Examen SOA1 : Services & Intégration

- Date : 16 novembre 2015
- Durée : 13h30 – 17h30 (4 heures)
- **Aucun document autorisé; barème donné à titre indicatif.**

Lisez tout le sujet (5 pages) avant de commencer à répondre aux questions;
Les questions sont identifiées en gras dans le texte du sujet.
Les différentes parties sont indépendantes.

Vous devez répondre aux questions posées dans le sujet sur la copie d'examen fournie, avec d'éventuelles feuilles intercalaires. Vous ne pouvez pas sortir de la salle d'examen durant la première heure (avant 14h30), ni durant le dernier quart d'heure (après 17h15). **Toute fraude identifiée sera systématiquement transmise au conseil de discipline de l'UNS**.

Partie #1 : QCM & Prise de recul /4 (0h30)

Définissez 4 questions de Questionnaire à Choix Multiples (QCM) qui permettent de valider la compréhension de la matière par un étudiant.

Ces questions doivent permettre de valider à titre égal *(i)* les concepts intrinsèques de l'intégration de services et *(ii)* la mise en œuvre technique avec Apache Camel et Service Mix. Chaque question du votre QCM doit proposer 4 alternatives de réponses et comporter au moins une bonne réponse. Pour chaque question, vous indiquerez la ou les bonnes réponses et ce que vous pensez valider avec cette question. Vous serez évalué sur la justesse et la pertinence de chacune des questions ainsi que sur la couverture de la matière par votre QCM.

Partie #2 : Analyse du Pokédex /6 (1h00)

Le site **http://pokeapi.co** propose une API orientée ressources pour gérer un Pokédex. Le Pokédex est l'annuaire de tous les pokémons repertoriés dans l'univers du jeu Pokemon. Il contient des informations sur les différentes familles de pokemon (egg groups), leurs pouvoirs spéciaux (abilities), leurs attaques (moves) et leurs évolutions en d'autres pokémons. Un extrait de cette API est donné en Annexe #1.

1. **Identifiez les forces et faiblesses des services** (interface et structure de données) définis par pokeapi.co ;
2. L'API ne supporte que du GET. **Proposez une modification de cette API** permettant de la rendre *read-write*, i.e., permettant la création et l'ajout de nouvelles ressources dans le système ;
3. **Définissez en pseudo-code une version RPC** d'une interface *read-write* du Pokédex ;
4. **Définissez en pseudo-code une version Document** d'une interface *read-write* du Pokédex.

Partie #3 : Gestion d'une flotte de voiture /10 (2h30)

La *Communauté d'Agglomération de Sophia-Antipolis* (CASA) dispose d'une flotte de voitures à destination de ses agents, réparties dans plusieurs parkings sur les différentes villes du territoire sophipolitain. Dans le cadre du processus de modernisation de son système d'information, la CASA met en place un système de planification de mission pour permettre aux agents de réserver des véhicules adaptés à leurs besoins.

- Les parkings, de capacité limitée, accueillent un certain nombre de véhicules, à une adresse donnée. Les véhicules peuvent être de type différent (*i.e.,* berline, prestige, camionnette), et ont un certain kilométrage ainsi qu'un niveau d'essence (parmi Diesel, SP95 et SP98) dans le réservoir.
- La CASA a noué un partenariat financier avec la société américaine E-Corp qui propose un service RPC permettant de calculer la distance entre deux adresses, ainsi qu'un service permettant de connaître les tarifs d'une station essence donnée. Les interfaces de ces services sont données en annexe #2.
- Lorsqu'un agent de la CASA part en mission, il crée une demande de mission indiquant son lieu de départ et d'arrivée, et les raisons motivant le besoin d'un véhicule. Le système lui affecte alors un véhicule correspondant à ses besoins en émettant un bordereau d'utilisation de véhicule.
- A son retour de mission, l'agent remet le bordereau au gardien du parking d'arrivée, et le système identifie la distance parcourue, vérifie qu'elle correspond à la mission qui était confiée à l'agent, et envoie une facture à son service d'affectation, correspondant au coût d'utilisation du véhicule.
- Si le système identifie lors de l'émission du bordereau que le véhicule attribué passe près d'une station essence ayant un tarif intéressant, il est demandé à l'agent d'y faire le plein lors de son passage a proximité (champ « note » du bordereau).

1. **Définissez en pseudo-code les interfaces des services** dont vous avez besoin pour répondre à cette étude de cas. Justifiez le ou les paradigmes choisis par rapport à l'étude de cas ; *(3 points)*
2. Parmi ces services, **quels services sont implémentés par des flots d'intégration** ? Définissez en pseudo-code Camel ou via la langage graphique des EIP l'implémentation de ces flots d'intégration. *(7 points)*

Deleted: s

Deleted: e

Deleted: u

Appendix #1 : Pokédex – pokeapi.co

Documentation extraite du site http://pokeapi.co/docs/

This is a **consumption-only** API - only the HTTP GET method is available on resources.

