Cloud

-- Contexte

Vous travaillez pour une entreprise qui souhaite mettre en place un système de Storage as a service (STaaS), permettant de mettre à disposition de ses clients un stockage de fichiers à distance. Vous êtes en charge d'une partie de l'application web permettant de gérer ce service.

-- Règles de gestion

Les clients peuvent disposer d'espaces de stockage (nommés disques) permettant de stocker leurs données. Chaque disque est loué à un certain tarif par le client ; le tarif comprend :

- Un prix (mensuel)
- Un quota (capacité de stockage exprimée dans une unité (o, Ko, Mo, Go ou To)
- Une marge de dépassement, à ne pas dépasser sous peine d'être facturé d'un surcoût.

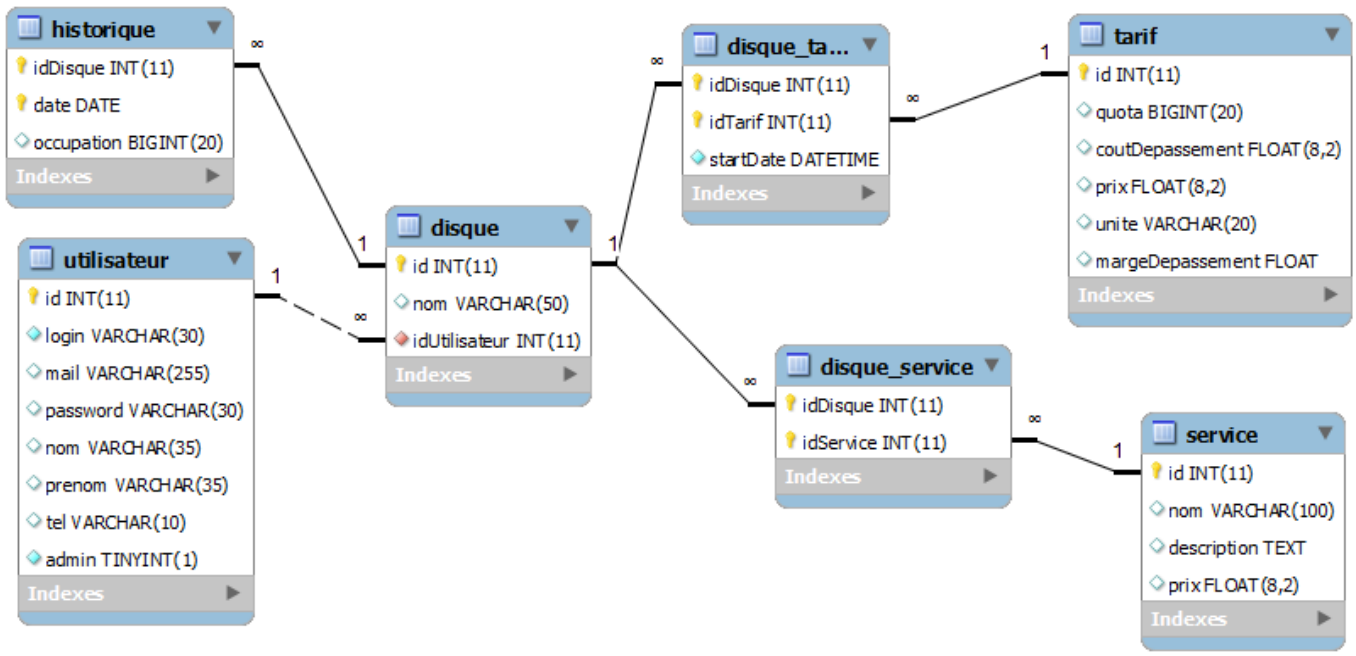
Le client a la possibilité de changer de tarif à tout moment pour chaque disque, de façon à diminuer ou augmenter son quota, en fonction de ses besoins. L'historique des tarifs associés à un disque est conservé.

Le tarif courant appliqué à un disque est celui correspondant à la date la plus récente.

Chaque location de disque peut être associée à des services (backup, loadBalancer...).

L'historique d'occupation des disques est obtenu de manière automatisée par un script PHP lancé de manière quotidienne par une tâche CRON.

