INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Compte rendu
POSIT'IF

TP DASI n°1

GOUX Paul
TURCAN-JOUVE Clément

RENDU LE 26/03/2019
A Villeurbanne

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

I. Application POSIT'IF

A. Description de l'application

POSIT'IF est un service permettant de contacter différents voyants. Toute personne adhérant à ce service peut demander une voyance. Ces dernières sont réalisées par des employés de l'entreprise. Le client sélectionne un voyant de son choix parmis ceux proposés et demande à réaliser la voyance. L'employé ayant réalisé le moins de voyance et pouvant interpréter ce voyant est sélectionné. Une fois que l'employé accepte, un message est envoyé au client pour que celui-ci appelle. Une fois la voyance réalisée, l'employé peut mettre un commentaire, et la voyance est enregistrée dans l'historique du client.

B. Description des cas d'usage

CLIENT :

Le client a souscrit au service POSIT'IF qui lui permet à partir d'une application web d'accéder à un service de voyance. Ce dernier est désigné par ses informations personnelles ainsi que, entre autres, une adresse mail unique (il ne peut pas y avoir deux clients ayant la même adresse mail).

Le client doit donc pouvoir se connecter ou s'inscrire sur l'application lors de sa première visite. Une fois connecté ce dernier peut demander une voyance ou bien accéder à l'historique de ses appels ainsi qu'à ses informations personnelles. Dans le but de réaliser sa voyance, l'individu peut choisir son médium favori dans la liste de ceux disponible. Lorsque l'employé accepte de réaliser la voyance, le client reçoit un message lui permettant d'appeler le voyant. Une fois l'appel terminé, la voyance apparaît dans son historique comme étant terminée.

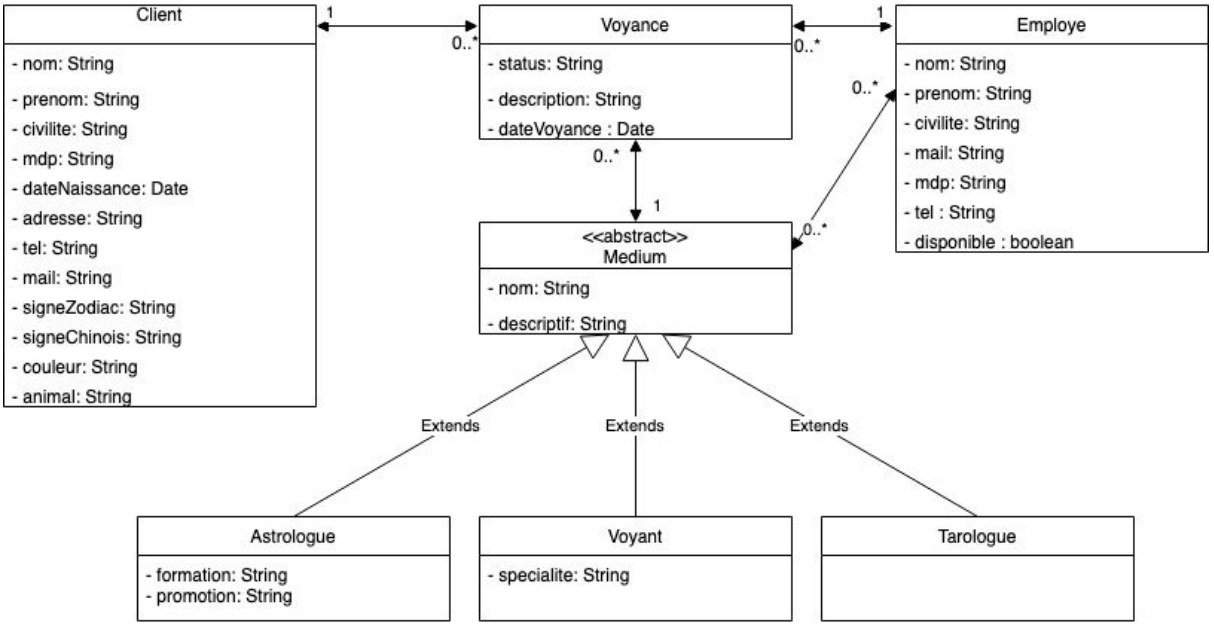
EMPLOYÉ :

