMyAdmin pour insérer rapidement des instances (e.g., des joueuses, une configuration par défaut).

Pour démarrer une partie :

- Configurer la partie. Sur un écran, la joueuse humaine saisit son prénom et choisit le nombre de détectives (3, 4 ou 5);
- Initialiser la partie. En particulier, créer la partie et la joueuse (si besoin) dans la base de données, générer les positions de départ des joueuses (attention, uniquement sur les quartiers dont l'attribut *isQuartierDepart* vaut 1), initialiser une session, etc.

Un tour de jeu se décompose ainsi :

- Déplacer Mister X. La machine choisit aléatoirement un quartier d'arrivée valide (i.e, avec une route entre le quartier de départ et celui d'arrivée). Il est évident qu'il faut aussi vérifier que Mister X ne se déplace pas sur un quartier occupé par un.e détective. S'il n'y a pas de quartier valide, les détectives ont encerclé Mister X et gagnent donc immédiatement la partie ! La base de données est mise à jour (nouvelle position, type de transport utilisé);
- Permettre à la joueuse humaine de se déplacer. Elle doit donc sélectionner (bouton radio) ou saisir (champ texte) son quartier d'arrivée. L'application doit s'assurer que le déplacement est autorisé (i.e., qu'il y a bien une route directe entre son quartier actuel et celui sélectionné), sinon affiche un message explicatif et demande de sélectionner une nouvelle destination;
- Déplacer les autres détectives (stratégie *basique*). Pour chaque autre détective, la machine choisit aléatoirement un quartier valide (i.e., via une route). Cependant, si Mister X se trouve sur l'un de ces quartiers valides, alors l'application choisit celui-ci;
- En fin de tour, vérifier si la partie est terminée : si un.e détective est dans le même quartier que Mister X, alors les détectives ont gagné. Si c'est la fin du $20^{\text{ème}}$ tour, alors Mister X a gagné la partie.

Si vous avez le temps, vous pouvez ajouter d'autres options pour configurer une partie (saisie de l'email, choix des images des cartes, possibilité de choisir son nom dans une liste extraite de la BD, etc.).
Lors de la soutenance, il est fort probable que les enseignant.e.s testent votre application en saisissant des valeurs absurdes susceptibles de déclencher des erreurs dans votre application. Alors essayez de penser au pire !

5 Fonctionnalité 3 : gérer les tickets de transport (stratégie économe)

Nous ajoutons maintenant la gestion des tickets de transport. Les tickets sont montrés ci-dessous (taxi, bus, métro, et le "black ticket" pour n'importe quel transport, y compris bateau). Mister X dispose d'un nombre de tickets illimité[2] et il est le seul à pouvoir utiliser les "black tickets". Chaque détective possède un nombre limité de tickets : 10 tickets de taxi, 8 de bus, et 4 de métro.

Pour cette fonctionnalité, il faut :

- Insérer les tickets dans la BD (avec le chemin vers l'image stockée sur le serveur), puis créer une configuration qui utilise ces tickets. Vous pouvez ajouter d'autres images plus originales;
- Même si Mister X se déplace sans contrainte, il faut stocker en BD ses déplacements. Il faut aussi afficher les moyens de transport successifs qu'il a utilisé depuis le premier tour;

2. Peut-être est-ce la raison pour laquelle il est recherché...

- Gérer les tickets de chaque détective : avant un déplacement, vérifier que le détective possède encore un ticket pour le transport choisi. Mettre à jour son nombre de tickets après le déplacement. Prévoir le cas où un détective ne peut plus se déplacer (trop épuisé, sa traque est terminée et il va boire un verre).

Si vous avez le temps, vous pouvez adapter l'algorithme de déplacement des détectives (stratégie *économe*) afin de ne pas griller rapidement leurs (rares) tickets de métro...

6 Fonctionnalité 4 : gérer les apparitions de Mister X (stratégie pistage)

Cette dernière fonctionnalité gère les apparitions de Mister X, aux tours 3, 8, 13, 18 (ainsi qu'en fin de partie).

- Après son déplacement, afficher la position de Mister X pour les détectives;
- Développer un algorithme de plus court chemin, par exemple en s'inspirant de l'algorithme de Dijkstra ou l'algorithme A* ou celui de Bellman-Ford. Si des poids sont nécessaires, vous pouvez considérer `+1` par quartier, ou mettre un poids différent selon le moyen de transport;
- Afficher (pour la joueuse humaine) et utiliser (pour les autres détectives) le chemin calculé par votre algorithme (stratégie *pistage*).

