LOG430 Architecture logicielle

Objectif principal

L'objectif principal de ce projet de cours est de proposer un système de comparaison de temps de trajet basée sur une architecture de microservices. Ce système permettra de comparer les temps de trajets entre les autobus et les automobiles. Il sera réalisé au travers d'un seul laboratoire de 36 heures au total. Les spécifications des exigences portent sur l'analyse, la conception et l'implantation de ce système. Ce projet sera réalisé selon un seul livrable, qui devra être validé régulièrement auprès du chargé de laboratoire. Ce livrable comprendra les objectifs d'affaires, les cas d'utilisation, les scénarios de qualité priorisés, des vues architecturales permettant de démonter chacune des tactiques, une ou plusieurs vues de modules, une ou plusieurs vues composant et connecteur, une ou plusieurs vues d'allocation. Vous devrez par la suite faire faire une révision par les pairs (ATAM) sur votre architecture par une autre équipe et répondre aux préoccupations de celle-ci concernant votre architecture.

Le contexte de l'organisation

Vous êtes nouvellement embauché par l'organisation LOG430STM pour développer le système de comparateur de trajet. La réussite de ce projet n'est pas optionnelle. La carrière des étudiants peut grandement être impactée s'ils échouent ce cours. C'est pour cette raison que l'organisation a décidé de séparer les responsabilités selon les différentes parties prenantes.

Le comparateur de trajet

Vous devez développer l'architecture d'un système de comparaison de temps de trajet en utilisant les données temps réel de la STM disponible sur le site suivant: https://www.stm.info/fr/a-propos/developpeurs et les donnés fournie par des services externes

Services externes

Utiliser l'API ou la page web des fournisseurs externe pour l'estimation en temps réel du temps nécessaire pour parcourir la distance entre deux points correspondant aux points de départ et point d'arrivée d'un trajet d'autobus. L'estimation doit se faire en empruntant exactement le même trajet que l'autobus.

Exemples

- https://www.google.ca/maps/dir/Marché+Jean-Talon/45.51433,+-73.55014/45.58927,+-73.50912/@45.5503686,-73.5965441,13z/data=!3m2!4b1!5s0x4cc919136bb582b5:0xf0b087b71589c626!4m20!4m19!1m5!1m1!1s0x4cc919136130849d:0x5c1098d838d87981!2m2!1d-73.6148902!2d45.5361095!1m5!1m1!1s0x0:0xf12664c8f968830d!2m2!1d-73.5502413!2d45.514437!1m5!1m1!1s0x0:0xfeb4b67d79e8fa7c!2m2!1d-73.5091582!2d45.589417!3e0

Fournisseurs potentiels

- google map
- https://www.viamichelin.com/web/Routes
- https://en.mappy.com/itineraire
- https://ca.bonnesroutes.com
- Waze
- BingMaps
- en.mappy.com
- Ajouter vos suggestions...

Parties prenantes du projet

Chargé de laboratoire (client)

- Effectuera l'évaluation en continu de l'architecture de chaque équipe (Documentation, Intégration et Implémentation)
- Responsable de répondre aux questions des étudiants (durant les périodes de laboratoire seulement)
- Responsable d'aider les étudiants à maîtriser les concepts d'architecture
- Veux un rapport détaillé de l'architecture et des interfaces

Classe

- Chaque classe est séparée en deux groupes

Groupe

- Chaque groupe est séparé en équipe de 5 étudiants.
- Chaque groupe est en compétition avec les groupes de toutes les classes.
- Les équipes de chaque groupe peuvent partager leurs microservices avec toutes les équipes du groupe pour permettre aux autres équipes de faire de l'intégration au lieu de faire de l'implémentation. Utiliser l'intégration pour satisfaire une exigence est plus payant en termes de point. Voir le ficher de la grille de correction.
- Le chargé de cours crée les groupes.

Équipe de coordination

- L'équipe de coordination est aussi responsable de répartir équitablement les tâches de réalisation de la conception et l'implémentation des différents microservices nécessaire à ce projet.
    - Dois conserver une trace écrite pour savoir quelle équipe implémente quel microservice.

- L'équipe a la responsabilité de valider et de diffuser la documentation des interfaces touchant aux composants implémentés/utilisés par plusieurs équipes.
    - Une version d'interface publiée ne peut pas être changée. Vous devez obligatoirement publier une nouvelle version.
- L'équipe de coordination peut démettre de ses fonctions un étudiant qui ne répond pas à ses attentes.
    - L'équipe affectée devra nommer un nouveau représentant
- Les équipes de coordination ne doivent pas travailler ensemble ils sont des compétiteurs

Équipe (Étudiants)

- L'équipe doit concevoir et réaliser une architecture qui satisfait toutes les exigences documentées dans ce document de spécification.
- Je recommande à l'équipe d'utiliser les microservices implémentés par les autres équipes puisque c'est 2 fois plus payant en termes de points accumulés. Voir la grille de pointage pour plus d'information.
- Un étudiant par équipe est nommé pour faire partie de l'équipe de coordination
    - Une équipe peut révoquer son représentant de l'équipe de coordination s'il ne répond pas à leurs attentes
- L'équipe doit conserver une traçabilité de quel étudiant est responsable de quelles tâches. Ceci correspond à une vue d'allocation à insérer dans votre rapport.
- L'équipe doit connaître en tout temps l'état d'une tâche assignée à un étudiant
- L'équipe a la responsabilité de concevoir/documenter et diffuser (à l'équipe de coordination) la documentation des interfaces des microservices qui leur ont été assignés.
- Le chargé de cours crée les équipes de laboratoires.