Pokédex

A Pokedex returns the names and resource_uri for all pokemon. This is a special resource call. Always call **/api/v1/pokedex/1/**.

```
GET http://pokeapi.co/api/v1/pokedex/1/
```

Result:

```
{
  "created":  "2013-11-09T15:14:48.957604",
  "modified": "2013-11-09T15:14:48.957565",
  "name": "national",
  "pokemon": [
    { "name": "pidgeotto", "resource_uri": "api/v1/pokemon/17/" },
    ...
  ]
}
```

Pokémon

A Pokemon resource represents a single Pokémon. Note: the ID for Pokémon is special. Use the **National pokédex** number as the ID to return the desired resource.

```
GET http://pokeapi.co/api/v1/pokemon/{ID}/
```

Result (excerpt):

```
{
  "abilities": [
    { "name": "overgrow",    "resource_uri": "/api/v1/ability/1/" },
    { "name": "chlorophyll", "resource_uri": "/api/v1/ability/2/"}
  ],
  "attack": 49,  "defense": 49,
  "created": "2013-11-02T12:08:25.745455",
  "egg_groups": [
    { "name": "Monster", "resource_uri": "/api/v1/egg/1/" },
    { "name": "Grass",   "resource_uri": "/api/v1/egg/8/" }
  ],
  "evolutions": {
    "level": 16, "method": "level up", "resource_uri": "/api/v1/pokemon/2/", "to": "Ivysaur"
  },
  "height": "2'4", "weight": "15.2lbs", "hp": 45,
  "modified": "2013-11-02T13:28:04.914889",
  "name": "Bulbasaur", "national_id": 1,
  "resource_uri": "/api/v1/pokemon/1/"
  "moves": [
    { "learn_type": "other", "name": "Tackle", "resource_uri": "/api/v1/move/1/" }, ...
  ],
}
```

Ability

An Ability resource represents a single Pokémon ability.

```
GET http://pokeapi.co/api/v1/ability/{ID}/
```

Result:

```
{
  "created": "2013-11-02T12:21:28.166682",
  "description": "When a Pokémon with Overgrow uses a Grass-type move, the power will increase
                  by 1.5x if the user has less than or equal to 1/3 of its maximum HP
                  remaining.",
  "id": 1,
  "modified": "2013-11-02T13:27:06.591413",
  "name": "Overgrow",
  "resource_uri": "/api/v1/ability/1/"
}
```

Egg group

An Egg group resource represents a single Pokémon egg group.

```
GET http://pokeapi.co/api/v1/egg/{ID}/
```

Result:

```
{
  "created": "2013-11-02T12:24:55.532023",
  "id": 1,
  "modified": "2013-11-02T12:24:55.531989",
  "name": "Monster",
  "pokemon": [
    { "name": "Ivysaur",   "resource_uri": "/api/v1/pokemon/2/" },
    { "name": "Bulbasaur", "resource_uri": "/api/v1/pokemon/1/" },
    ...
  ],
  "resource_uri": "/api/v1/egg/1/"
}
```

Move

A Move resource represents a single move.

```
GET http://pokeapi.co/api/v1/move/{ID}/
```

Result:

```
{
  "accuracy": 100, "name": "Tackle",   "power": 50,   "pp": 35,
  "category": "physical",
  "created": "2013-11-02T12:37:05.266762",
  "description": "A physical attack in which the user charges and slams into the target with
                  its whole body.",
  "id": 1,
  "modified": "2013-11-03T11:51:57.009696",
  "resource_uri": "/api/v1/move/1/"
}
```

Appendix #2 : Services from E-corp

Itinerary service definition

The service supports the computation of the minimal distance between two given adresses. It can compute the minimal distance, or identify a list of distances corresponding to alternatives routes (e.g., highway, local roads). It also provides an operation to build the shortest itinerary between two addresses. An itineray contains the list of the crossed cities (e.g., from the Allianz Riviera stadium to the school, the list will contain Nice, Saint Laurent du Var, Cagnes sur Mer, Villeneuve-Loubet and Biot).

```
Service Itinerary {

  // Returns the minimal distance (in miles) between two addresses
  Operation distanceByRoad(from: Address, to: Address): Numeric

  // Returns the "classical" distances for alternative routes between two addresses
  Operation altDistancesByRoad(from: Address, to: Address): List<Numeric>

  // Build the shortest itinerary, identifying the crossed cities as a list of address.
  Operation buildItinerary(from: Address, to: Address) List<Address>

}

Structure Address {
  Line1: String
  Line2: String
  ZipCode: Numeric
  Country: String
}
```

Gas station service definition

The service lists all the gas stations around a given address. Fot a given station, it can returns the price of a given fuel type.

```
Service Gas_Station {

  // Returns the price per galon for a given fuel type at a given station
  Operation getByStationID(id: Int, kind: Fuel): Numeric

  // Find all the station around an address within a given radius (in miles)
  Operation getStationByLocation(address: String, radius: numeric): List<Int>

}

Enumeration Fuel { DIESEL, REGULAR, SPECIAL }
```