Le rôle de l'employé est de réaliser les voyances demandées par les clients. Il peut interpréter plusieurs médiums différents. L'employé reçoit un message quand une voyance est demandée par un client. Il peut accéder, en se connectant à l'application, au profil du client. Quand il le souhaite il choisit d'envoyer d'un message au client concerné et attend son appel. L'employé est aidé par le service AstroTest pour ses prédictions. Pendant la voyance, le statut de l'employé passe à indisponible, il ne peut alors plus être contacté pour une voyance jusqu'à la fin de l'appel. Lorsque l'appel se termine, le préposé peut laisser un commentaire sur la voyance réalisée. Ce dernier peut aussi accéder à ses informations personnelles ainsi que des informations statistiques sur l'entreprise via le tableau de bord.

II. Services et analyse de l'application

Le cahier des charges conduit à réaliser le diagramme UML ci-dessous. Une voyance met en relation un client, un employé ainsi qu'un médium. Il est aussi indiqué qu'un

employé de POSIT'IF possède une liste de médium qu'il peut représenter et inversement, un médium donné peut s'exprimer à travers plusieurs employés. Le client n'interagit jamais directement avec les médiums ou les employés, la classe Voyance permet de faire le lien entre toutes les classes.



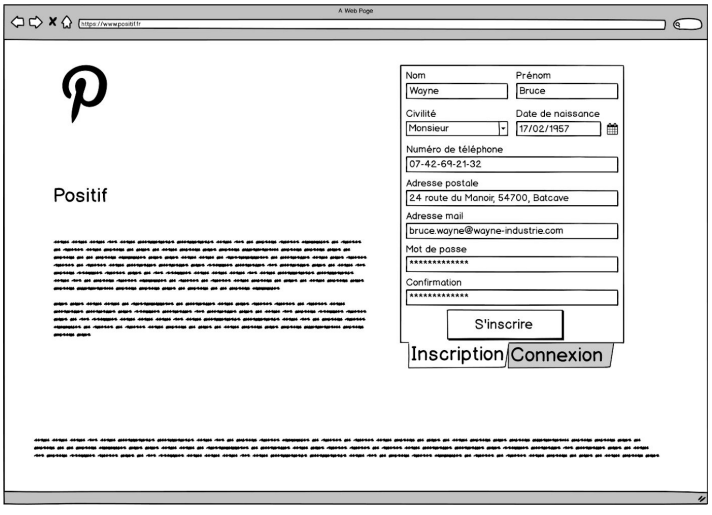
Les classes Astrologue, Voyant ainsi que Tarologue, hérite de la classe Medium. En effet ces trois types de médiums possède un nom et un descriptif en commun, attributs qui se retrouvent dans la classe Medium. Nous avons défini cette dernière comme classe abstraite pour que l'on ne puisse pas instancier de médium, en effet tous les médiums possèdent forcément un type qui est soit Tarologue, Voyant ou Astrologue.

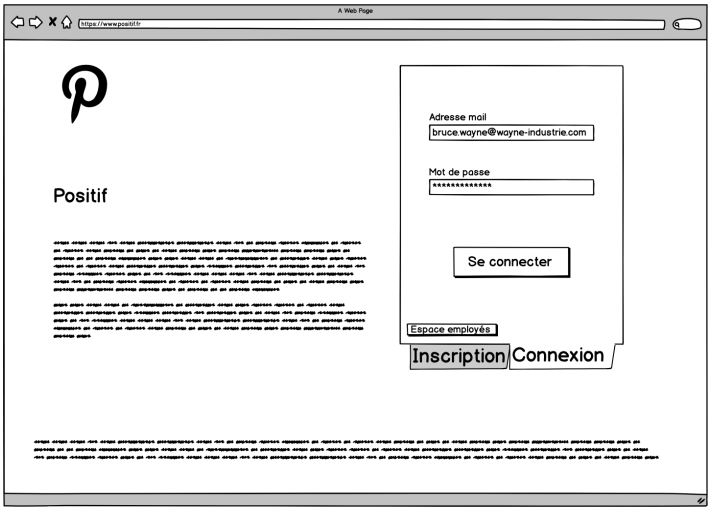
III. Cas d'utilisation et IHM

Vous pourrez retrouvez les fenêtre sur le lien suivant : https://balsamiq.cloud/shbpeag/pugrgtf

