Projet virtualhosts

Vous travaillez pour une entreprise proposant des services d'hébergement. Vous êtes chargé d'élaborer une application web permettant à l'entreprise et à ses clients de gérer la configuration de leurs applications web.

Résumé

Projet initial à utiliser	Projet Github à cloner
Outils	• Semantic-UI • phpMv-UI • Phalcon php
Principales fonctionnalités	• Module client/Admin-client
Livraison	• sur Moodle

Ressources

- Phalcon framework
- phpMv-UI
- API phpMv-UI

Règles de gestion

L'application permet aux utilisateur de gérer et de configurer facilement leur hôtes virtuels (**Virtualhost**), présent sur des serveurs dédiés (**Host**) ou simplement mutualisés (dans ce cas le client ne connaît que le virtualhost).

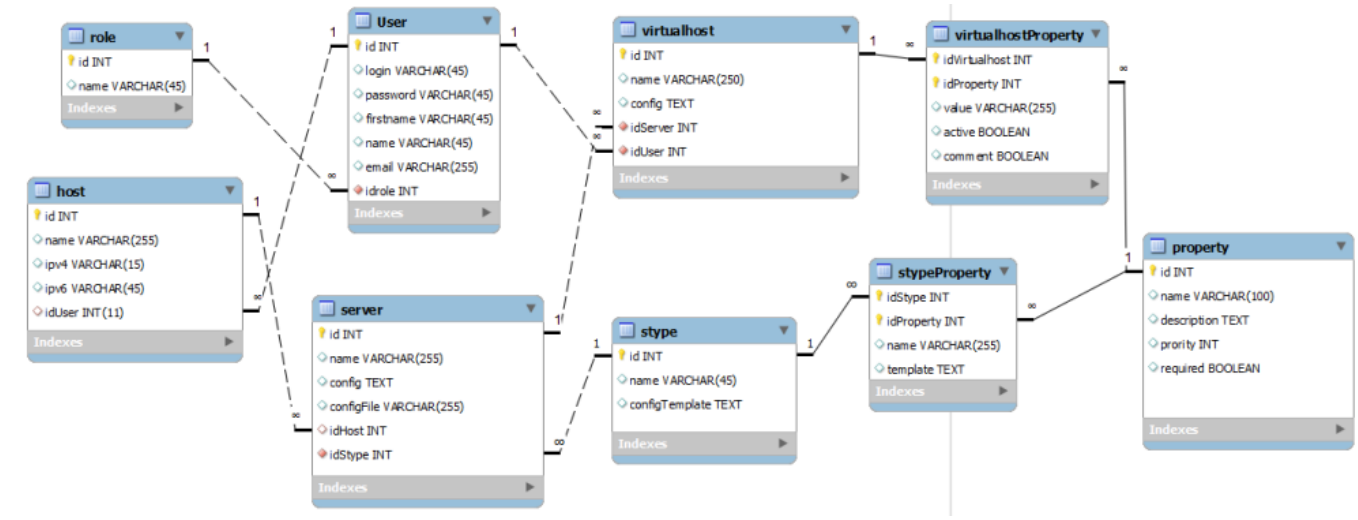
Sur les machines (**Host**) sont installés des serveurs Http (**Server**).

Ces serveurs sont d'un certain type (**sType**) : Apache, Node, NginX...
Le type de Serveur définit les propriétés de configuration qu'il peut recevoir (**sTypeProperty**).

La configuration d'un virtualhost est stockée dans la table virtualhostProperties, qui permettra ensuite de générer automatiquement le fichier de configuration.

Ce fichier généré pourra être ensuite uploadé sur le serveur, et le service web rechargé pour la prise en compte de la nouvelle configuration.