Exigences

Objectifs d'affaires

- OA-1. Faciliter le recrutement des nouveaux chargés de laboratoire.
- OA-2. Validez si le transport par autobus est toujours plus rapide, peu importe l'heure de la journée.

Exigences fonctionnelles

CU01. Veux comparer les temps de trajet.

1. Le (chargé de laboratoire) CL sélectionne une intersection de départ et une intersection d'arrivée, ainsi que le taux de rafraichissement de la prise de mesure.
2. Le CL sélectionne le service externe qu'il veut utiliser pour faire la comparaison des temps de trajet avec les donnés temps réel de la STM.
3. Le système affiche un graphique du temps de déplacement et met celui-ci à jour selon le taux de rafraichissement.

> **Cas alternatifs**
>
> - 2.a Service externe: Utiliser plusieurs services externes disponibles pour faire le comparatif.

CU02. Veux pouvoir mettre le chaos dans les services en mode.

- **Option 1**: Manuel 1.1. Le CL consulte la liste des microservices avec leur latence moyenne. 1.2. Le CL change la latence d'un ou plusieurs microservices.
- **Option 2**: Automatique 2.1. Le CL sélectionne le mode automatique tout en spécifiant la fréquence de la perturbation en seconde. 2.2. Le système détruit[1] un microservice de façon aléatoire a tous les x secondes.
- Le système conserve un log des différents changements apportés que nous pourrons utiliser pour vérifier les données accumulées.

> **Cas alternatifs**
>
> - 1.2.a Le CL détruit un ou plusieurs microservices.
> - 1.2.b Le CL détruit tous les microservices d'une équipe.

Exigences de qualité

- Vous devez réaliser un scénario de qualité pour chacun des attributs (A) de qualité et vous devrez concevoir et réaliser une architecture qui utilisera au minimum une tactique architecturale pour chacune des sous-catégories (SC) suivantes:

(A) Disponibilité

- (SC) détection de faute
- (SC) Préparation et réparation
- (SC) Réintroduction
- (SC) Prévention des fautes

(A) Modifiabilité

- (SC) Réduire la taille des modules
- (SC) Augmenter la cohésion
- (SC) Réduire le couplage
- (SC) Defer binding

(A) Performance

- (SC) Contrôler la demande en ressources
- (SC) Gérer les ressources

(A) Sécurité

- (SC) Détecter les attaques
- (SC) Résister aux attaques
- (SC) Réagir aux attaques
- (SC) Récupérer d'une attaque

(A) Testabilité

- (SC) Controle and observe l'état du système
- (SC) limiter la complexité

(A) Convivialité

- (SC) Supporter l'initiative de l'usager
- (SC) Supporter l'initiative du système

(A) Interopérabilité

- (SC) Localiser
- (SC) Gérer les interfaces

Contraintes de réalisation

Language¶

Nous n'imposons aucune contrainte au niveau du langage de développement utilisé.

Réalisation

Vous devez réaliser votre projet avec des microservices

Contrainte d'équipe

L'équipe de coordonnateur peut imposer aux équipes les contraintes quelle juge nécessaires pour le bon déroulement du projet.

Grille de pointage

Voir la grille de pointage pour connaître le nombre de points associé à chacun des artéfacts que vous réaliserez durant ce projet.

Le nombre normal d'étudiants dans une équipe est de 5 personnes.

La note finale de chaque équipe sera pondérée en fonction du nombre d'étudiants dans l'équipe.

La note finale de chaque équipe sera pondérée en fonction des notes de toutes les équipes de toutes les classes.

Directive de planification

L'équipe de coordination répartit la charge de travail au niveau des équipes et indique clairement (avec document à l'appui) ses attentes par rapport à chaque équipe.

Je vous recommande d'utiliser un outil de suivi des tâches.

- Trello
- Github issue
- todoist
- etc.

- Notion task list

Directive de documentation de l'architecture

Toutes les équipes d'une classe doivent utiliser les mêmes outils de documentation. L'équipe de coordination est responsable de gérer l'utilisation des outils, de définir et gérer le processus de réalisation de la documentation.

