1.*Nous voulons développer un service pour gérer les études supérieures du GIGL. Mettez les styles SOA dans l’ordre du meilleur au pire*. **REST, RPC, SOAP**. REST : 4 méthodes (GET, PUT, POST, DELETE), 3 ressources (étudiants, cours, directeurs). RPC : possiblement 12 méthodes pour chaque combinaison. SOAP : une possibilité comme REST, car le type de la ressource peut être utilisé pour distinguer entre les fonctionnalités (surcharge). add(Student s). add(Course c).\***REST** : style architectural pour créer des services web. Basé essentiellement sur http. Utilisé pour accéder aux ressources disponibles sur le web. Toutes les ressources ont un identificateur unique. On peut accéder à une ressource spécifique ou à une collection de ressources par une URL. On peut appliquer cinq opérations sur ces ressources : GET, PUT, POST, PATCH, DELETE. Les requêtes des clients au service sont atomiques, c.-à-d. il y a une réponse (ou un exception) par requête. Aucune information du client est stocké sur le serveur. Authentification🡪Token. Avantages : Une interface simple (juste 5 méthodes). La performance est la priorité. Modificabilité, fiabilité, évolutivité, portabilité augmentées. Désavantages : Le manque de normes et de spécifications formelles ajoutent des défis aux tâches périphériques pour la maintenance des systèmes et des services (découverte, recherche, composition, évolution, adaptation). Les fonctionnalités plus complexes ne sont pas toujours disponibles. Il faut une organisation des données. \***RPC** : Les services REST fonctionnent sur des ressources et juste avec des méthodes simples. Les « RPC » peuvent appeler n’importe quelle méthode (comme sur un objet). Elles utilisent des formats standards pour transférer les données (XML ou JSON). SOAP est une évolution et CORBA est une implémentation des RPC. Les avantages et désavantages sont presque les mêmes que pour REST, mais avec plusieurs capacités au niveau des méthodes disponibles, et sans imposer une organisation des données. \***SOAP** : Protocole de messages. Il est indépendant du langage, mais aussi du protocole de communication. SOAP utilise XML pour spécifier les données. SOAP inclue le message (en XML) et les détails de la communication (comme une demande http). Les services qui utilise SOAP utilisent les normes WS(WSDL). 🡪 **WSDL :** Les interfaces de services SOAP sont définies en WSDL. Langage basé sur le XML. Le fichier WSDL spécifie les opérations, leurs types de retour, et les types de leurs paramètres. Il spécifie aussi la localisation du service et comment on peut y accéder (la liaison et le protocole de communication). Avantages SOAP : Indépendant du protocole de communication. Il est basé sur XML, donc tous les outils du langage sont disponibles (chercher, comparer, etc.) Il est possible de définir plusieurs normes et spécifications à nouveau. Désavantages SOAP : n’est pas efficace. Les interfaces de service et les formats des messages sont longs et complexes. Parcourir les fichiers est couteux.

2. *Quelle est la propriété de SOA qui facilite la portabilité mais empêche la testabilité et la maintenabilité?* **Les interfaces abstraites**. Les clients et les services sont indépendants au niveau du langage, mais les détails de l’implémentation du service sont cachés, empêchant de comprendre comment le service fonctionne ou peut être modifié.

3. Une composition de services peut ressembler à quel autre style d’architecture logicielle? **Pipe-filter**. Le tuyau est le réseau.

4. Nous voulons développer un logiciel pour la bibliothèque. Le logiciel permettra l’ajout, l’élimination, la mise à jour et la lecture des informations sur les livres. En plus, on pourra chercher un livre par son titre, créer des collections de livres, ou trier une collection de livres par le nom des auteurs. Concevez une architecture composite en utilisant les styles SOA. **REST** pour les méthodes GET, PUT, POST, DELETE et un service **RPC/SOAP** pour les autres

