Architecture Micro-services pour Pile ou Face

Vous avez pour le moment deux modules modèles. Les façades n'ont pas été intégrées dans les modules pour des raisons techniques (@Component). Vous intégrerez vos façades FacadePartie(Impl) et FacadeJoueur(Impl) dans les packages pileouface.modele et authent.modele de vos web-services respectifs. Vous trouverez classes et interfaces à la fin de ce sujet.

Développement d'un service Rest authent

Le but est simple. Nous voulons que le script suivante fonctionne.

```
POST localhost:8081/authent/inscription?pseudo=yoh&mdp=yoh

> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 200, "Le compte n'a pas été créé");
});
%}
###



POST localhost:8081/authent/inscription?pseudo=yoh&mdp=fred

> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 409, "Il y aurait du y avoir un conflit");
});
%}
###



POST localhost:8081/authent/token?pseudo=yoh&mdp=yoh


> {%
client.global.set("auth_token", response.headers.valueOf("token"));
client.test("Request executed successfully", function() {
  client.assert(response.status === 200, "Le token aurait dû être créé");
});
%}
###
```

```
POST localhost:8081/authent/token?pseudo=yoh&mdp=fred


> {%

client.test("Request executed successfully", function() {
  client.assert(response.status === 401, "Les identifiants ne sont pas corrects et
auraient dû provoquer un 401");
});
%}
###


POST localhost:8081/authent/token?pseudo=fred&mdp=fred


> {%

client.test("Request executed successfully", function() {
  client.assert(response.status === 404, "L'utilisateur n'existe pas et cela aurait dû
provoquer un 404");
});
%}
###







GET localhost:8081/authent/token?token={{auth_token}}


> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 200, "Le token aurait dû être valide");
});
%}

###

GET localhost:8081/authent/token?token=rezarzerar


> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 404, "Le token aurait dû être valide");
});
%}
```

```
###


DELETE localhost:8081/authent/inscription/yoh
Content-Type: application/x-www-form-urlencoded

mdp=fred

> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 401, "Code 401 attendu car mot de passe
incorrect");
});
%}

###



DELETE localhost:8081/authent/inscription/fred
Content-Type: application/x-www-form-urlencoded

mdp=fred

> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 404, "Code 404 attendu car utilisateur inconnu");
});
%}

###

DELETE localhost:8081/authent/inscription/yoh
Content-Type: application/x-www-form-urlencoded

mdp=yoh

> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 200, "Le compte aurait dû être supprimé");
});
%}
```

Nous voulons que :

- Le service soit déployé sur localhost:8081
- que les uris suivantes soient gérées :
    - /authent/inscription [POST] (param pseudo, mdp) : permet d'inscrire un utilisateur au

service

- /authent/inscription/{pseudo} [DELETE] (param mdp) : permet de supprimer un utilisateur si le mdp est ok. De plus, il faudra maintenir l'autre web-service à jour en effaçant les données relatives à ce joueur.
- /authent/token [POST] (param pseudo, mdp) : permet de créer un token pour un utilisateur inscrit ce qui lui permettra d'accèder au service de jeu
- /authent/token [GET] (param token) : retour du pseudo dans le body pour le token donné