Schéma de la base



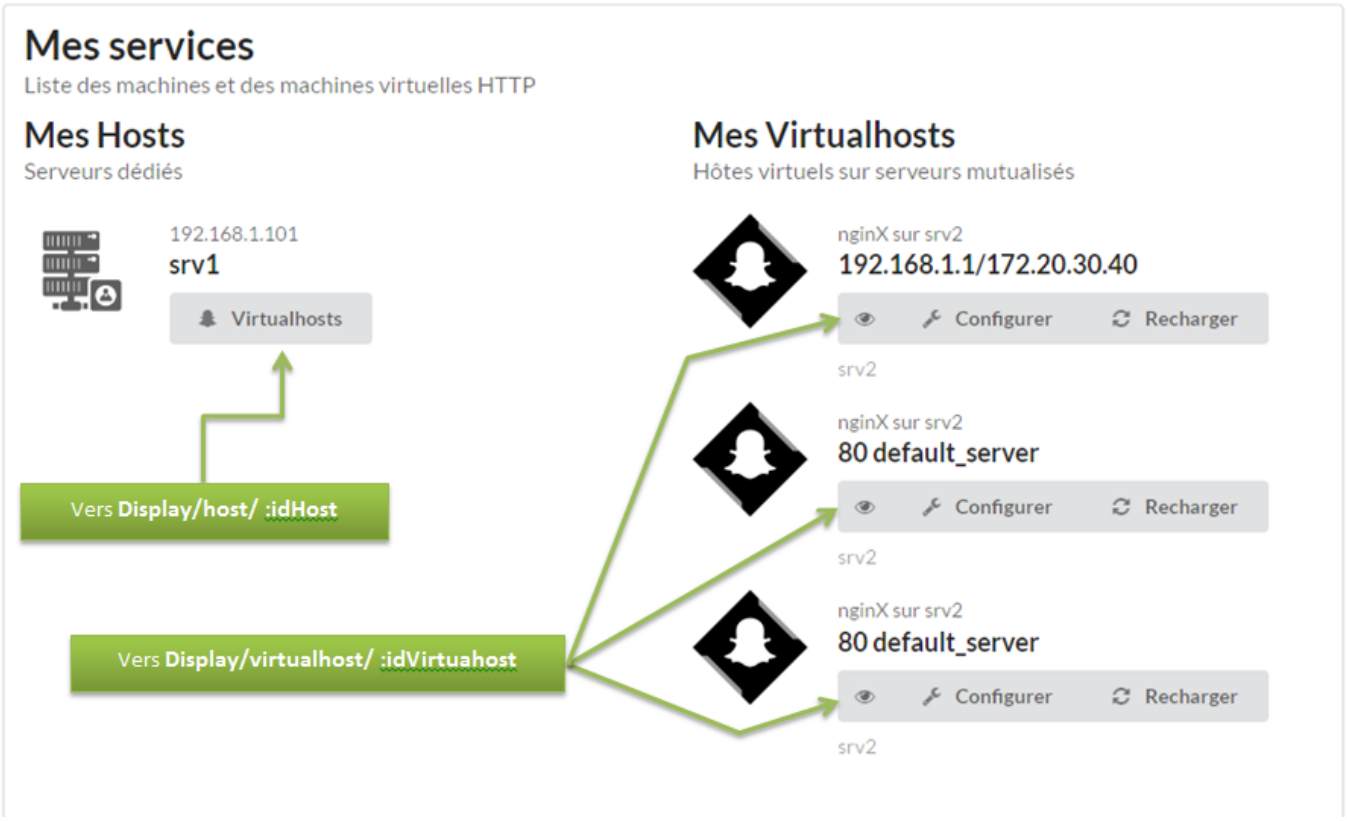
Détail des fonctionnalités à mettre en place

Le dossier root de votre application devra être de la forme : **phalcon-prenom.nom**

-- Url /My (5 points)

//TODO 1

Affiche la liste des Hosts et virtualhosts de l'utilisateur authentifié.