Suggestion d'outils pour la documentation

notion.so

https://notion.so/[1] ÉTAPES POUR ÊTRE ENREGISTRÉ DANS LA DOCUMENTATION NOTION

1. Créer un compte notion avec votre adresse Google de l'ÉTS @etsmtl.net
2. Associer votre compte notion à un compte équipe 2.0. pour complété l'étape 2 entrée sur l'interface de notion authentifier avec votre compte google 2.1. sélectionner l'onglet Settings & Members dans notion 2.2. Cliquer l'onglet Plans 2.4. Scroll passer la table de prix 2.3. Cliquer le bouton Activate Student plan
3. Mettre votre courriel etsmtl.ca ici pour qu'on vous ajoute à l'espace de documentation

confluence

confluence.com

Google doc

googledoc.com

markdown

avec le plugin plantuml ou c4-plantuml

structurizer

https://structurizr.com[1]

Eclipse Papyrus

https://www.eclipse.org/papyrus/[1]

Directive d'implémentation

Chaque équipe doit implémenter sa propre solution tout en réalisant l'intégration avec des microservices développés par d'autres équipes.

La seule contrainte est que ces composants doivent être déployés par l'équipe propriétaire ou l'équipe de coordonnateur.

La coordination et l'échange d'information entre les équipes deviennent cruciaux pour le bon succès de votre projet.

Vous devrez clairement démontrer comment vous faites l'intégration des composants développés par les autres équipes. L'utilisation de diagramme de séquence et de diagramme de composant est particulièrement adaptée à ce besoin.

Directive de déploiement

Vous pouvez déployer votre solution sur n'importe quel serveur. Dans le cadre du laboratoire nous vous fournirons l'accès à un serveur virtuel. Ce serveur vous permettra de déployer des microservices réalisés à l'aide de docker et docker-compose.

Directive de Démonstration

- Vous n'aurez droit qu'à une seule démonstration pour l'intégration et/ou l'implémentation de chaque exigence.
- Chaque équipe disposera d'un maximum de 10 minutes par démonstration/exigence.
- Vous pouvez demander à faire une démonstration à n'importe quelle séance.
- Plus vous faites votre démonstration tôt durant la session plus cela sera payant pour votre équipe. Voir la colonne pourcentage dans la grille de pointage.
- Donc soyez bien préparé
    - Assurez-vous d'avoir testé vos microservices individuellement et dans le système
    - Assurez-vous de ne pas faire des modifications de dernières minutes qui pourraient impacter votre démonstration
- À chaque démonstration, le chargé de laboratoire peut vous demander de créer des issues que vous devrez avoir satisfaits lors de la démonstration subséquente.

Directives de vérification de la documentation

Une attention particulière sera portée sur les éléments suivants au niveau de votre documentation d'architecture:

1. Le stéréotype de chaque élément est bien identifié
2. Les interfaces sont explicites dans les diagrammes et documentées en fonction du type d'interface.
3. Chaque interface doit être adéquatement détaillée dans un fichier séparé.
4. Les choix technologiques sont visibles dans les vues architecturales.
5. Chaque diagramme possède un texte explicite pour le décrire. Ne décrivez pas chaque élément du diagramme, vous le ferez dans le tableau des éléments. Nous voulons savoir à quoi sert ce diagramme, quelle est son utilité, qu'est ce qu'il nous permet de comprendre ou démontrer.
6. Chaque diagramme possède une légende
7. La relation entre les vues est facilement compréhensible.
8. Les relations entre les cas d'utilisation et les éléments de votre architecture sont bien documentés.
9. La relation entre les scénarios d'attributs de qualité et les tactiques de votre architecture sont bien documentées.
10. La relation entre les tactiques et les éléments de votre architecture sont bien documentés.
11. Les tactiques sont clairement visibles et bien documentées dans les vues architecturales.
12. Les propriétés associées aux tactiques sont bien documentées.

13. Vous utilisez des liens pour toute référence à de l'information se trouvant dans le document
14. Votre rapport contient au moins une vue de module.
15. Votre rapport contient au moins une vue de C&C.
16. Votre rapport contient au moins une vue d'allocation.
17. Assignation des tâches
18. Déploiement
19. Votre rapport contient des diagrammes de séquence/activité pour démontrer le fonctionnement des composants dans la réalisation des différentes tactiques.
20. Vous vous êtes assuré de la correspondance entre la documentation d'architecture et votre implémentation.

Directives pour la remise

Toutes les remises se font directement sur le répertoire Github de votre équipe, sur la branche principale [«main»]. Assurez-vous que votre rapport est situé dans le répertoire DOC, qu'il est au format PDF et se nomme documentationArchitecture.PDF .

Vous devez mettre votre documentation et vos sources à jour dans la branche main, et ensuite vous générer un tag correspondant à la semaine ou vous faites votre remise.

Semaine	Tag
2	git tag semaine2
3	git tag semaine3
n	git tag semainen

Parasites et mollasson

Les membres de l'équipe devront réaliser une évaluation par les pairs pour chacun des membres de l'équipe. La note finale du laboratoire de ce membre sera pondérée par rapport à cette évaluation. Référez-vous à l'article Parasites et mollasson pour vous aider à faire l'évaluation des autres étudiants. Voir les fichiers EvaluationParLesPairs-etudiantN.xls .

Bonus projet laboratoire (5% point bonus)

Impressionnez-nous en intégrant de nouvelles fonctionnalités / Apis offrant de nouveaux services ou interagissant avec de nouveaux services externes.