Développement d'un service Rest jeu

Le jeu de script suivant devra valider le bon fonctionnement de votre web-service

```
POST localhost:8081/authent/inscription?pseudo=yoh&mdp=yoh

> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 200, "Le compte n'a pas été créé");
});
%}
###



POST localhost:8081/authent/inscription?pseudo=yoh&mdp=fred

> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 409, "Il y aurait du y avoir un conflit");
});
%}
###



POST localhost:8081/authent/token?pseudo=yoh&mdp=yoh


> {%
client.global.set("auth_token", response.headers.valueOf("token"));
client.test("Request executed successfully", function() {
  client.assert(response.status === 200, "Le token aurait dû être créé");
});
%}
###


POST localhost:8081/authent/token?pseudo=yoh&mdp=fred
```

```
> {%

client.test("Request executed successfully", function() {
  client.assert(response.status === 401, "Les identifiants ne sont pas corrects et
auraient dû provoquer un 401");
});
%}
###


POST localhost:8081/authent/token?pseudo=fred&mdp=fred


> {%

client.test("Request executed successfully", function() {
  client.assert(response.status === 404, "L'utilisateur n'existe pas et cela aurait dû
provoquer un 404");
});
%}
###







GET localhost:8081/authent/token?token={{auth_token}}


> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 200, "Le token aurait dû être valide");
});
%}

###

GET localhost:8081/authent/token?token=rezarzerar


> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 404, "Le token aurait dû être valide");
});
%}


###
```

```
POST localhost:8082/jeu/partie
token: {{auth_token}}
Content-Type: application/x-www-form-urlencoded

prediction=Pile

> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 200, "La partie aurait dû être possible");
});
%}

###


POST localhost:8082/jeu/partie
token: bababr
Content-Type: application/x-www-form-urlencoded

prediction=Pile

> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 404, "On aurait dû avoir un 404 car le token
n'existe pas");
});
%}

###


POST localhost:8082/jeu/partie
token: {{auth_token}}
Content-Type: application/x-www-form-urlencoded

prediction=Pile

###

POST localhost:8082/jeu/partie
token: {{auth_token}}
Content-Type: application/x-www-form-urlencoded

prediction=Pile

###

POST localhost:8082/jeu/partie
token: {{auth_token}}
Content-Type: application/x-www-form-urlencoded
```

```
prediction=Pile

###




POST localhost:8082/jeu/partie
token: {{auth_token}}
Content-Type: application/x-www-form-urlencoded

prediction=Pile

###

POST localhost:8082/jeu/partie
token: {{auth_token}}
Content-Type: application/x-www-form-urlencoded

prediction=Pile

###

GET localhost:8082/jeu/partie
token: {{auth_token}}

> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 200, "On aurait dû pouvoir récupérer les
parties");
});
%}

###

GET localhost:8082/jeu/statistiques
token: {{auth_token}}

> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 200, "On aurait dû récupérer les statistiques");
});
%}
###



GET localhost:8082/jeu/statistiques
token: babar
```

```
> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 404, "On aurait dû avoir un 404 car le token
n'existe pas");
});
%}
###

DELETE localhost:8081/authent/inscription/yoh
Content-Type: application/x-www-form-urlencoded

mdp=fred

> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 401, "Code 401 attendu car mot de passe
incorrect");
});
%}

###



DELETE localhost:8081/authent/inscription/fred
Content-Type: application/x-www-form-urlencoded

mdp=fred

> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 404, "Code 404 attendu car utilisateur inconnu");
});
%}

###

DELETE localhost:8081/authent/inscription/yoh
Content-Type: application/x-www-form-urlencoded

mdp=yoh

> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 200, "Le compte aurait dû être supprimé");
});
%}
```

Nous voulons que :

- Le service soit déployé sur localhost:8082
- que les uris suivantes soient gérées :
  - /jeu/partie [POST] (param prediction, header token) : le joueur ayant le token fait une partie de pile ou face. Il est nécessaire d'aller interroger l'autre service pour connaître le pseudo du joueur
  - /jeu/partie [GET] (header token) : Permet de récupérer toutes les parties du joueur caché derrière le token
  - /jeu/statistiques [GET] (header token) : Permet de récupérer les statistiques du joueur caché derrière le token
  - /jeu/joueur/{pseudo} [DELETE] : Permet de supprimer le joueur de la plate-forme pile ou face.

Introduction d'une Gateway

En vous inspirant de https://cloud.spring.io/spring-cloud-gateway/reference/html/#gatewayfilter-factories, nous voulons mettre en place une gateway qui fonctionnera sur localhost:8080.

Vous pouvez aussi vous inspirer de https://www.youtube.com/watch?v=71ZH1FFECRQ&list=PLbIvwFkPyMFs40q_Vl5vAEJ2pREDs0b_A&index=4 (en n'intégrant pas Consul).