Données :

- L'interface doit faire apparaître pour l'utilisateur connecté :
  - ses Hosts

- ses virtualhosts (exclure celles qui sont présentes sur les hosts appartenant à l'utilisateur)

Composants utilisables :

Semantic UI (phpMv-UI) :

- ui grid (htmlGrid)
- ui items (htmlItems)
- ui header (htmlHeader)
- ui button (htmlButton ou htmlButtonGroups)

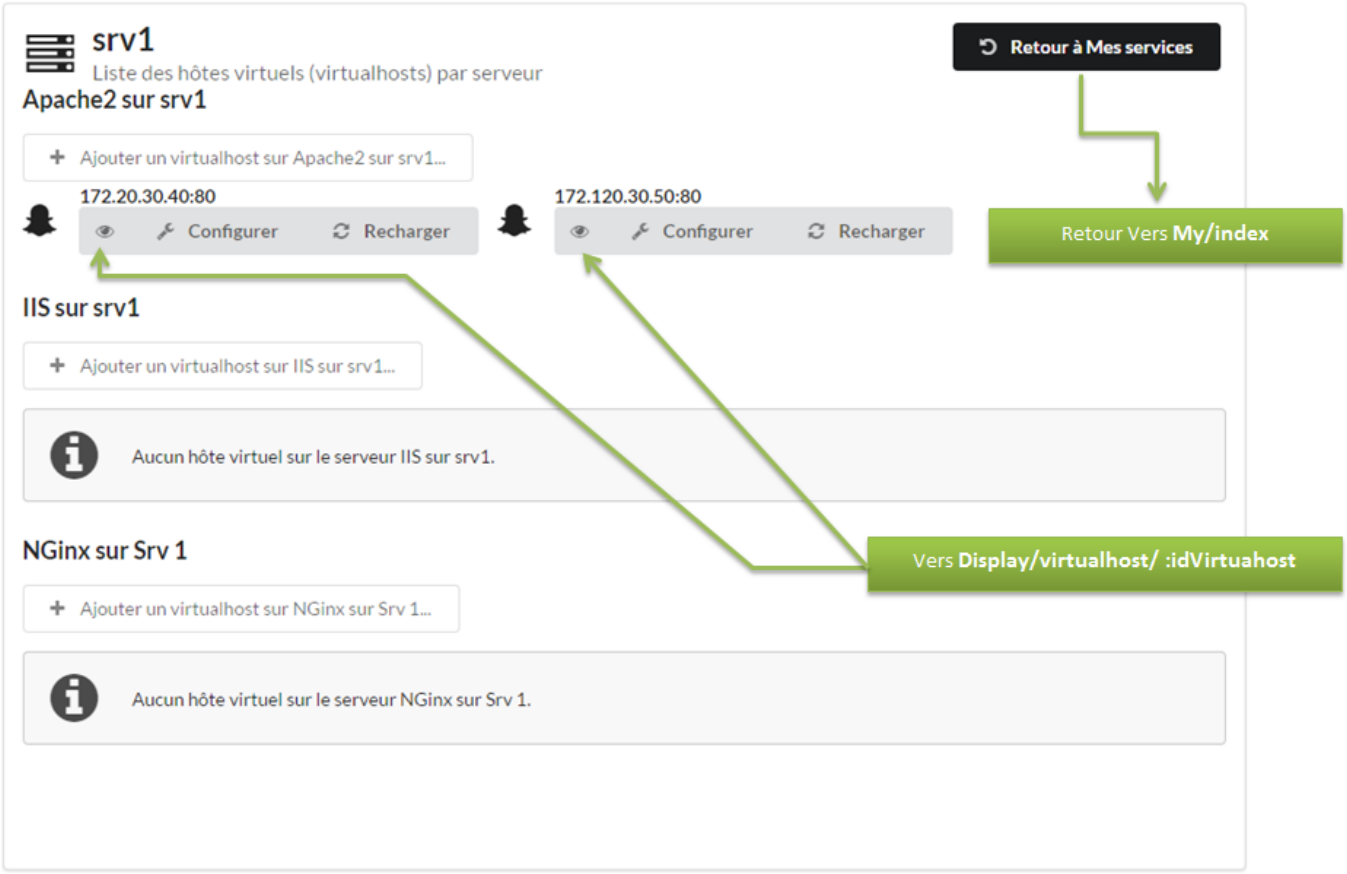
Info

Pour info, la div qui reçoit le résultat des requêtes Ajax est **#content-container**

-- Url /Display/host/:idHost (5 points)

//TODO 2

Affiche la liste des virtualhosts correspondant au idHost de l'Host passé en paramètre.