A. IHM en commun

Page Connexion/Inscription

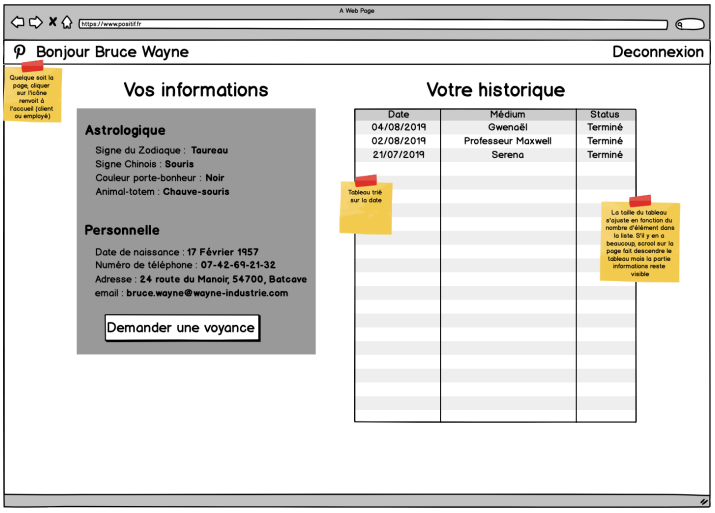




Intention	Contrôle	Action	Réponse	Service
Se connecter	Bouton	clic	Renvoie la page Accueil du client	connecterClient(String mail, String mdp)
S'inscrire	Bouton	clic	Inscrit le Client en le rentrant dans la base	inscrireClient(Client c)
Se connecter en tant qu'employé	Bouton	clic	Renvoie la page Accueil de l'employé	connecterEmploye(String mail, String mdp)

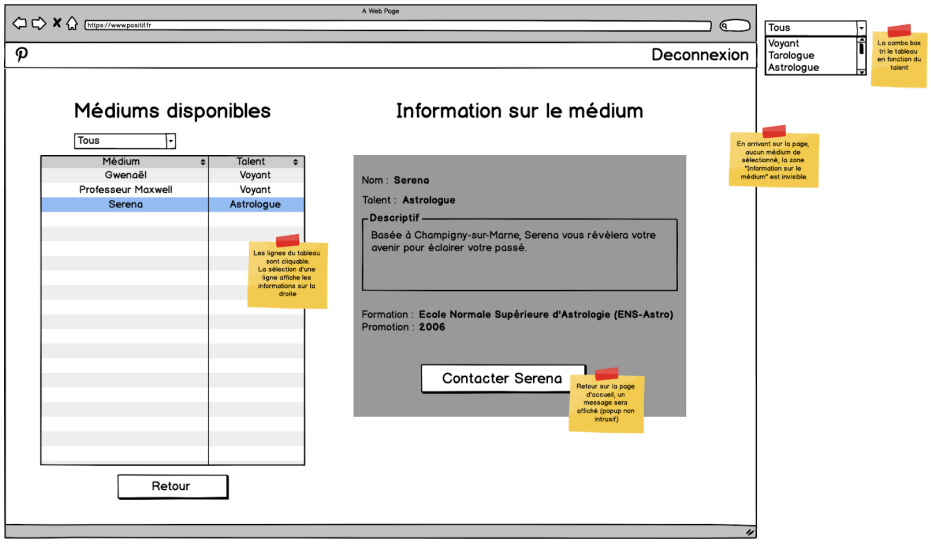
B. IHM Client

Page accueil client



Intention	Contrôle	Action	Réponse	Service
Init	/	/	Rempli l'historique du client	
Déconnexion	bouton	clic	Retour à la page Connexion/Inscritpion	
Demander une voyance	bouton	clic	Ouverture de la page Sélection de voyance	