- Nous voulons faire matcher toutes les uri disponibles sur authent sur des uri de la forme : /api/auth/*
- Nous voulons faire matcher toutes les uri disponibles sur jeu sur des uri de la forme : /api/pileouface/*

A vous de jouer.

Normalement le script ci-dessous devrait confirmer le bon fonctionnement de votre archiutecture.

```
POST localhost:8080/api/auth/inscription?pseudo=yoh&mdp=yoh

> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 200, "Le compte n'a pas été créé");
});
%}
###



POST localhost:8080/api/auth/inscription?pseudo=yoh&mdp=fred

> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 409, "Il y aurait du y avoir un conflit");
});
```

```
%}
###



POST localhost:8080/api/auth/token?pseudo=yoh&mdp=yoh


> {%
client.global.set("auth_token", response.headers.valueOf("token"));
client.test("Request executed successfully", function() {
  client.assert(response.status === 200, "Le token aurait dû être créé");
});
%}
###


POST localhost:8080/api/auth/token?pseudo=yoh&mdp=fred


> {%

client.test("Request executed successfully", function() {
  client.assert(response.status === 401, "Les identifiants ne sont pas corrects et
auraient dû provoquer un 401");
});
%}
###


POST localhost:8080/api/auth/token?pseudo=fred&mdp=fred


> {%

client.test("Request executed successfully", function() {
  client.assert(response.status === 404, "L'utilisateur n'existe pas et cela aurait dû
provoquer un 404");
});
%}
###









GET localhost:8080/api/auth/token?token={{auth_token}}


> {%
```

```
client.test("Request executed successfully", function() {
  client.assert(response.status === 200, "Le token aurait dû être valide");
});
%}

###

GET localhost:8080/api/auth/token?token=rezarzerar


> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 404, "Le token aurait dû être valide");
});
%}


###


POST localhost:8080/api/pileouface/partie
token: {{auth_token}}
Content-Type: application/x-www-form-urlencoded

prediction=Pile

> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 200, "La partie aurait dû être possible");
});
%}

###


POST localhost:8080/api/pileouface/partie
token: bababr
Content-Type: application/x-www-form-urlencoded

prediction=Pile

> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 404, "On aurait dû avoir un 404 car le token
n'existe pas");
});
%}

###
```

```
POST localhost:8080/api/pileouface/partie
token: {{auth_token}}
Content-Type: application/x-www-form-urlencoded

prediction=Pile

###

POST localhost:8080/api/pileouface/partie
token: {{auth_token}}
Content-Type: application/x-www-form-urlencoded

prediction=Pile

###

POST localhost:8080/api/pileouface/partie
token: {{auth_token}}
Content-Type: application/x-www-form-urlencoded

prediction=Pile

###




POST localhost:8080/api/pileouface/partie
token: {{auth_token}}
Content-Type: application/x-www-form-urlencoded

prediction=Pile

###

POST localhost:8080/api/pileouface/partie
token: {{auth_token}}
Content-Type: application/x-www-form-urlencoded

prediction=Pile

###

GET localhost:8080/api/pileouface/partie
token: {{auth_token}}

> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 200, "On aurait dû pouvoir récupérer les
parties");
```

```
});
%}

###

GET localhost:8080/api/pileouface/statistiques
token: {{auth_token}}

> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 200, "On aurait dû récupérer les statistiques");
});
%}
###



GET localhost:8080/api/pileouface/statistiques
token: babar

> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 404, "On aurait dû avoir un 404 car le token
n'existe pas");
});
%}
###

DELETE http://localhost:8080/api/auth/inscription/yoh
Content-Type: application/x-www-form-urlencoded

mdp=fred

> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 401, "Code 401 attendu car mot de passe
incorrect");
});
%}

###



DELETE http://localhost:8080/api/auth/inscription/fred
Content-Type: application/x-www-form-urlencoded

mdp=fred

> {%
client.test("Request executed successfully", function() {
```

```
  client.assert(response.status === 404, "Code 404 attendu car utilisateur inconnu");
});
%}

###

DELETE http://localhost:8080/api/auth/inscription/yoh
Content-Type: application/x-www-form-urlencoded

mdp=yoh

> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 200, "Le compte aurait dû être supprimé");
});
%}----
```

```
[source, java]
```