-- Schéma de la base



Script de création de la base Cloud à exécuter sur votre serveur

-- Fonctionnalités à implémenter

-- Récupération du tarif actuel (2 points)

//TODO 4.1

Dans la table **Disque_tarif**, sont stockés les tarifs appliqués à un disque à partir d'une certaine date (champ **startDate**). Le tarif actuel d'un disque correspond donc au dernier tarif appliqué, celui dont la startDate est la plus récente.

En utilisant les relations définies sur les modèles, et en privilégiant une approche objet, implémentez la méthode retournant le tarif actuel d'un disque dans la classe **ModelUtils** :

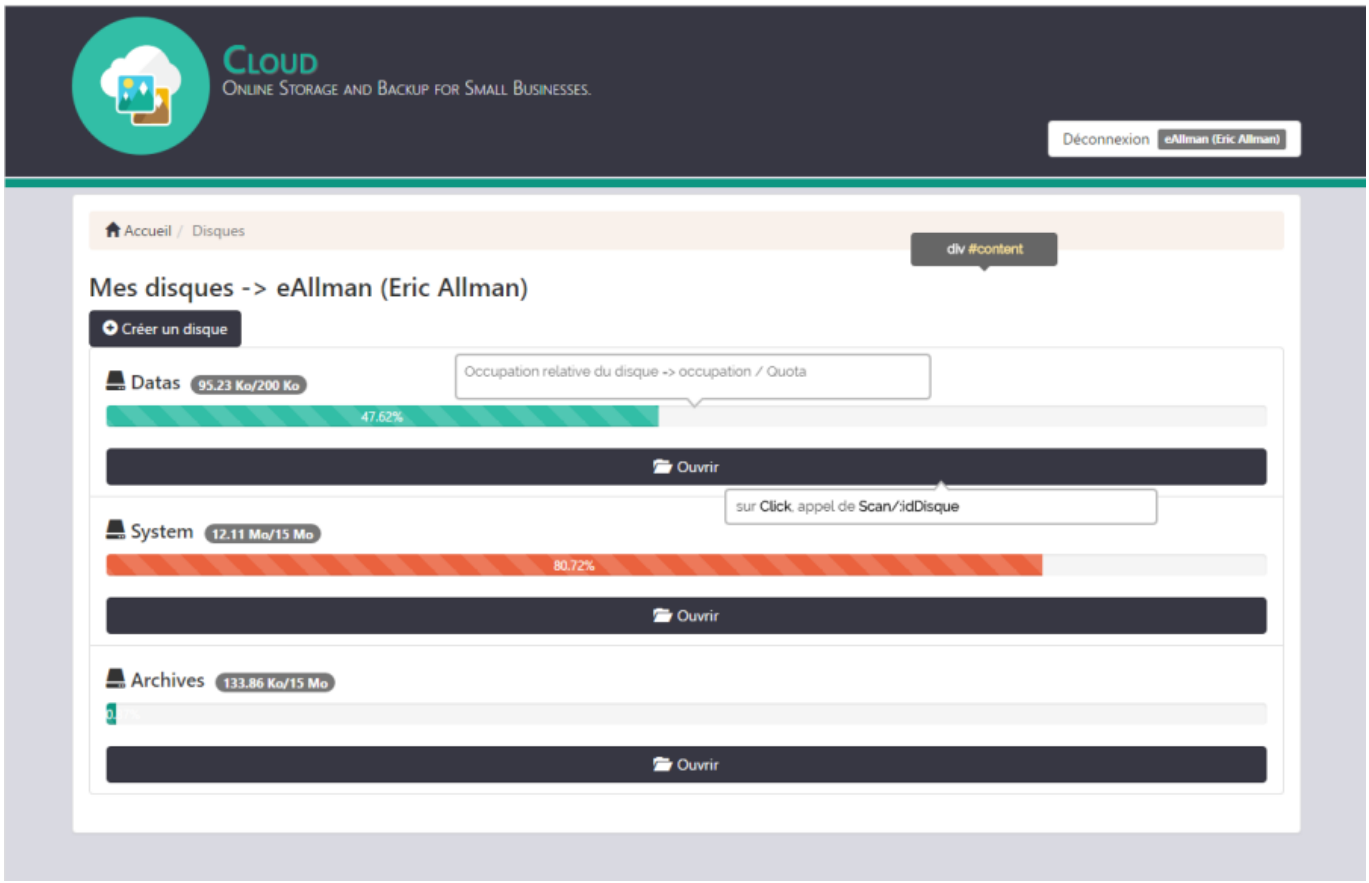
```
/**
 * Retourne le tarif appliqué actuellement à $disque
 * @param Disque $disque
 * @return Tarif tarif actuel de $disque
 */
public static function getDisqueTarif($disque){
    //TODO
}
```

-- Affichage des disques d'un client (7 points)

//TODO 4.2

A l'adresse **MyDisques/index**, on souhaite afficher les disques de l'utilisateur actuellement connecté.

