Projet : Réalisation d'une application web "Eat and See!"

Ce projet de WebX vise la mise en pratique des notions abordées en cours et en TP dans le cadre d'une production à réaliser en équipe de 4 étudiants (ou 3).
On souhaite réaliser un Web Service (*back-end* Jersey) et une application en ligne (*front-end* Vue.js) qui permettra à un utilisateur anglophone, supposé en vacances à l'étranger, de visualiser des informations pouvant accompagner son séjour touristique…
Plus précisément, en fonction de l'adresse IP qu'il utilise, l'application web détectera le pays où l'utilisateur se trouve et affichera au chargement de la page, une vue comportant une recette d'un plat typique, et plusieurs photos prises dans la région.
Cette application web devra s'appuyer sur une réponse JSON exposée par une API REST respectant le **schéma JSON** (équivalent JSON d'XSD :) que nous diffuserons sur Moodle.
Vous interrogerez un certain nombre de services REST à votre disposition :

- l'API REST "IP Geolocation API" :
  https://ipgeolocationapi.com/
  permet notamment de récupérer à partir de l'IP fournie, le nom du pays associé et une "bounding box" de celui-ci
  (cette API ne nécessitant pas d'authentification)
- l'API REST de TheMealDB :
  https://www.themealdb.com/
  permettant de rechercher des recettes de plats typiques et de les filtrant par pays : 25 pays sont référencés, https://www.themealdb.com/api/json/v1/1/list.php?a=list
  (l'auteur nous autorisant à utiliser l'API TheMealDB avec la clé "1")
- l'API REST de Flickr :
  https://www.flickr.com/services/api/
  permet notamment d'accéder à des photos géolocalisées en fonction de coordonnées géographiques ;
  (il faudra qu'au moins un membre de l'équipe crée un compte :
  https://www.flickr.com/signup
  et une non-commercial API key dédiée au projet :
  https://www.flickr.com/services/apps/create/apply/)

1. Exigences fonctionnelles

- *User Story 1a* (US1a) : implémentation d'un client Java qui, à partir d'une adresse IP *(pour connaître la vôtre facilement sous Linux, faire* « curl https://ipinfo.io/ip »*)* renvoie (au moins) le nom de ce pays en anglais, son code en ISO 3166-1 alpha-2, et sa bounding box {min_latitude, max_latitude, min_longitude, max_longitude}
- *User Story 1b* (US1b) : implémentation d'un client Java qui, à partir de l'adjectif en anglais qualifiant une région particulière (p.ex. l'adjectif "Canadian" pour le Canada, cf. https://www.themealdb.com/api/json/v1/1/list.php?a=list) renvoie les informations d'une recette choisie parmi celles références dans cette région ; ou alors si l'adjectif de lieu est manquant ou non reconnu, renvoie les informations d'une recette aléatoire.
- *User Story 1c* (US1c): implémentation d'un classe Java qui, à partir du code d'un pays en ISO 3166-1 alpha-2 (e.g. "FR"), renvoie le nom du pays en anglais (e.g. "France") et l'adjectif en anglais qui le caractérisera dans TheMealDB (e.g.

"French"). Une exception pourra être levée pour un pays non référencé dans TheMealDB.

- *User Story 1d* (US1d) : implémentation d'un client Java qui, à partir d'une bounding box {min_latitude, max_latitude, min_longitude, max_longitude}, renvoie une liste de 5 URL de photos hébergées sur Flickr prises dans des lieux inclus dans la boîte englobante.
- User Story 2 (US2) : implémentation d'un Web Service REST avec Jersey produisant une réponse JSON (conforme au schéma JSON fourni dans l'annexe A) à partir des paramètres reçus (un nom de pays pouvant être omis, lorsque le nom du pays est censé être récupéré à partir de l'adresse IP du client).
  En particulier, la réponse devra 5 URL de photos prises dans la région considérée et hébergées sur Flickr, et les détails d'une recette typique du pays sélectionné choisie aléatoirement (ou à défaut, d'une recette sélectionnée aléatoirement parmi toutes les recettes, ce cas de figure étant signalé par un booléen).
- *User Story* 3 : implémentation d'une application en ligne exploitant le Web Service de l'US2, et dont le rendu comporte toutes les informations renvoyées par l'US2. En particulier, elle permettra de choisir un pays ou de laisser celui-ci être détecté automatiquement.

2. Exigences extra-fonctionnelles

2.1. Environnement technique, méthodologie de développement

Vous devrez utiliser les technologies vues en cours :

- Pour toutes les API REST consommées dans l'US1*, si plusieurs représentations de ressources sont disponibles, vous privilégierez **JSON** plutôt que XML.
- Le langage principal sera **Java 11** (pas de PHP ni de Python…)
- Le Web Service correspondant à l'US2 sera implanté en suivant le style d'architecture REST à l'aide du framework JAX-RS **Jersey**, en utilisant l'archetype « org.glassfish.jersey.archetypes : **jersey-quickstart-grizzly2** : **3.0.1** ».
- Le client pour la partie front-end correspondant à l'US3 sera réalisé avec **Vue.js** en exploitant la librairie **Axios**.

D'un point de vue technique, la consommation des Web Services devra être illustrée à travers des **tests unitaires et d'intégration** écrits à l'aide d'un framework dédié (JUnit 4.13). Vous testerez notamment que les sorties produite par votre WS sont conformes au JSON Schema fourni dans l'annexe A, en vous appuyant sur la dépendance Maven « **org.leadpony.justify : justify : 2.0.0** ». Vous ne vous contenterez pas de tester la marche normale mais vous illustrerez la gestion des erreurs que vous mettrez en place.