package authent.modele;

public interface FacadeJoueur { /** * Inscription d'un nouveau joueur à la POFOL * * @param nouveauJoueur * @param mdp * @throws PseudoDejaPrisException */ void inscription(String nouveauJoueur, String mdp) throws PseudoDejaPrisException;

```
/**
 * Connexion à POFOL
 *
 * @param pseudo
 * @param mdp
 * @return
 * @throws JoueurInexistantException
 */
String genererToken(String pseudo, String mdp) throws
        JoueurInexistantException, OperationNonAutorisee;
```

```
/**
 * Permet de se désinscrire de la plate-forme
 * @param pseudo
 * @param mdp
 * @throws JoueurInexistantException
 */
void desincription(String pseudo, String mdp) throws JoueurInexistantException,
OperationNonAutorisee;
```

```
    /**
     * Permet de récupérer le pseudo du joueur possédant ce token
     * @param token
     * @return le pseudo correspondant au token
     * @throws MauvaisTokenException
     */
    String checkToken(String token) throws MauvaisTokenException;
}
```

```
[source, java]
```

package authent.modele;

import org.springframework.stereotype.Component;

import java.util.HashMap; import java.util.Map; import java.util.UUID;

@Component("facadeJoueurs") public class FacadeJoueurImpl implements FacadeJoueur {

```
/**
 * Dictionnaire des joueurs inscrits
 */
private Map<String,Joueur> joueurs;
```

```
/**
 * Dictionnaire des joueurs connectés indexés par une clé aléatoire
 */
private Map<String,Joueur> joueursConnectes;
```

```
public FacadeJoueurImpl() {
    this.joueurs = new HashMap<>();
    this.joueursConnectes = new HashMap<>();
}
```

```
@Override
public void inscription(String nouveauJoueur, String mdp) throws
PseudoDejaPrisException {
```

```
if (joueurs.containsKey(nouveauJoueur))
    throw new PseudoDejaPrisException();
```

```
    this.joueurs.put(nouveauJoueur,new Joueur(nouveauJoueur,mdp));
}
```

```
private void checkIdConnexion(String idConnexion) throws MauvaisTokenException {
    if (!this.joueursConnectes.containsKey(idConnexion))
        throw new MauvaisTokenException();
}
```

```
/**
 * Génération d'un token pour jouer
 * @param nouveauJoueur
 * @param mdp
 * @return
 * @throws JoueurInexistantException
 */
@Override
public String genererToken(String nouveauJoueur, String mdp) throws
        JoueurInexistantException, OperationNonAutorisee {
```

```
if (!joueurs.containsKey(nouveauJoueur))
    throw new JoueurInexistantException();
```

```
    Joueur j = joueurs.get(nouveauJoueur);
    if (j.checkPassword(mdp)) {
        String idConnection = UUID.randomUUID().toString();
        this.joueursConnectes.put(idConnection, j);
        return idConnection;
    }
    else {
        throw new OperationNonAutorisee();
    }
}
```

```
@Override
public void desincription(String pseudo, String mdp) throws JoueurInexistantException,
OperationNonAutorisee {
    if (!joueurs.containsKey(pseudo))
        throw new JoueurInexistantException();
```

```
Joueur j = joueurs.get(pseudo);
if (j.checkPassword(mdp)) {
    this.joueurs.remove(pseudo);
}
else {
    throw new OperationNonAutorisee();
}
```

```
}
```

```
@Override
public String checkToken(String token) throws MauvaisTokenException {
    if (joueursConnectes.containsKey(token)){
        return joueursConnectes.get(token).getNomJoueur();
    }
    else {
        throw new MauvaisTokenException();
    }
}
```