L'utilisateur connecté est obtenu par l'appel de la méthode **Auth::getUser($controller)** où **$controller** est le contrôleur actif.



Indications

Elément	**Indications**
Contrôleur	MyDisques
Action	index
Utilisateur connecté	L'utilisateur connecté est obtenu par l'appel de la méthode **Auth::getUser($controller)** où **$controller** est le contrôleur actif.
Occupation, Quota	le quota est obtenu sur le tarif actuel du disque, la classe ModelUtils permet de connaître l'occupation en cours du disque
Composants visuels Bootstrap	on pourra utiliser les composants Phalcon-Jquery **htmlProgressbar**, **htmlGlyphButton**, **htmlListgroup**

Le style des progressbars doit donner une indication sur le taux d'occupation en % :

Style	**Valeurs**
info	de 0 à 10%
success	de 10 à 50%
warning	de 50 à 80%
danger	plus de 80%

Le composant **HtmlProgressbar** dispose d'une méthode **setStyleLimits** permettant de modifier le style de la **progressbar** en fonction de sa valeur

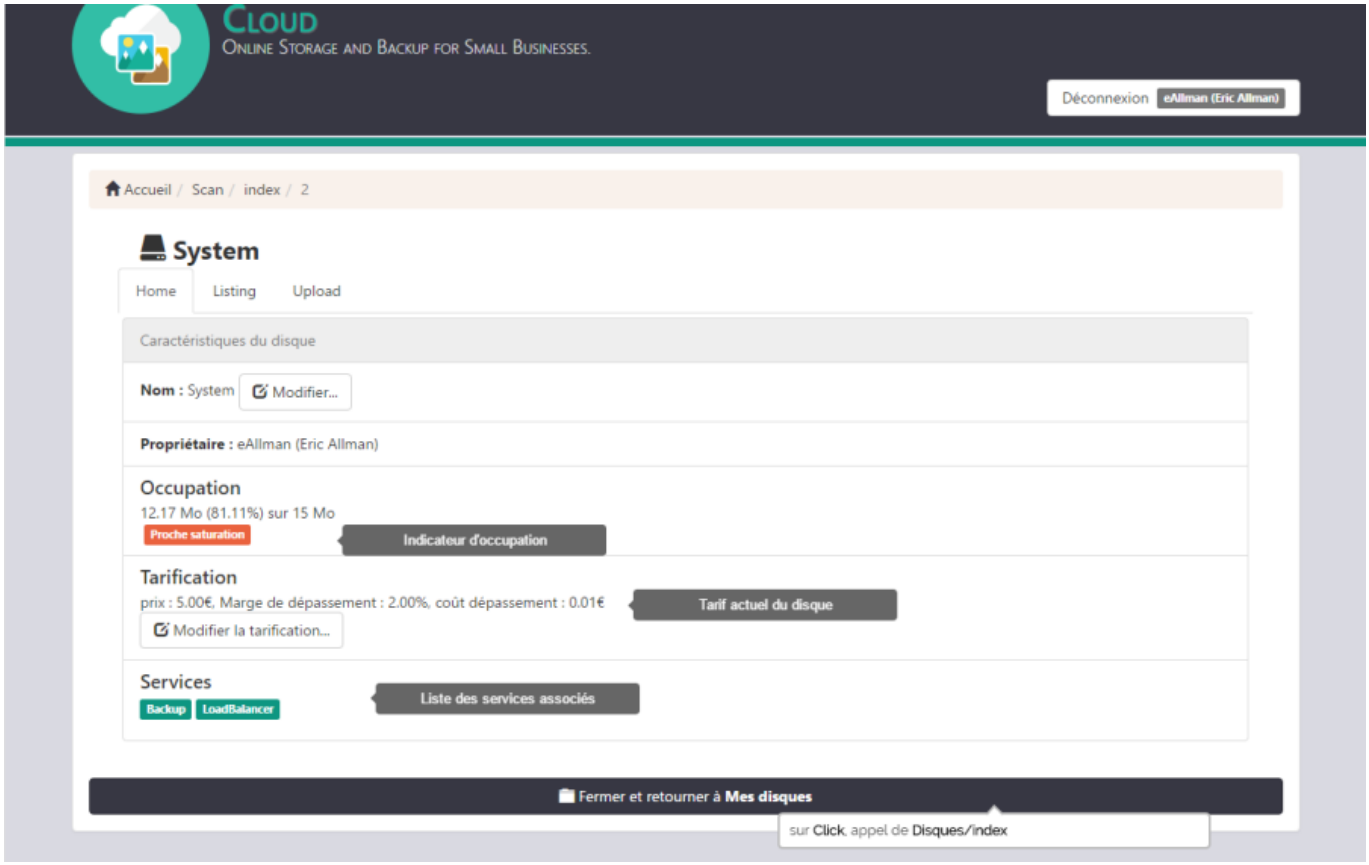
-- Affichage d'un disque (6 points)