Données :

- Pour le Host passé en paramètre, l'interface fait apparaître :
  - les Serveurs installés
  - les virtualhosts par serveur

Composants utilisables :

Semantic UI (phpMv-UI) :

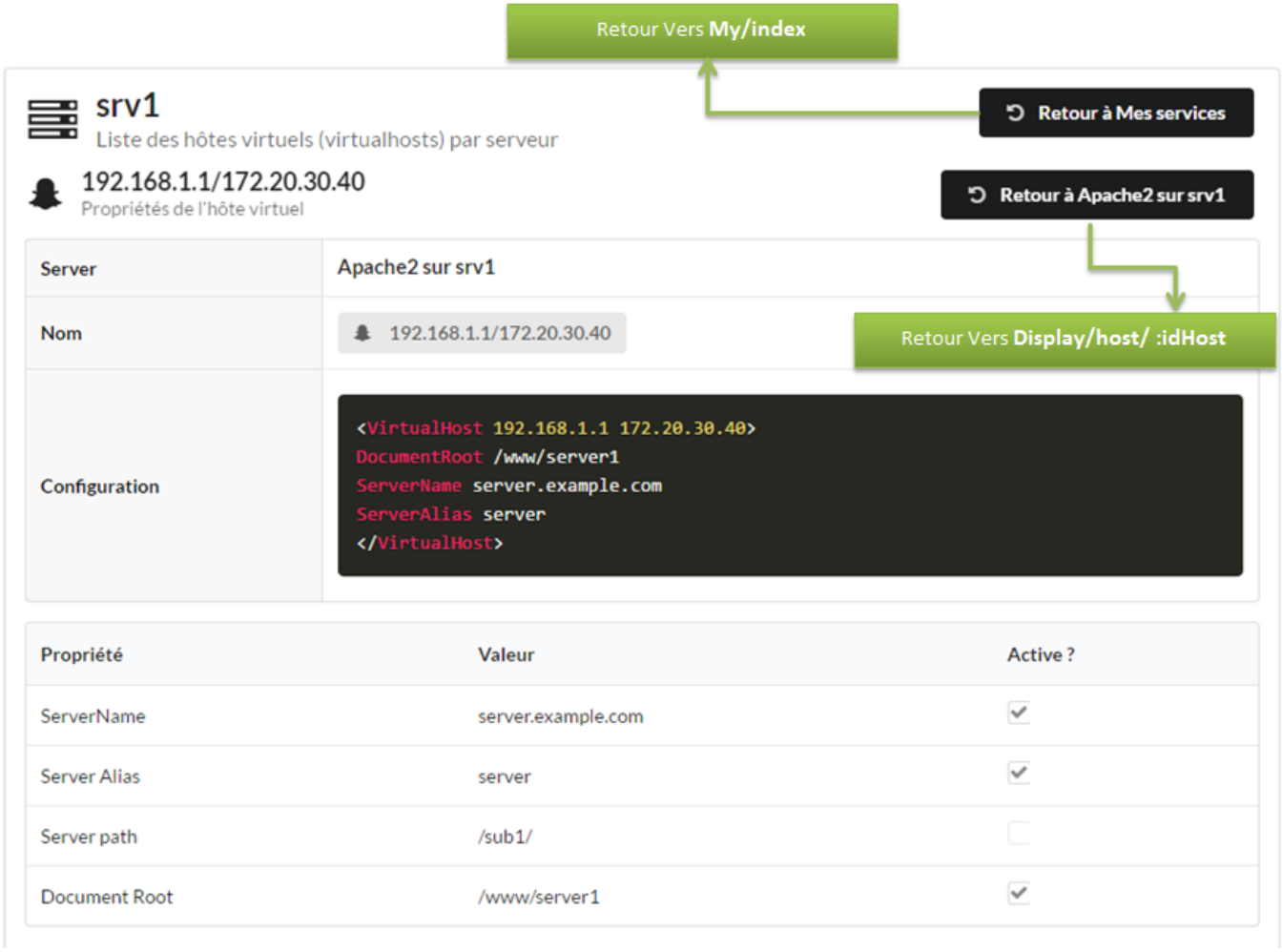
- ui list (htmlList)
- ui header (htmlHeader)
- ui message (htmlMessage)

