Contexte et objectifs

Au livrable 2, vous avez implémenté les rudiments d'une application dorsale ("backend") qui permettait de servir à la partie "front-end" (navigateur web) les données météorologiques pour une journée historique donnée ainsi que les prévisions pour les prochains jours.

Ce livrable sera composé de trois parties. La première partie consistera à intégrer une API REST pour l'accès aux données au niveau "back-end". La seconde partie consistera à implémenter un système de mise en cache des données au moyen de MongoDB. Finalement, la troisième partie consistera à développer une nouvelle fonctionnalité dans le front-end qui permettra d'afficher les données et prévisions météo pour différentes villes canadiennes sur une carte.

Note importante: Pour ce livrable, et tel que réalisé pour le livrable 2, nous ne nous intéressons qu'aux villes canadiennes reliées à un aéroport selon le fichier `station_mapping.json` (fourni avec le livrable 2).

Objectifs et tâches

Les objectifs et tâches sont les suivants:

T1: API REST

L'API REST de votre back-end devra proposer une interface structurée pour accéder aux différentes données météo, en suivant les principes de design d'APIs REST vus en cours. Vous devez notamment structurer les différents éléments du domaine conceptuel en collections et éléments structurés. Vous devrez utiliser les verbes adéquats pour les différentes opérations (e.g., `GET`, `POST`, `PUT`, etc.). Le front-end va consommer les APIs REST exposées par le back-end.

Note 1: Il est possible que vous ayez déjà commencé à implémenter une API REST au livrable 2 tel qu'indiqué dans l'énoncé -- bien que cela n'était pas évalué, si c'est le cas, vous aurez pris un peu d'avance pour ce livrable.

Note 2: Vous êtes libres de choisir l'interface exacte des URLs de l'API REST -- nous n'imposons pas de schéma particulier. Toutefois, cette interface devra respecter les principes que nous verrons sous peu en classe (en particulier, les *verbes*, *noms*, et la structure arborescente) -- vous aurez à la décrire dans votre rapport.

T2: Cache structuré des requêtes à l'API

T2.1: En plus de proposer une interface REST pour l'accès aux données météo, vous devrez également mettre en cache les requêtes à l'API effectuées par l'usager afin d'éviter de surcharger les back-ends d'Environnement Canada (apis CSV et XML). Le cache ici devra être **stocké et structuré dans une base de données MongoDB** qui sera persistante et organisée logiquement.

T2.2: Si une requête est reçue, et que son résultat est disponible en cache, et que le résultat n'est pas *expiré*, vous devez retourner le résultat en cache. Si le résultat est expiré, vous devez rafraîchir l'élément via le back-end d'Environnement Canada (CSV ou XML).

T2.3: Tout élément placé en cache devra être conservé pour une durée déterminée maximale selon sa nature:

- Données météo historiques (flux CSV): 24h
- Données météo actuelles et prévisions (flux XML): 1h

Note: Plusieurs éléments peuvent être mis à jour simultanément si cela s'avère pratique (et ce, même avant l'expiration de la durée maximale spécifiée). Par exemple, la requête XML auprès du service d'Environnement Canada pour obtenir les prévisions pour la journée actuelle retournera également les prévisions pour les prochains jours -- il serait donc logique de mettre à jour ces informations dans le cache également.

T3: Affichage des données météo sur une carte

Dans cette section, vous devez utiliser une API tierce partie pour faire apparaître une carte géographique qui affichera les conditions météo actuelles ou les prévisions pour les prochains jours (tâche T2 du livrable 2).

Recommandation: nous recommandons l'utilisation du service de cartographie "Leaflet" (https://leafletjs.com) qui est très simple d'utilisation et très flexible. Leaflet peut fonctionner avec plusieurs fournisseurs de cartes ("TileLayer") tels que MapBox ou OpenStreetMaps. Plusieurs exemples et tutoriels sont fournis pour vous aider à comprendre leur API. Les exemples utilisent par défaut le fournisseur "MapBox" -- vous aurez toutefois besoin de vous créer un compte pour obtenir une clé API gratuite (https://www.mapbox.com/studio/account/tokens/), qui permet un volume de requêtes assez élevé. *Nous vous encourageons à utiliser Leafjs; cependant, notez que vous êtes libres d'utiliser le service cartographique et l'API de votre choix (ex., Google Maps, etc.) pour ce livrable, tant que ce dernier vous permet de réaliser l'ensemble des exigences demandées.*

T3.1 Vous devez ajouter un nouvel onglet à votre interface, qui permettra d'accéder à la vue des prévisions sous forme cartographique.

