

Réalisation du projet.

Preuve de concept d'un outil de gestion des connexions des prestataires depuis un bastion (Guacamole).



Organisme : Centre Hospitalier Henri Laborit
Adresse : 370 Av. Jacques Cœur, 86021 Poitiers
Direction : Service Informatique
Auteur : LOUINEAU Noah
Durée : 6 semaines

1. Présentation du projet

L'établissement utilise un bastion (Guacamole) pour autoriser des connexions de prestataires à leurs serveurs. Le service informatique exprime le besoin d'un POC (Proof of Concept) visant à mettre en place un outil permettant une gestion simplifiée des connexions des prestataires depuis ce bastion. L'objectif est de simplifier le processus d'autorisation pour les utilisateurs qui supervisent et autorisent ces connexions.

Parmi les choix de simplification, deux options ont retenu mon attention.

La première étant de créer un bouton qui active et désactive la possibilité que des utilisateurs s'authentifient, l'autre est de paramétrer une plage horaire de connexion dans laquelle les utilisateurs peuvent s'authentifier, après expiration de cette date, il n'est plus possible de s'authentifier.

2. Recherche

Afin de répondre à la demande, je me suis documenté sur le fonctionnement de Guacamole en utilisant sa documentation. ([Doc Guacamole.](#))

Cependant, je me suis mal orienté et je n'ai pas su repérer la bonne solution au départ.

3. Solution pour répondre à la demande

Pour commencer la mission qui m'a été attribuée, j'ai commencé à chercher comment répondre à la demande.

La solution numéro 1. Manipulation de la base de données

La première solution est d'interagir directement avec la base de données en envoyant des requêtes à MariaDB. Cette solution n'était pas adéquate, car elle n'était pas suffisamment sécurisée et pas assez complète.

Le soutien numéro 2. Création d'une application web à l'aide de l'API Guacamole

Pour la 2^e solution, je vais utiliser les fonctions l'API existante de Guacamole qui fonctionne en effectuant des requêtes vers le bastion. En premier temps, j'ai commencé avec du JavaScript avec le protocole AJAX, puis dû à la complexité, j'ai préféré utiliser du PHP et utiliser CURL.

4. Étape avant mise en place des solutions

MODIFIER UTILISATEUR

Identifiant: montech
Mot de passe: *****
Répéter mot de passe:

PROFIL

Nom: Technicien
Adresse Mail: montech@poitiers.fr
Organisation:
Rôle: Technicien

RESTRICTIONS DE COMPTE

Connexion désactivée: ☐
Mot de passe expiré: ☐
Autoriser l'accès après: -- : -- : --
Ne pas autoriser l'accès après: -- : -- : --
Activer le compte après: jj / mm / aaaa
Désactiver le compte après: jj / mm / aaaa
Fuseau horaire utilisateur: Europe Paris

PERMISSIONS

Administration du système: ☐
Créer de nouveaux utilisateurs: ☐
Créer de nouveaux groupes d'utilisateurs: ☐
Créer de nouvelles connexions: ☐
Créer de nouveaux groupes de connexion: ☐
Créer de nouveaux profils de partage: ☐
Modifier son propre mot de passe: ☐

Avant de mettre en place les solutions, il faut créer un utilisateur, ainsi qu'une connexion pour pouvoir essayer mes solutions.

Paramètre d'utilisateur:

Identifiant: Montech

Mot de passe: Montech

Profil : Technicien

Restriction de compte: Aucune

Permissions: Aucune

Groupes: Technicien

Connexions: SSH-SRV1

CONNEXIONS

Connexions en cours Toutes les Connexions

SSH-SRV1

5. Mise en application des différentes solutions

I-Solution numéro 1. Manipulation de la base de données

Pour commencer, je me suis connecté en local à ma base de données.

```
# mysql -u root -p
```

Après, je sélectionne la base de données utilisée par guacamole.

```
MariaDB [(none)]> USE guacadb;
```

J'ai listé toutes les tables avec la commande SHOW TABLES;

```
MariaDB [(none)]> SHOW TABLES;
```

```

+-----+
| Tables_in_guacadb |
+-----+
| guacamole_connection |
| guacamole_connection_attribute |
| guacamole_connection_group |
| guacamole_connection_group_attribute |
| guacamole_connection_group_permission |
| guacamole_connection_history |
| guacamole_connection_parameter |
| guacamole_connection_permission |
| guacamole_entity |
| guacamole_sharing_profile |
| guacamole_sharing_profile_attribute |
| guacamole_sharing_profile_parameter |
| guacamole_sharing_profile_permission |
| guacamole_system_permission |
| guacamole_user |
| guacamole_user_attribute |
| guacamole_user_group |
| guacamole_user_group_attribute |
| guacamole_user_group_member |
| guacamole_user_group_permission |
| guacamole_user_history |
| guacamole_user_password_history |
| guacamole_user_permission |
+-----+

```

Puis j'ai exploré chaque table, afin de comprendre leur fonctionnement. [MariaDB guacadb](#) (Lien vers une Documentation expliquant chaque table.)