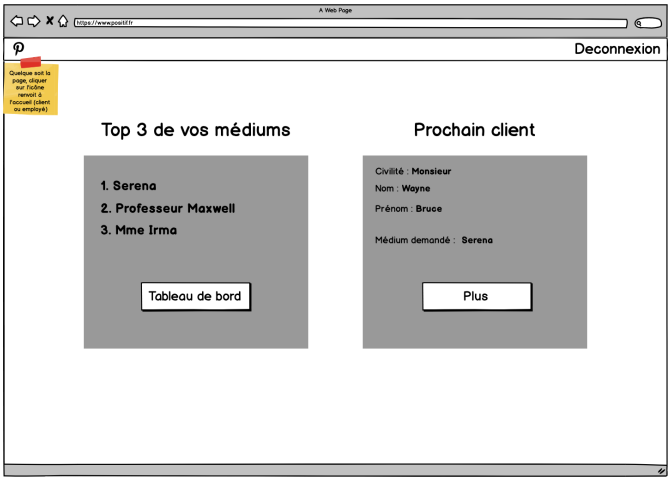
Page Demande de voyance

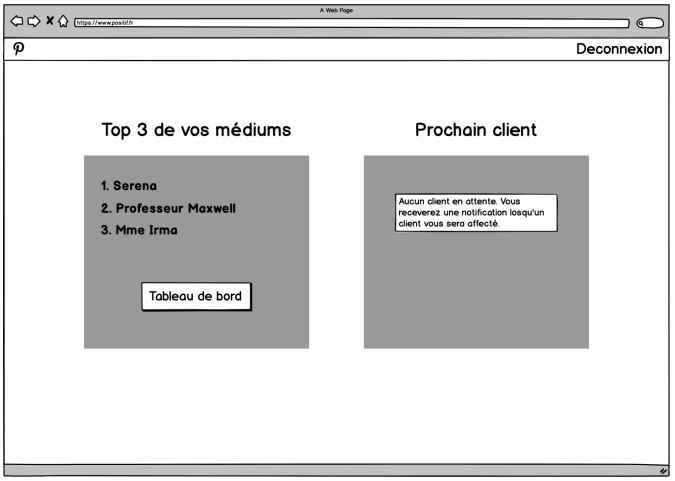


Intention	Contrôle	Action	Réponse	Service
Init	/	/	Tableau remplit avec tous les talents triés par ordre alphabétique du nom	getTousMediums()
Retour	bouton	clic	Retour sur la page Accueil Client	
Filtrer Médiums	menu déroulant	clic puis sélection	Sélection sur le tableau des médiums. Seuls les talents sélectionnés sont affichés	getTousVoyants() getTousAstrologues( ) getTousTarologues() getMediumsParTyp (boolean voyant, boolean astro, boolean taro)
Détail médium	ligne dans le tableau	clic	Affichage des détails du médium et du bouton Contacter Médium	findMediumParId(int id) findMediumParNom(String nom
Contacter le médium	bouton	clic	Envoie d'une notification à l'employé, retour sur la page Accueil Client avec l'historique mis à jour	demanderVoyance(C lient c, Medium m)