*1.Nous avons un système pour gérer le contenu académique de Polytechnique. Nous avons des cours, des programmes, des concentrations. Les étudiants entrent dans des programmes, ils choisissent des cours et ils reçoivent leurs diplômes en fonction d'une concentration (quels cours ils ont suivis). Chaque département gère ses programmes et ses cours de manière indépendante, mais les étudiants peuvent également choisir des cours entre les départements. Lors de la création de nouveaux cours, nous devons soumettre une analyse de cours et un plan, qui devront être approuvés par le département. Lors de la sélection d'un cours, les étudiants sont inscrits dans le catalogue du cours et ils reçoivent un accès et du matériel. Lorsqu’un programme est créé, il doit inclure des cours hors catalogue du département selon un certain nombre de crédits et être approuvé par le département et l’Université. Comment décomposeriez-vous le système en microservices? Lesquels auraient leur propre base de données et lesquels devraient être partagés?* **Microservices**🡪 Création de cours -> Cours BD. Création de programme -> Programmes BD (accès à Cours BD). Sélection de cours -> Accès à la base de données des cours, accès potentiel à une base de données des étudiants, éventuellement accès indirect à la base de données des programmes. Chaque département a son propre ensemble de microservices (décomposer par domaine), l'Université dispose d'un microservice avec accès à tous les programmes BD.

2. *Vous développez une application mobile pour aider Polytechnique à suivre les services alimentaires à Polytechnique et à l'UdeM. L'application contient des informations sur ce qui est disponible, quand elle est ouverte, où elle se trouve, ce qui est sur le menu et autres. Les étudiants peuvent se connecter, rechercher des restaurants / cafés / comptoirs de collations, vérifier leurs informations, laisser un avis ou un commentaire et fournir des commentaires pour l'application. Vous collectez vous-même les données de chaque station alimentaire du campus. Comment concevriez-vous cette application?* **Monolithe**. Petite équipe (une personne, peut-être deux), une fonctionnalité claire, des modifications via l'application (ajouter / mettre à jour / supprimer une station alimentaire).

3. Votre application de restauration est un énorme succès! Les stations alimentaires du campus vous contactent pour vous envoyer leurs nouveaux éléments de menu, pour publier des offres spéciales pour les étudiants, pour obtenir des commentaires sur les éléments de menu saisonniers. Vos amis pensent que ce serait une bonne idée de lier l'application aux médias sociaux et d'autoriser la publication croisée et le chat entre les utilisateurs de l'application. De plus, en raison du COVID-19, les étudiants ne peuvent pas être présents en permanence sur le campus et les services de restauration ont des difficultés à planifier leur production pour les personnes qui se trouvent sur le campus. Ils vous demandent de mettre en place une fonction de précommande pour les personnes qui seront sur le campus un jour donné. Vous entrez en contact avec les 96 de vos camarades de classe LOG8430 pour étendre votre application. Quel type d'architecture leur présenteriez-vous? Quels composants mettriez-vous en œuvre? Comment l'équipe de développement sera-t-elle organisée? **Architecture microservice**. Un service par capacité business: Service des stations alimentaires (ajouter / mettre à jour / supprimer) -> DB des stations alimentaires. Service de médias sociaux (se connecter à une plateforme, publier des commentaires / questions) -> Commentaires DB. Service de chat (connexion, chat) -> Utilisateurs DB. Service de précommande -> Commandes DB, accès potentiel à la base de données de la station alimentaire et à la base de données des utilisateurs. Équipes interfonctionnelles par service (backend, UI, testeurs, QA, intégrateurs système externes) pas plus de 12 personnes par équipe.

*1.Nous sommes dans un restaurant de fast food. Les clients passent leur commande à la caisse. Si la commande ne peut pas être servie immédiatement, les clients se retirent et attendent. Lorsque la commande est prête, ils reculent devant la file pour recevoir leur repas de la caisse*. **Monitor Object**. Le caissier est l'objet moniteur. Les clients sont les requêtes synchronisées. Un seul client peut accéder à la caisse à un moment donné. Les clients « attendent » pendant que le caissier prépare une commande ou sert d'autres clients. Le caissier informe le client suivant lorsqu'il est prêt. Le caissier est l'objet moniteur. Les clients sont les requêtes synchronisées. Un seul client peut accéder à la caisse à un moment donné. Les clients « attendent » pendant que le caissier prépare une commande ou sert d'autres clients. Le caissier informe le client suivant lorsqu'il est prêt.