}

```
[source, java]
```

package pileouface.modele;

import java.util.Collection;

public interface FacadeParties {

```
/**
 * Permet de jouer une partie
 *
 * @param idConnexio
 * @param choix
 * @return le résultat de la partie
 * @throws MauvaisIdentifiantConnexionException
 */
Partie jouer(String idConnexio, String choix) throws
MauvaisIdentifiantConnexionException;
```

```
/**
 * Permet de récupérer les statistiques d'un joueur
 * @param idConnexion
 * @return
 * @throws MauvaisIdentifiantConnexionException
 */
```

```
Statistiques getStatistiques(String idConnexion) throws
MauvaisIdentifiantConnexionException;
```

```
/**
 * Permet de récupérer l'historique des parties d'un joueur connecté
 *
 * @param idConnexion
 * @return
 * @throws MauvaisIdentifiantConnexionException
 */
```

```
Collection<Partie> getAllParties(String idConnexion) throws
MauvaisIdentifiantConnexionException;
```

```
/**
 * Permet de récupérer un joueur par son pseudo s'il existe.
 * S'il n'existe pas, un nouveau joueur est créé
 * @param pseudo
 * @return
 */
Joueur getJoueur(String pseudo);
```

```
    /**
     * Permet de supprimer un joueur du SI
     * @param pseudo
     */
    void suppressionJoueur(String pseudo);
}
```

```
[source, java]
```

package pileouface.modele;

import org.springframework.stereotype.Component;

import java.util.Collection; import java.util.HashMap; import java.util.Map;

@Component("facadeParties") public class FacadePartiesImpl implements FacadeParties {

```
/**
 * Dictionnaire des joueurs connectés indexés par leur pseudo
 */
private Map<String,Joueur> joueursActuels;
```

```
public FacadePartiesImpl() {
    this.joueursActuels = new HashMap<>();
}
```

```
private void checkIdConnexion(String idConnexion) throws
MauvaisIdentifiantConnexionException {
    if (!this.joueursActuels.containsKey(idConnexion))
        throw new MauvaisIdentifiantConnexionException();
}
```

```
@Override
public Partie jouer(String idConnexio, String choix) throws
MauvaisIdentifiantConnexionException {
    this.checkIdConnexion(idConnexio);
    Joueur j = this.joueursActuels.get(idConnexio);
    Partie partie = j.jouer(choix);
    return partie;
}
```

```
/**
 * Permet de récupérer les statistiques d'un utilisateur connecté
 * @param idConnexion
 * @return
 * @throws MauvaisIdentifiantConnexionException
 */
@Override
public Statistiques getStatistiques(String idConnexion) throws
MauvaisIdentifiantConnexionException {
    this.checkIdConnexion(idConnexion);
    Joueur j = this.joueursActuels.get(idConnexion);
    int nb = j.getNbPartiesJouees();
    double ratio =
(double)this.getJoueur(idConnexion).getNbPartiesGagnees()/((double)nb);
    return new Statistiques(nb,ratio);
}
```

```
/**
 * Permet de récupérer l'historique des parties d'un joueur connecté
 * @param idConnexion
 * @return
 * @throws MauvaisIdentifiantConnexionException
 */
```

```
@Override
public Collection<Partie> getAllParties(String idConnexion) throws
MauvaisIdentifiantConnexionException {
    this.checkIdConnexion(idConnexion);
    return this.joueursActuels.get(idConnexion).getHistorique();
}
```

```
@Override
public Joueur getJoueur(String pseudo) {
```

```
    if (joueursActuels.containsKey(pseudo)){
        return joueursActuels.get(pseudo);
    }
    Joueur j = new Joueur(pseudo);
    joueursActuels.put(pseudo,j);
    return j;
}
```

```
@Override
public void suppressionJoueur(String pseudo) {
    this.joueursActuels.remove(pseudo);
}
```

}

```
```