T3.2 Un ensemble de liens ou une liste défilante doit permettre de sélectionner entre les informations météo actuelles et une prévision pour les prochains jours/nuits.

T3.3 La carte devra afficher, pour l'ensemble des villes (les mêmes villes que pour les autres livrables), la température actuelle ou la température correspondant aux prévisions (*T3.2*, selon l'option choisie par l'utilisateur). Cette information devrait être affichée sur la carte à l'emplacement géographique de la ville/station météo (la latitude et la longitude sont disponibles dans les fichiers CSV). *Note: la sélection d'une station météo à gauche n'aura donc aucune incidence sur la fonctionnalité cartographique.*

T3.4 Un clic sur la température d'une ville doit faire apparaître une infobulle qui affichera les conditions détaillées ou la prévision détaillée.

T3.5 La carte doit être initialement centrée pour afficher l'ensemble des villes canadiennes.

Note: l'affichage des données pour l'ensemble des villes de manière simultanée sur la carte sous-entend que vous devrez obtenir les données de votre back-end pour l'ensemble les villes. Bien évidemment, votre back-end devra initialement obtenir les données du back-end d'Environnement Canada, mais puisque les données seront en cache, les requêtes suivantes devraient plutôt retourner les données en cache.

Utilisation de cadriciels

Bien que les livrables puissent être totalement accomplis avec les technologies web standard (HTML/CSS/VanillaJS), nous autorisons et encourageons l'utilisation des cadriciels web si vous le désirez. Dans ce cas, il sera de votre responsabilité de vous documenter sur le cadriciel choisi et de l'utiliser selon les bonnes pratiques, et de résoudre les problèmes éventuels rencontrés. Le chargé de laboratoire tentera de vous aider au meilleur de ses connaissances, mais il est possible qu'il ne connaisse pas précisément tous les cadriciels web.

Rapport

Un rapport est demandé. Vous devez discuter des points suivants (maximum 8 pages). Une pénalité sera appliquée pour les fautes de français (voir le barème) et une mise en page incorrecte ou un manque de rigueur dans la présentation.

R1: Décrivez en détail les choix de conception effectués pour votre API REST (pour répondre aux différentes fonctionnalités demandées) (total 10 points). Vous devez notamment décrire:

- Les verbes et noms des différentes ressources (3 points)
- La structure arborescente (collections) (2 points)
- Le ou les formats de sortie (1 points)
- Une justification pour vos choix, et les limites potentielles (4 points)

R2: Décrivez le schéma et la structure de votre base de données MongoDB (total 8 points). Vous devez notamment décrire:

- Les collections pour les différents requis du TP et la structure des documents. Vous pouvez le faire à travers un diagramme (4 points).
- Justifiez vos choix de conception et les limites potentielles. (4 points)

R3: Décrivez les modifications apportées à l'architecture logicielle de votre back-end et de votre front-end par rapport au livrable précédent (total 10 points). Décrivez notamment:

- Illustration de l'architecture de votre application à travers un/des diagrammes de classes, de paquetages ou de composants (4 points);
- L'organisation et le rôle des éléments principaux (classes, fonctions) de votre code JavaScript en lien avec vos diagrammes (3 points).
- Justifiez vos choix de conception et les limites potentielles. (3 points)

R4: Comment procédez-vous pour retirer ou mettre à jour de manière périodique les entrées périmées du cache sur le back-end? (4 points)

R5: De quelques façon avez-vous subdivisé les tâches en équipe pour ce TP? Décrivez le rôle et les tâches assignées à chacun des membres. (4 points)

R6: Notez qu'une brève introduction et conclusion est également demandée. (4 points)

