TP9-20

Visualisation de statistiques sur les stations-service françaises

Introduction

L'objectif de ce projet est de développer une application permettant d'analyser et de présenter des statistiques sur les prix des carburants en France, en utilisant des données en Open Data au format JSON. L'application doit offrir aux utilisateurs la possibilité de sélectionner différentes granularités (ensemble de départements ou de régions) ainsi que des types spécifiques de carburants pour obtenir des statistiques pertinentes.

Fonctionnalités

Sélection de Granularité

L'utilisateur doit pouvoir choisir entre deux granularités principales : un ensemble de départements ou un ensemble de régions.

Sélection de Types de Carburants

L'utilisateur doit pouvoir sélectionner un ou plusieurs types de carburants parmi ceux disponibles dans les données conformes au schéma JSON fourni. L'application doit optionnellement donner, pour chaque type de carburant sélectionné, l'adresse du moins cher dans chaque département ou région (selon granularité choisie).

Sélection de Statistiques

L'application doit permettre à l'utilisateur de choisir un ou plusieurs types de statistiques parmi les suivantes :

- Prix moyen de chaque type de carburant selon la granularité choisie.
- Prix médian de chaque type de carburant selon la granularité choisie.
- Prix minimum de chaque type de carburant selon la granularité choisie.
- Nombre de stations qui proposent chaque type de carburant selon la granularité choisie.
- Nombre de stations qui proposent des services spécifiques (sélectionnés par l'utilisateur parmis ceux mentionnés dans le fichier JSON : Station de gonflage, Boutique alimentaire, Lavage automatique, Wifi, Bornes électriques, etc.).

Pour chaque statistique sélectionnée, l'utilisateur doit pouvoir choisir le type de diagramme qu'il souhaite parmi ceux disponibles (Camemberts, digrammes barre, diagrammes colonnes, etc.).

Génération de Rapports

L'application doit générer des rapports sous forme de pages web contenant les informations et résultats des statistiques sélectionnées. Les rapports doivent être clairs, informatifs et facilement compréhensibles.

Autres fonctionnalités

Afin de vous démarquer des équipes concurrentes, vous êtes libres d'ajouter de nouvelles fonctionnalités originales à cette application.

Interface Utilisateur (UI)

L'interface utilisateur doit être conviviale et intuitive. Les options de sélection de granularité, de types de carburants et de statistiques doivent être présentées de manière claire. L'utilisateur doit pouvoir naviguer facilement à travers l'application.

Contraintes Techniques

L'application doit être développée en utilisant le langage de programmation Java. Elle devra impérativement utiliser la bibliothèque "visustats" développée aux TP précédents pour toutes les visualisations diagrammatiques (Camemberts, digrammes barre, diagrammes colonnes etc.).

L'Interface Utilisateur sera programmée en Java avec le framework Swing.

Concernant la source de données, dans un premier temps on utilisera le fichier :

`https://data.economie.gouv.fr/api/explore/v2.1/catalog/datasets/prix-carburants-fichier-quotidien-test-ods/exports/json?lang=fr&timezone=Europe%2FBerlin`

On ne téléchargera ce fichier qu'une seule fois. Tous les tests seront effectuées avec cette source de données.

Dans un second temps, au lieu de prendre les informations ce fichier JSON téléchargé, on utilisera pour obtenir les données en temps réel l'API REST :

`https://data.economie.gouv.fr/explore/dataset/prix-des-carburants-en-france-flux-instantane-v2/api/`

L'équipe de développement est encouragée à utiliser des outils comme ChatGPT ou Copilot pour accélérer le développement. La contrainte reste que vous **devez parfaitement** comprendre le code que vous obtenez par ce biais. Cela sera évalué en fin de projet.

Livrables Attendus

- Code source de l'application et de ses tests dans le GitLab de l'ISTIC.
- Fichier .jar exécutable contenant toute l'application.
- Documentation technique détaillée expliquant l'architecture à l'aide de diagrammes UML (réalisés par exemple avec `https://dotuml.com/index.html`), les choix de conception et les dépendances vis-à-vis de librairies externes.
- Bref manuel utilisateur expliquant comment installer et utiliser l'application.

Mode de Développement

Le projet sera réalisé en utilisant une approche agile avec la méthodologie SCRUM. Les itérations seront planifiées, et les retours réguliers seront pris en compte pour améliorer le produit.

Équipe de Développement

L'encadrant de TP jouera le rôle de client de cette application.

L'un ou l'une des étudiantes de l'équipe jouera le rôle de *Product Owner*, responsable de définir la vision du produit (en lien avec le client), de prioriser le backlog du produit (liste des fonctionnalités à développer), et de prendre des décisions sur les fonctionnalités à implémenter.

Un autre membre de l'équipe jouera le rôle de *Scrum Master*, responsable de s'assurer que l'équipe suit les principes et les pratiques SCRUM. Tout en contribuant activement au développement, il élimine les obstacles, facilite les réunions SCRUM, et aide l'équipe à atteindre ses objectifs.

L'ensemble de l'équipe de développement est responsable de la réalisation des fonctionnalités définies dans le backlog du produit. Elle est auto-organisée, et se fixe des objectifs pour chaque itération (sprint). La qualité du logiciel doit être une priorité dès le début, avec une architecture de tests mise en place dès la première itération.

Planning

Le développement de l'application doit respecter le planning suivant :

Sprint 1 (Semaines 11-13) : Établir les Fondations
- Mettre en place l'environnement de développement et de test.
- Rédiger les *user stories*.
- Élaborer l'architecture de base de l'application.
- Implémenter le chargement initial d'un sous-ensemble simplifié des données JSON.
- Mettre en place une interface utilisateur (UI) de base, avec par exemple la mise en œuvre de la sélection de granularité et la sélection des types de carburants.
- consolider le package visustats en implémentant les éventuels diagrammes manquant
- Rétrospective de l'équipe pour évaluer le processus SCRUM.

Sprint 2 (Semaines 14-16) : Implémentation des fonctionnalités de base.
- Ajouter à l'UI les fonctionnalités pour configurer les statistiques de base.
- Implémenter le chargement complet des données JSON.
- Intégrer la génération de rapports de base.
- Rétrospective de l'équipe pour évaluer le processus SCRUM.

Sprint 3 (Semaines 18-20) : Finalisation et Livraison
- Finaliser les fonctionnalités restantes.
- Utilisation de l'API REST temps réel au lieu du fichier JSON comme source de données.
- Effectuer des tests finaux et résoudre les problèmes.
- Améliorer la documentation technique et le manuel utilisateur.
- Démonstration finale du produit au client.
- Rétrospective de l'équipe pour évaluer le processus SCRUM.

Remarques :
- Certaines fonctionnalités plus avancées peuvent être simplifiées ou reportées pour les itérations suivantes.
- La communication et la collaboration rapides entre les membres de l'équipe seront cruciales pour respecter ce calendrier.
- La flexibilité est toujours essentielle pour s'adapter aux changements et aux retours d'expérience.
- Le Sprint 2 se fera en grande partie en autonomie (semaines 14 et 16 non encadrées). C'est un test du sérieux et de la motivation de l'équipe.

Validation et Acceptation

L'application sera soumise à une phase de validation interne par l'équipe de développement, suivie d'une phase d'acceptation par le client lors d'une démonstration formelle à la fin du projet. Une évaluation individuelle de performance sera aussi réalisée par l'encadrant de TP la dernière semaine du projet.