-- Url /Display/virtualhost/:idvirtualhost (5 points)

//TODO 3.a //TODO 3.b

Affiche le virtualhost correspondant au idVirtualhost de l'hôte virtuel passé en paramètre.

Cette fonctionnalité est accessible depuis **My** et **Display/host/:idHost**



Données :

- Pour le Virtualhost passé en paramètre, l'interface fait apparaître :
  - les informations liées au virtualhost
  - la liste des propriétés du virtualhost (virtualhostProperties)

Coloration syntaxique :

Le champ **config** sera colorisé avec Prism :

- On pourra utiliser la fonction setValueFunction pour transformer le champ config du virtualhost dans le DataElement.
- Pour obtenir la coloration Prism, le champ devra être entouré des balises **pre** et **code**, et spécifier la class Css à utiliser pour coloriser (le champ **prism** de la table **Stype** précise la classe de coloration à utiliser):

```
"<pre class='language-".$prism."'><code>".$conf."</code></pre>"
```

La coloration est ensuite réalisée par l'appel du script :

```
    $this->jquery->exec("Prism.highlightAll();",true);
```

Composants utilisables :

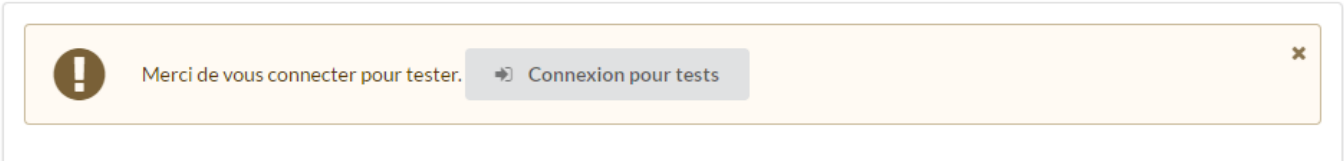
phpMv-UI :

- dataElement
- dataTable

-- Url /Auth/pleaseLogin (2 points)

//TODO 4

Protection du contrôleur My contre les utilisateurs non authentifiés.



Comportement :

- L'accès aux actions du contrôleur My renvoie à l'URL Auth/pleaseLogin

Composants utilisables :

- librairie Auth du dossier library

-- Idées d'approfondissement : Bonus (3 points)

- Saisie/modification de virtualhost (avec ses propriétés)
- Génération et intégration du script de conf généré avec la librairie **ConfGenerator**

Contraintes techniques

- L'application sera développée en PHP objet, elle utilisera Phalcon et phpMv-UI.
- Elle respectera au mieux la séparation des couches (objets Métiers), classes techniques et vues (interfaces web de saisie et d'affichage).
- Elle utilisera la base de données Mysql fournie en annexe.
- L'utilisation d'ajax est une obligation.
- Semantic-UI sera utilisé pour la partie présentation.

Fichiers

- Tout est dans le projet Git (y compris la base de données)

Compléments

Bonnes pratiques

- respecter l'architecture MVC
- respecter la Normalisation HTML 5/Css 3
- Structurer les fichiers et dossiers de manière cohérente et respecter les consignes
- Nommer en respectant les normes et de manière significative (Contrôleurs, vues, méthodes, variables...)

Ne pas oublier de créer le fichier **readme.md** sur votre repo github Phalcon pour y consigner vos exploits...