C. IHM Employé

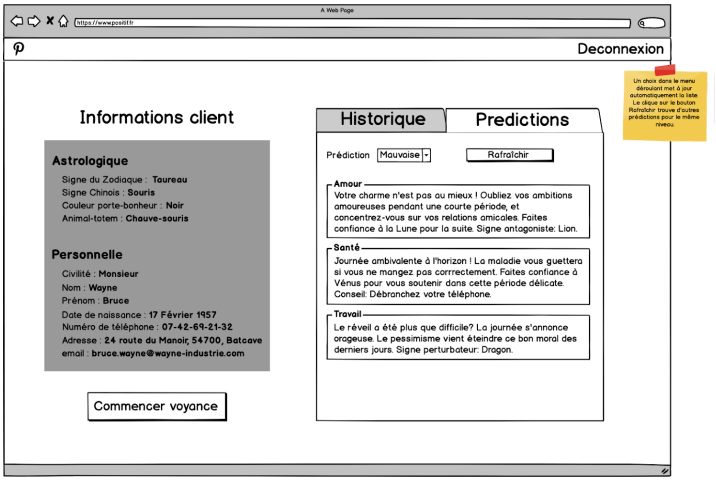
Page Accueil Employé

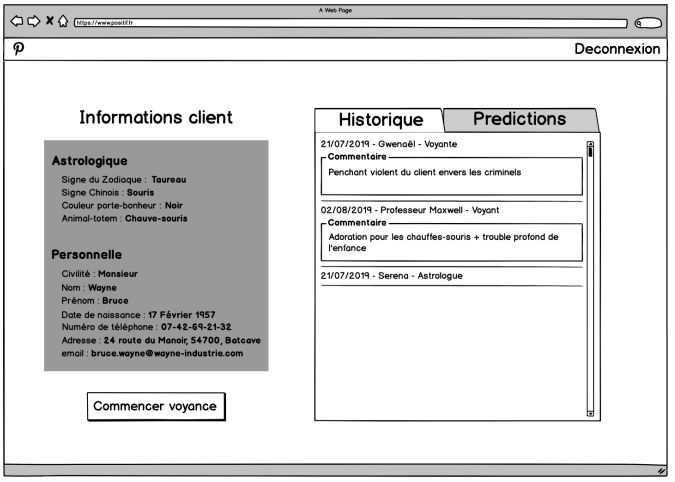




Intention	Contrôle	Action	Réponse	Service
Init	/	/	Si il n'y pas de client, un message descriptif s'affiche à droite Si il y a un client en attente, ses informations s'affichent à droite	getClientEnAttente(Employe e) getVoyanceEnAttente(Client c, Employe e)
Déconnexion	bouton	clic	Retour sur la page Login	
Voir détails client	bouton	clic	Ouvre la page Début Voyance	
Afficher tableau de bord	bouton	clic	Ouvre la page correspondant au tableau de bord	

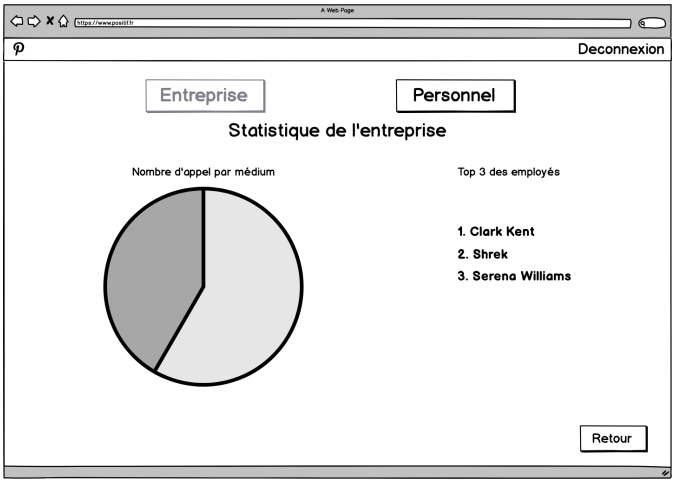
Page Début Voyance

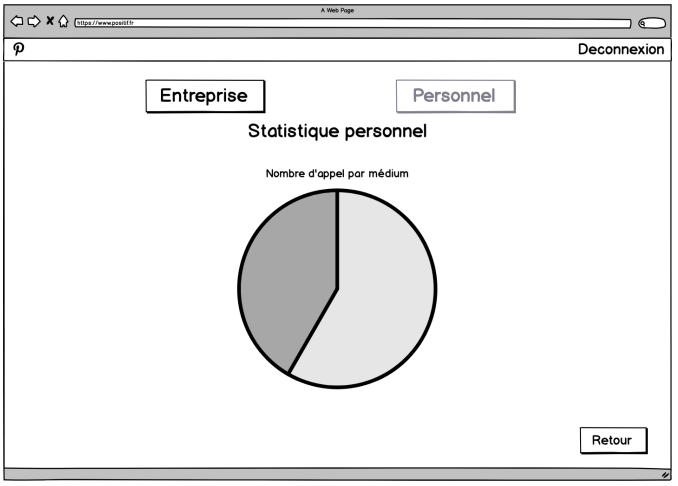




Intention	Contrôle	Action	Réponse	Service
Init	/	/	Affichage des informations personnelles du client concerné ainsi que son historique de voyance	getClientEnAttente(Employe e)
Commencer voyance	bouton	clic	Envoie de la notification au client Cache le bouton Commencer voyance et affiche le bouton Fin voyance Ajoute l'onglet Prédictions	commencerVoyance(Voyance v)
Accéder au prédiction	Onglet	clic	Affichage des prédictions standards	getPredictions(Client c, int niveau)
Changer le niveau des prédictions	Menu déroulant	clic	Ajuste les prédictions avec le niveau choisi	getPredictions(Client c, int niveau)
Rafraîchir les prédictions	bouton	clic	Change les prédictions en gardant le même niveau	getPredictions(Client c, int niveau)
Finir voyance	bouton	clic	Ouvrir un champ texte pour mettre le commentaire de la voyance réalisée avec un bouton Valider	
Valider Commentaire	bouton	clic	La voyance est enregistrée avec le commentaire Retour sur la page Accueil Employé	finirVoyance(Voyance v)