Apache Maven devra être utilisé pour structurer votre projet, gérer les dépendances et faciliter le lancement des tests du *back-end* (US1 et US2).

Il n'est pas demandé de réaliser des tests unitaires pour le *front-end* en Vue.js.

Tout choix technologique qui sort de ce cadre devra être discuté avec vos encadrants de TP, en particulier si vous souhaitez utiliser des librairies javascript spécifiques.

L'utilisation de **Git** et de GitHub est exigée pour faciliter le suivi de version pour ce projet collaboratif.

Un malus sera attribué si la qualité du code écrit est insuffisante; il est attendu un code lisible, bien indenté, avec des commentaires pertinents et des méthodes et des classes documentées.

Votre dépôt Git devra contenir deux sous-dossiers dédiés pour le *back-end* et le *front-end* (donc le fichier *pom.xml* du *back-end* ne se trouvera pas à la racine du dépôt, mais dans un sous-dossier, similairement au dépôt GitHub du TP3).

Le fichier **README.md** à la racine du dépôt GitHub devra contenir :

- Les prénom/nom des membres du groupe, associés aux noms d'utilisateur GitHub
- une section documentant les étapes à exécuter pour lancer votre projet (e.g. avec **mvn** et **npm**).

Pour constituer les équipes du projet WebX, vous devez suivre les instructions figurant sur la page Moodle du cours.
En résumé, le premier membre du groupe procède à la création d'une équipe en choisissant un nom de team GitHub Classroom commençant par "**webx-**" puis les autres membres du groupe peuvent rejoindre l'équipe existante et accéder au dépôt GitHub privé qui commence par https://github.com/UE-WebX/… (ce dépôt étant créé pour la durée du semestre).
Enfin, l'un des membres complète le **README.md** de votre projet privé, en mentionnant le Prénom/NOM/TPAnn/nom-d'utilisateur-GitHub de chaque membre.
N'oubliez pas de vous ajouter comme **«watcher»** au projet pour recevoir les notifications ultérieures.

2.2. Livrables

Item	**Format**	**Livraison sur**	**Date limite**
constitution d'une d'équipe WebX	README.md	GitHub	avant le : dimanche 4 avril 2021 à 23h
code source du projet : rendu intermédiaire (avec l'US1 fonctionnelle)	Git	GitHub : un tag "**1.0**" créé pour la livraison (**git tag -a 1.0 -m 1.0**)	avant le : samedi 17 avril 2021 à 23h
code source du projet : rendu final	Git	GitHub : un tag "**2.0**" créé pour la livraison	**avant le : samedi 24 avril 2021 à 23h**
rapport	PDF	Moodle	avant le : **dimanche 25 avril 2021 à 23h**

Votre rapport comportera nécessairement les éléments suivants :

- un diagramme de classes de votre application Java ;
- pour chacune des *user stories*, vous expliquerez comment elle est adressée par votre projet (quel moyens techniques ? quel(le)s packages/classes ? et si applicable, quelle représentation JSON/XML et méthodologie de *parsing* retenues), une estimation en % du niveau d'aboutissement de la solution proposée et en particulier si tout est fonctionnel ou pas (max. 5 pages). Des diagrammes de séquence ou d'utilisation[1] pourront être pertinents pour illustrer votre propos ;
- vous présenterez votre politique de tests (unitaires, intégration, etc), leur nombre ainsi que les verdicts obtenus (max. 2 pages) ;
- vous indiquerez les difficultés rencontrées ainsi que la façon dont vous avez procédé pour y remédier (donnez les liens vers les docs pertinentes que vous avez pu trouver, les liens vers les discussions que vous avez pu initier sur des forums spécialisés ou sur Moodle ; max. 2 pages).

Il est attendu un rapport **soigné** sur la présentation, l'orthographe et la grammaire. Le code est essentiel mais le rapport ne doit pas être négligé ou bâclé car, pour le correcteur, il est la porte d'entrée de votre projet.

3. Conseils et remarques diverses

- Dans ce projet, vous serez évalués sur la mise en œuvre des connaissances et des compétences relatives à l'UE WebX.
- En cas de question, **vous privilégierez l'utilisation du salon textuel Discord commun à toute la promotion (#webx)** à l'envoi d'un e-mail aux enseignants.
- Certaines APIs pourraient ne pas renvoyer le bon type MIME, dans ce cas l'interface JAX-RS ClientResponseFilter pourra être utile : https://stackoverflow.com/q/21380475
- Durant la **séance de TP consacrée au projet** (organisée dans Discord), vos enseignants seront disponibles pour toute question et aide dont vous auriez besoin. Vous pouvez commencer par les tâches qui vous semblent les plus pertinentes (créer des issues / se répartir les User Stories / etc.) mais nous vous suggérons en tout cas de **commencer par créer le projet backend dans IntelliJ avec le bon archetype**[2], et vérifier que tous les membres de l'équipe peuvent compiler le projet.
- Ne vous laissez pas tenter par un plagiat entre équipes. Lors de la correction, des outils de mesure de similarité de code seront utilisés. Il y a quelques années, un cas de plagiat en M1 Informatique "Développement Logiciel" a conduit la section disciplinaire du conseil d'administration de l'Université Paul Sabatier à prononcer une exclusion de l'Université de 24 mois (dont 3 mois ferme) pour les étudiants fautifs.
- Le barème sera le même pour tous, que le projet soit fait par une équipe de 3 ou bien de 4.

[1] On pourra les réaliser à l'aide de l'outil PlantUML : https://plantuml.com/fr/sequence-diagram

[2] « org.glassfish.jersey.archetypes : **jersey-quickstart-grizzly2** : **3.0.1** ».