//TODO 4.3

Apportez les modifications aux relations du modèle **Disque** pour pouvoir accéder à ses services (ManyToMany

?).

A l'adresse **Scan/:idDisque**, afficher l'interface suivante :



Elément	**Indications**
Contrôleur	Scan
Action	index
Paramètre	idDisque
Utilisateur connecté	L'utilisateur connecté est obtenu par l'appel de la méthode **Auth::getUser($controller)** où **$controller** est le contrôleur actif.
Occupation, Quota	le quota est obtenu sur le tarif actuel du disque, la classe ModelUtils permet de connaître l'occupation en cours du disque
Composants visuels Bootstrap	on pourra utiliser les composants Phalcon-Jquery **htmlLabel**, **htmlGlyphButton**, **htmlListgroup**

L'indicateur d'occupation (**htmlLabel**) doit donner une indication sur le taux d'occupation en % :

Texte	**Style**	**Valeurs**
Peu occupé	info	de 0 à 10%
RAS	success	de 10 à 50%
Forte occupation	warning	de 50 à 80%
Proche saturation	danger	plus de 80%

Conseils importants :

- Ajax est recommandé pour toutes les requêtes
- Penser à l'éventuel **{{script_foot}}** dans les vues
- Préférer l'instanciation des composants phalcon JQuery en utilisant les méthodes de l'objet **$this→jquery→bootstrap()**

-- A poursuivre en dehors du TP...

-- Création d'un disque (2 points)

//TODO 4.4.1

A partir de l'adresse **Disques/frm**

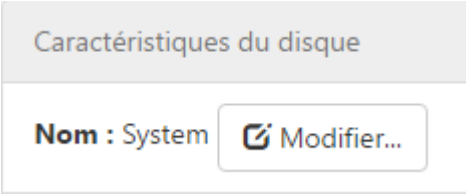
Afficher le formulaire de création d'un disque.

Soumettre le formulaire à l'adresse **Disques/update** pour créer le disque puis rediriger vers l'adresse **Scan/index/:idDisque**

Penser à le créer physiquement.

-- Renommage disque (1 points)

A partir de l'adresse **Scan/index/:idDisque**



Implémenter la modification du nom d'un Disque (avec renommage éventuel du dossier associé + vérification du nom)

-- Contrôle d'accès (2 points)

URL	Accès	Résultat/message
Disques/index	Accessible uniquement pour un utilisateur connecté	Affiché message sur l'absence de connexion et proposer la connexion
Scan/index/:idDisque	Accessible uniquement pour un disque appartenant à l'utilisateur connecté	Afficher un message d'erreur “Accès à une ressource non autorisée”

Utiliser le dispatcher phalcon pour ce faire.

From:
http://slamwiki2.kobject.net/ - **SlamWiki 2.1**

Permanent link:
http://slamwiki2.kobject.net/slam4/php/phalcon/project/cloud?rev=1458257208

Last update: **2019/08/31 14:41**