Code source

Bien que le code ne soit pas évalué, nous nous réservons quand même le droit de le vérifier, et **le code complet devra être soumis avec la remise**. Toutefois, nous nous attendons à ce que vous mettiez des bonnes pratiques de conception et d'implémentation en oeuvre. L'évaluation de cet aspect sera faite de manière indirecte par l'évaluation de vos réponses aux questions du rapport qui traitent de ces aspects. De plus, le livrable 3 vous demandera d'ajouter encore une fois un nouvel ensemble de fonctionnalités et de *refactorer* votre code, ce qui sera plus facile à réaliser avec une bonne conception/implémentation.

Utilisation d'un entrepôt Git

Pour le projet de session, vous devez utiliser un entrepôt Git (GitHub ou GitLab disponible à l'ÉTS) pour votre projet. Tous les membres de l'équipe devraient "pousser" du code pour étayer la contribution de tous et chacun. Advenant le cas où nous recevions une plainte concernant un membre de l'équipe ne fournissant pas sa juste part de travail, nous pourrions vous demander de nous donner accès à votre entrepôt ou de nous fournir une copie des "logs" (bien entendu, nous espérons fortement que cela ne se sera pas nécessaire!).

Barème de correction

Tâche	Description	Points max.
T1	API REST back-end (évalué surtout par le rapport)	Total 5
T2	Cache structuré des requêtes à l'API	Total 12
T2.1	Cache des requêtes API STM MongoDB	5
T2.2	Retourner/rafraîchir le résultat en cache	4
T2.3	Durée de conservation respectée	3
T3	Affichage des données météo sur une carte	Total 23
T3.1	Nouvel onglet pour vue cartographique	3
T3.2	Choix entre infos actuelles ou prévisions	4
T3.3	Affichage de la carte	9

T3.4	Infobulle avec informations détaillées	4
T3.5	Carte centrée sur l'ensemble des villes	3
Rapport		Total 40
R1	(voir la description)	10
R2	(voir la description)	8
R3	(voir la description)	10
R4	(voir la description)	4
R5	(voir la description)	4
R6	Introduction et conclusion	4
Pénalité	Code JS mal structuré (-10)	
Pénalité	Code JS dans fichier HTML (-5)	
Pénalité	Pratiques non-recommandées (ex., insertion de code brut HTML dans le DOM) (-5)	
Pénalité	Mise en page HTML/CSS mal structurée (ex., utilisation de tableaux) (-5)	
Pénalité	Des fonctionnalités du livrable précédent ne fonctionnent plus (-10)	
Pénalité	Fautes de français dans le rapport (-0.5% par faute, jusqu'à -10%)	
Total		Max. 80

Politique sur le plagiat

Tel qu'énoncé dans le plan de cours:

Les clauses du « Règlement sur les infractions de nature académique de l'ÉTS » s'appliquent dans ce cours ainsi que dans tous les cours du département. Les étudiants doivent consulter le Règlement sur les infractions de nature académique (https://www.etsmtl.ca/A-propos/Direction/Politiques-reglements/Infractions_nature_academique.pdf) pour identifier les actes considérés comme étant des infractions de nature académique ainsi que prendre connaissance des sanctions prévues à cet effet. À l'ÉTS, le respect de la propriété intellectuelle est une valeur essentielle et les étudiants sont invités à consulter la page Citer, pas

plagier ! (https://www.etsmtl.ca/Etudiants-actuels/Baccalaureat/Citer-pas-plagier).

Nous attirons votre attention sur le fait qu'il est **interdit de réutiliser, en tout ou en partie**, le travail d'une autre équipe ou réalisé à une session antérieure. Nous conservons l'ensemble des travaux des sessions antérieures et avons accès au code source soumis, et nous nous réservons le droit d'utiliser des outils de comparaison du code source et de détection de plagiat. Tout plagiat détecté sera sanctionné selon les politiques en vigueur à l'ÉTS.

Note: si vous reprenez le cours GTI525, vous devez également refaire les laboratoires.

Procédure et date de remise

Les instructions de remise sont sur Moodle (ENA).