Page Tableau de Bord





Intention	Contrôle	Action	Réponse	Service
Init	/	/	Affichage du tableau de bord avec deux onglets : Entreprise et Personnel Ouverture sur l'onglet Personnel (le bouton Personnel n'est plus cliquable)	findEmployeParId(int id) findEmployeParMail(String mail)
Afficher Tabeau de bord Entreprise	bouton	clic	Affichage des statistiques de l'entreprises sur les Médiums et les Employés	getClassementMedium() getClassementEmploye()
Afficher Tableau de bord Personnel	bouton	clic	Affichage des statistiques personnelles sur les Médiums	getClassementMediumPourEmploye(Employe e)
Retour	bouton	clic	Retour sur la page Accueil Employé	

IV. Spécification des services

Selon l'utilisateur ainsi que où se situe ce dernier dans l'IHM, différents services lui seront proposés.

A. Service Client

Service d'inscription

```
/**
* Inscrit un client à Positif. Envoie une message en cas de succès
ou
* d'échec. Le signe du zodiac, chinois, la couleur favorite et
l'animal
* totem sont calculés et remplace les valeurs existante dans le
paramètre.
* @param c Le client à inscrire
* @return Le client inscrit avec les informations mises à jour.
*/
public Client inscrireClient(Client c)
```

Service de connexion

```
/**
* Connecte un Client.
* @param mail Mail saisi (utilisé comme identifiant)
* @param mdp Mot de passe saisi.
* @return Le client si celui s'est connecté. Null si le mail et/ou
mot de
* passe sont invalides, ou si une erreur est survenue.
*/
public Client connecterClient(String mail, String mdp)
```

Service de recherche de médium

```
/**
* Permet d'avoir la liste de tous les médiums.
* @return La liste des médiums.
*/
public List<Medium> getTousMediums()

/**
* Permet d'avoir la liste de tous les voyants.
* @return La liste des voyants.
*/
public List<Medium> getTousVoyants()

/**
* Permet d'avoir la liste de tous les astrologues.
* @return La liste des astrologues.
*/
public List<Medium> getTousAstrologues()

/**
* Permet d'avoir la liste de tous les tarologues.
* @return La liste des tarologues.
*/
public List<Medium> getTousTarologues()

/**
* Permet d'avoir la liste de différents type de médiums
(Tarologues,
* Astrologues, Voyants).
* @param voyant true pour avoir la listes des voyants.
* @param astro true pour avoir la listes des astrologues.
* @param taro true pour avoir la listes des voyants.
* @return la liste des médiums avec les types demandés.
*/
```

```
public List<Medium> getMediumsParType(boolean voyant, boolean
astro, boolean taro)
```

Service de demande de voyance

```
/**
* Permet de demander une voyance pour un client. Une nouvelle
* voyance est crée avec un status à 0 et les différentes listes
* employé est choisi automatiquement, il n'est démormais plus
disponible. Une notification est envoyée
* à l'employé
* @param c Le client qui demande la voyance.
* @param m Le médium choisi par le client pour la voyance.
* @return La voyance créée, null si une erreur est survenue.
*/
public Voyance demanderVoyance(Client c, Medium m)
```

B. Service Employé

Service de connexion

```
/**
* Connecte un employe.
* @param mail Mail saisi (utilisé comme identifiant)
* @param mdp Mot de passe saisi.
* @return L'employe si celui s'est connecté. Null si le mail et/ou
* mot de passe sont invalides, ou si une erreur est survenue.
*/
public Employe connecterEmploye(String mail, String mdp)
```

Service d'obtention du client en attente

```
/**
* Retourne le client en attente d'un employé (càd dont le status
d'une de
* ces voyances est à 0).
* @param e L'employé
* @return Le client en attente, null si il n'y en a aucun ou
* qu'une erreur est survenue.
*/
public Client getClientEnAttente(Employe e)
```

