Polytech'Nice Sophia SI5 / M2 IFI - 2017/2018 G. Molines
UNS Février 2018

Examen d'Architecture logicielle

Aucun document autorisé

Durée : 3 heures, de 8h à 11h le 23 février 2018.

Lisez tout le sujet avant de commencer à répondre aux questions. Les parties 1 et 2 sont indépendantes.
Remarque : Toute ambiguïté que vous pourriez rencontrer dans cet examen devra être résolue en décrivant brièvement le choix que vous avez fait.
Vous devez répondre aux questions posées dans le sujet sur la copie d'examen fournie, avec d'éventuelles feuilles intercalaires.
Vous pouvez répondre en français ou en anglais, avec pour condition que l'intégralité de la copie soit rédigée dans une seule de ces deux langues.
Vous ne pouvez pas sortir de la salle dans la première moitié de l'épreuve, soit avant 9h30, ni dans le dernier quart d'heure, soit après 10h45.
Toute fraude identifiée sera systématiquement transmise au conseil de discipline de l'UNS.

1. Etude de cas : Myrina (14 points)

Jeffrey Bozes vient de brillamment décrocher son diplôme d'ingénieur à Polytech et décide de lancer une libraire en ligne, Myrina, couvrant tout le département des Alpes-Maritimes. Sa particularité sera de livrer des livres par drones, la faim de lecture ne souffrant pas d'attendre les aléas des Postes.
En terme d'infrastructure, il établit un entrepôt près du port de Nice pour recevoir les livres étrangers par containers. Il se dote également d'un petit camion qui fera des allers-retours entre la gare de Nice et l'entrepôt afin d'y apporter les livres des éditeurs parisiens livrés en train. Enfin, il se dote d'une flotte d'une dizaine de drône pouvant transporter chacun un colis de 1 kg.
Il souhaite offrir un site web permettant de parcourir le catalogue de livres, de les commander et de les payer. Une fois le paiement validé, le site web doit indiquer au client l'heure de livraison prévue en fonction des stocks et des drones disponibles.
Il a donc besoin d'un logiciel lui permettant de mettre à jour ce site web, mais aussi de planifier les tournées des drones, c'est à dire leur plan de vol, la gestion de leur autonomie énergétique. On fait l'hypothèse qu'un drone livre un seul client à la fois, puis revient à l'entrepôt pour la livraison suivante. D'autre part, si un client commande pour plus d'un kg de livres, on allouera plusieurs drones qui volent en formation pour arriver en même temps chez le client, afin de ne pas l'obliger à attendre plusieurs passages. Dans le cas d'un vol en formation, un drone est désigné maître et détient le plan de vol, les autres sont esclaves et doivent le suivre.
Il a également besoin d'une interface sur une tablette dans le petit camion qui fait les aller-retours entre la gare et l'entrepôt, et qui permette au chauffeur de savoir quel colis aller récupérer sur quel train, et de mettre à jour les stocks de l'entrepôt en arrivant.

On fait l'hypothèse (audacieuse) que la couverture réseau est bonne dans Nice (où opère le petit camion), mais devient assez aléatoire dès qu'on s'éloigne du centre et monte dans les collines, où vont les drones. On décide aussi que le port de Nice offre une API donnant les arrivées des bateaux, et qu'on a par ailleurs une liste des colis à bord de chaque bateau.

A/ (4 points)
Identifiez les différents acteurs et composants du système informatique dont doit se doter Jeffrey Bozes pour répondre à son besoin. Pour chaque composant, en décrire la fonction et caractériser son cahier des charge notamment non-fonctionnel.
Maximum deux pages (hors schémas éventuels)

B/ (2 points)
Quelles problématiques d'architecture voyez-vous émerger?
Maximum une page

C/ (4 points)
Proposez une architecture logicielle répondant au besoin, en décrivant les technologies utilisées, les types de communication, et surtout en justifiant les choix que vous faites. Vous relierez votre proposition aux problématiques mentionnées en **B**.
Maximum deux pages (hors schémas éventuels)

D/ (4 points)
Jeffrey se rend compte que les drones n'ont pas assez d'autonomie pour atteindre les coins les plus reculés du département. Il décide donc de se doter de quelques camions pour les atteindre. Sa stratégie est de charger dans le camion les livres pour une dizaine de clients proches, ainsi que deux ou trois drones, et de s'approcher de la zone intéressante. Le chauffeur du camion accroche des livres à un drone, l'envoie, puis roule un peu plus, envoie le drone suivant tout en récupérant le premier (qui sera donc revenu au nouveau point du camion et non pas à son lieu de départ), et continue sa boucle.
Il va donc falloir équiper ce chauffeur d'un logiciel intelligent lui permettant de programmer des plans de vols de drones à la volée, et d'indiquer au chauffeur les colis à composer et la tournée à faire. Bien sur, comme on part dans un coin reculé, on n'a pas de réseau. Le logiciel doit se synchroniser en partant de l'entrepôt de Nice, puis être autonome pour toute la tournée.

Comment structureriez-vous un tel logiciel ?
Comment modifier l'architecture présentée en **C** afin d'y inclure ce logiciel ?

Maximum deux pages (hors schémas éventuels)

2. Question de recul: (6 points)

Soit une architecture basée sur une application J2EE monolithique, composée des éléments suivants:

- une base de données relationnelle, dans laquelle on stocke les objets métiers, via un mapping objet-relationnel
- un container d'EJB (compris dans un serveur d'application), dans lequel on trouve des Entity Beans représentant le modèle des objets métiers, ainsi que des Session Beans offrant des services sur ces objets métiers

- un container de servlet (compris dans le même serveur d'application), contenant une application web sous forme de pages JSPs, s'interfaçant avec les Session Beans pour interagir avec le serveur

Pour fixer les idées, mettons que le modèle objet représente des biscuits, ingrédients de cuisine, inventaire de cuisine, catalogue de recettes, clients, et que les services permettent de commander des biscuits préparés à la demande, de vérifier qu'on ait les ingrédients suffisants pour les préparer, de les payer (on ignore les problématiques de paiement bancaire et suppose que chaque client a un compte pré-alimenté pour simplifier), notifier le client quand ils sont prêts, etc.
Cette application fonctionne sans problème à l'échelle d'un magasin.
On souhaite maintenant passer à l'échelle d'un pays.
Expliquez comment faire passer cette application à l'échelle en modifiant le moins possible l'architecture existante, et en essayant de garder le plus possible la philosophie J2EE, sans tout remplacer par une série de micro-services à la mode. Incluez un schéma

Maximum deux pages.