2. *Chrome utilise des plugins pour permettre la présentation de certains types de médias, comme Flash. Lorsqu'une page contenant de tels médias doit être chargée, le navigateur vérifie si le plug-in approprié est installé. Si tel est le cas, le navigateur appelle automatiquement le plugin pour gérer le média.* **Interceptor**. Le navigateur est le framework dont les fonctionnalités peuvent être étendues. Les plugins sont les intercepteurs. Le répartiteur identifie dynamiquement quel est le meilleur plugin à ajouter. Objectif : Permet à des services d’être ajoutés à un cadriciel de manière transparente et d’être déclenchés automatiquement lorsque certains évènements se produisent. Application : Lorsqu’un cadriciel doit être capable d’enregistrer et de déclencher de nouveaux services qui n’était pas planifiés originalement. Aussi, pour permettre à des applications de contrôler le comportement et la fonctionnalité du cadriciel.

3.*Un FedEx Airbill contient une section intitulée: "Vos informations de référence de facturation interne (facultatif: les 24 premiers caractères apparaîtront sur la facture)." L'expéditeur d'un colis utilise ce champ comme confirmation. Cette confirmation est renvoyée par FedEx à l'expéditeur avec la facture qui informe l'expéditeur que la transaction est terminée. FedEx définit délibérément ce champ de manière très vague: il s'agit d'un maximum de 24 caractères, qui sont autrement « non typés ». Par conséquent, les expéditeurs peuvent utiliser le champ de différentes manières. Par exemple, un expéditeur peut remplir ce champ avec l'index d'un enregistrement pour une base de données interne ou avec le nom d'un fichier contenant une « liste de tâches » à effectuer après la réception de l'accusé de réception du colis FedEx.* **Asynchronous Completion Token (ACT)**. Les « informations de référence de facturation interne » sont l'ACT. Ceci est ignoré par FedEx mais il est renvoyé à l'expéditeur afin qu'il puisse gérer la réponse. Par exemple, il peut s'agir d'une catégorie de produits afin que toutes les informations de suivi de cette catégorie soient regroupées.

*4.Vous avez probablement un téléphone mobile associé à un numéro de téléphone spécifique. Les gens peuvent appeler ce numéro et vous joindre. Lorsque quelqu'un le fait, votre téléphone sonne et vous pouvez choisir de répondre et d'engager la conversation avec la personne de l'autre côté de la ligne*. **Reactor** or **acceptor/connector**. Le fournisseur de télécommunications est le Reactor. Votre numéro de phone est le Handle. Vous (avec votre téléphone portable) êtes l’EventHandler. **Acceptor/Connector** peut être une autre réponse, mais généralement l'acceptation d'un appel vient après la partie connexion, donc Reactor est une meilleure réponse. Dans le cas d’un centrer d’appel, Acceptor/Connector est la meilleure solution.

5. *Les voyageurs arrivent à l'aéroport et certains d'entre eux vont prendre un taxi. Les taxis font la queue et les passagers aussi, attendent d'être servis un à la fois. Normalement, le premier passager sera servi par le premier taxi, mais dans certains cas, un chauffeur de taxi peut choisir de servir une destination particulière.* **Leader/Followers**. La ligne de taxis est le pool de fils. Les taxis sont les gestionnaires de l'événement. Les passagers sont les sources de l'événement. Objectif : Patron architectural qui fournit un modèle efficace de concurrence où plusieurs fils d’exécution partagent à tour de rôle un ensemble de sources d’évènements pour détecter, démultiplexer, soumettre et exécuter des requêtes de service qui arrivent d’une source d’évènements. Application : Dans une application basée sur la programmation événementielle où un grand nombre de requêtes de service arrivent sur un ensemble de sources d’évènements, qui doivent être traitées efficacement par plusieurs fils d’exécution qui partagent les sources d’évènements. Problème : L’utilisation de plusieurs fils d’exécution est une technique courante pour implémenter des applications qui doivent traiter de nombreux évènements de façon concurrente. Il est cependant difficile d’implémenter des serveurs haute-performance ayant plusieurs fils d’exécution. Ces applications doivent souvent traiter des grands nombres d’évènements de différents types qui arrivent tous simultanément. Solution : Structurer une queue de fils d’exécution pour partager de façon efficace un ensemble de sources d’évènements et démultiplexer à tour de rôle les évènements qui arrivent de ces sources, puis envoyer de façon synchrone les évènements aux services applicatifs pour qu’ils soient traités