Service d'obtention de la voyance en attente

```
/**
* Retourne la voyance en attente d'un employé pour un client
* donnée. C'est à dire une voyance commune au client et à
* l'employé et qui n'as pas encore commencé (cad status = 0).
* @param c Le client.
```

```
* @param e L'employé.
* @return La voyance en attente, null si aucune.
*/
public Voyance getVoyanceEnAttente(Client c, Employe e)
```

Service de début de voyance

```
/**
* Permet de commencer une voyance (déclencher par l'employé). Le
* status de la voyance passe à 1 et une notification est envoyée
* au client.
* @param v La voyance à commencer.
* @return La voyance mise à jour.
*/
public Voyance commencerVoyance(Voyance v)
```

Service de prédiction

```
/**
* Retourne les prédictions pour un client donné. Le niveau passé
en
* paramètre est utilisé pour les 3 prédictions.
* @param c Le client
* @param niveau Le niveau est compris entre 1 (mauvais) et 4
* (parfait)
* @return Retourne une liste de String, indice 0 = Amour, 1 =
* Santé, 2 =
* Travail. Retourne null si une erreur est survenue.
*/
public List<String> getPredictions(Client c, int niveau)
```

Service de fin de voyance

```
/**
* Doit être appelé pour clôre la voyance, c'est à dire une fois
* que le commentaire à été saisi. Le status de la voyance passe à
* 2. L'employé est à nouveau disponible.
* @param v La voyance avec le commentaire.
* @return La voyance mise à jour, null si une erreur s'est
produite.
*/
public Voyance finirVoyance(Voyance v)
```

Service de classement des médiums par employé

```
/**
* Retourne une map des médiums avec le nombre de fois que
* l'employé l'a interprété.
* @param e L'employé
```

```
* @return Map<Medium, Integer> trié par ordre décroissant du
* nombre de voyance avec le médium.
*/
public Map<Medium,Integer> getClassementMediumPourEmploye(Employe
e)
```

Service de classement des médiums de l'entreprise

```
/**
* Retourne une map des médiums avec le nombre de fois qu'ils ont
* été interprété.
* @return Map<Medium, Integer> trié par ordre décroissant du
* nombre de voyance avec le médium.
*/
public Map<Medium, Integer> getClassementMedium()
```

Service de classement des employés de l'entreprise

```
/**
* Retourne une map des employés avec le nombre de fois qu'ils ont
* réalisé une voyance.
* @return Map<Medium, Integer> trié par ordre décroissant du
* nombre de voyance avec le médium.
*/
public Map<Employe, Integer> getClassementEmploye()
```

C. Services en commun

Service de recherche de Client

```
/**
* Permet de retrouver un client par son id.
* @param id L'id du client
* @return Le client correspondant à cette id, null s'il n'y en a
* pas où qu'une erreur est survenue.
*/
public Client findClientParId(int id)

/**
* Permet de retrouver un client par son mail (qui est unique).
* @param mail Le mail du client
* @return Le client correspondant à cette id, null s'il n'y en a
* pas où qu'une erreur est survenue.
*/
public Client findClientParMail(String mail)
```

Service de recherche d'employé

```
/**
* Permet de retrouver un employe par son id.
* @param id L'id de l'employé
* @return L'employé correspondant à cette id, null s'il n'y en a
* pas où qu'une erreur est survenue.
*/
public Employe findEmployeParId(int id)

/**
* Permet de retrouver un employe par son mail.
* @param mail Le mail du client
* @return L'employé correspondant à cette id, null s'il n'y en a
* pas où qu'une erreur est survenue.
*/
public Employe findEmployeParMail(String mail)
```

Service de recherche de médium

```
/**
* Permet de trouver un médium par son id.
* @param id L'id du médium à trouver.
* @return Le médium correspondant à cette id, null s'il n'y en a
* pas où qu'une erreur est survenue.
*/
public Medium findMediumParId(int id)

/**
* Permet de trouver un médium par son nom (unique).
* @param nom Le nom du médium
* @return Le médium correspondant à cette id, null s'il n'y en a
* pas où qu'une erreur est survenue.
*/
public Medium findMediumParNom(String nom)
```