Si vous avez le temps, vous pouvez aussi prédire les positions possibles de Mister en utilisant votre algorithme (chemins possibles à partir du quartier de sa dernière apparition et de ses transports utilisés depuis).

7 Préparation des livrables

Trois livrables sont à rendre **le jour de soutenance de votre projet, avant minuit, sur Tomuss** :

- Une archive contenant votre site web, en zip ou rar (colonne *archive_projet*)
- L'URL de votre site, sur le serveur `bdw1.univ-lyon1.fr` (colonne *url_projet*)
- Les transparents de soutenance en pdf (colonne *transparents_projet*)

7.1 Archive avec le site web

L'archive est au format **zip** ou **rar**, et respecte les contraintes suivantes :

- Une **seule archive par binôme**
- L'archive contient au minimum **4 fichiers** :
  - le répertoire contenant les pages de votre site (code commenté et indenté). La page *index* contiendra vos noms et numéros étudiant en commentaire;
  - un fichier `.txt` ou `.sql` avec le script SQL **"exécutable"** de création de votre base de données;
  - deux images (ou PDF), l'une de votre diagramme E/A et l'autre de votre schéma Relationnel.

7.2 URL du site web

Après le rendu du projet, nous testerons votre application en utilisant l'URL saisie dans Tomuss :

- Elle doit être complète (login et répertoire(s), voire fichier index précisé si nécessaire) et elle doit pointer vers la page d'accueil de votre site (e.g., `http://bdw1.univ-lyon1.fr/p1234567/mon_site/`). Merci de la vérifier une fois qu'elle est saisie dans Tomuss !
- Votre site doit déjà avoir des instances (routes, quartiers, configuration, etc.) stockées en base en nombre suffisant pour simplifier les tests.

7.3 Soutenance

La soutenance dure 20 minutes par binôme et se décompose en deux parties : une présentation avec démo et avec transparents (8 minutes) et une séance de questions (12 minutes). Les conditions suivantes s'appliquent :

- Pour la partie présentation, respectez le temps ! Vous serez interrompus au bout des 8 minutes, et donc pénalisés.
- Les deux membres du binôme doivent parler lors de la présentation. Entrainez-vous.
- Vous présentez soit sur votre machine, soit sur une machine de la fac. Dans tous les cas, vérifiez avant que la machine permette de lire vos transparents, de faire la démo, etc. Prévoyez d'arriver quelques minutes en avance devant la salle, et quand on vous fait entrer, préparez-vous pour la présentation et les questions (i.e., lancement du site dans le navigateur, ouverture des transparents, ouverture des fichiers source et des diagrammes).
- Vous devez répondre à quatre questions minimum pendant les 12 minutes. Donc donnez des **réponses claires et concises** (pas de baratin). Si vous ne savez pas répondre à une question, dites-le pour ne pas perdre de temps.
- Si vous arrivez en retard à votre soutenance, vous passez en fin de séance... ou pas du tout selon les disponibilités de votre jury.

Vous pouvez réaliser vos transparents avec n'importe quel logiciel (e.g., Libre Office, Power Point, Latex Beamer). Vous devrez par contre **déposer un PDF de vos transparents** sur Tomuss. Les conditions suivantes s'appliquent pour les transparents :

- Les transparents seront soignés et numérotés. Évitez les longues phrases !
- Vos 5 transparents doivent respecter le plan suivant :
  - Transparent 1 : vos noms, prénoms, numéros étudiant, et si vous participez au concours, le nom et/ou le logo de votre site.
  - Transparent 2 : réalisations (fonctionnalités implémentées, améliorations éventuelles).
  - Transparent 3 : difficultés rencontrées au niveau technique, organisationnel, etc. (évitez l'éternel "manque de temps").
  - Transparent 4 : diagramme E/A (prévoir un agrandissement si c'est illisible).
  - Transparent 5 : une capture d'écran d'une partie en cours, qui permet d'annoncer la démo.
- La démo de votre site doit être préparée (déroulement d'un scénario). Vous avez 8 minutes pour les transparents et la démo, donc focalisez-vous sur les points importants.

Concours (optionnel) : Pour vous détendre (en cas de blocage sur un bug, d'engueulade avec votre binôme, etc.), deux concours sont organisés, celui du **meilleur nom de site** et celui du **plus beau logo**. Les deux binômes gagnants remporteront un paquet de chamallows. Soyez créatifs/ves !