Les tables intéressantes sont guacamole_user, car c'est là que nous trouvons la liste des utilisateurs que l'on doit reconnaître grâce à leur entity_ID. Dans la table guacamole_user, nous retrouvons leur autorisation à se connecter, leur nom, leur mot de passe hashé ainsi que leur plage horaire à paramétrer.

```

MariaDB [guacadb]> SELECT*FROM guacamole_entity;
+-----+-----+-----+
| entity_id | name          | type |
+-----+-----+-----+
| 15        | Intertech     | USER |
| 10        | intervenant   | USER |
| 13        | Intervenant-serveur | USER |
| 21        | montech       | USER |
+-----+-----+-----+

```

Pour connaître l'entity_ID d'un utilisateur, il faut afficher la table guacamole_entity.

Afficher la table : MariaDB [guacadb]> SELECT*FROM guacamole_entity;

J'ai remarqué que l'entity_id de l'utilisateur montech est 21.

Afficher la table guacamole_user : MariaDB [guacadb]> SELECT*FROM guacamole_user;

entity_id	disabled*	expired*2	acces_window_start*3	acces_window_end*4	valid_from*5	valid_until*6	timezone	full_name
21	0	0	NULL	NULL	NULL	NULL	Europe/ Paris	montech

*Compte désactivé 1, activé 0

*2 mots de passe expiré 1, pas expiré 0

*3 Droit d'accès à partir de HH/MN/SS

*4 Refus d'accès après HH/MN/SS

*5 Droit d'accès à partir de AAAA/MM/JJ

*6 Refus d'accès après AAAA/MM/JJ

Comme je l'ai expliqué dans la présentation du projet, il y a deux solutions qui permettent d'autoriser un utilisateur à s'authentifier lorsqu'on lui autorise.

Solution A : Créer un bouton qui active ou désactive l'autorisation d'un utilisateur à s'authentifier.

Ou

Solution B : Créer un formulaire dans lequel on peut définir une plage horaire pendant laquelle l'utilisateur peut s'authentifier.

Pour appliquer la solution A, il faut influencer la colonne 'disabled' dans la table "guacamole_user".

Pour faire cela, je dois utiliser la commande :

```
UPDATE guacamole_user
SET disabled = 1/0 #pour désactiver une connexion, 0 pour l'activer
WHERE entity_id = 21; #numéro de l'entity_id
```

Ensuite, selon l'état que je lui ai attribué, j'essaie de me connecter au serveur.

Avec disabled = 1

Ce compte utilisateur n'est pas valide pour le moment.



The image shows the Apache Guacamole login interface. At the top is the Apache Guacamole logo, which consists of a stylized green and yellow bird-like shape inside a black circle. Below the logo, the text "APACHE GUACAMOLE" is displayed in a bold, black, sans-serif font. Underneath the text are two input fields: the first one contains the username "montech", and the second one is labeled "Mot de passe" (Password). At the bottom of the form is a dark grey button with the text "Se connecter" (Log in) in white.

Avec disabled = 0



Je peux désormais me connecter.

Pour appliquer la solution B, on influence également la base de données sauf que cette fois-ci, on va modifier les champs `acces_window_start`, `acces_window_end`, `valid_from` et `valid_until`.

Afin que cela fonctionne, il faut que paramètre la date et l'heure dans le bon format qui est AAAA/MM/JJ et HH:MM:SS.

La commande a utilisé est :

```
UPDATE guacamole_user  
SET
```

```
    access_window_start = '2024-02-14',  
    access_window_end = '2024-02-14',  
    valid_from = 15:15:15',  
    valid_until = '18:00:00'
```

```
WHERE entity_id = 21;
```

Je vais maintenant aller vérifier que ces paramètres ont bien été pris en compte par guacamole.

Nous pouvons voir que l'heure et la date ont bien été définies comme on le souhaite.

Je peux à présent me connecter pendant cette plage horaire. Une fois qu'elle aura expiré, je ne pourrai plus m'authentifier.

Le problème lié à cette solution, c'est qu'elle n'est pas pratique par rapport à une solution déjà établie comme l'api de guacamole. Le risque d'injection est grandement augmenté, de plus, influencer directement la base de données n'est pas la bonne approche, car nous pouvons oublier des paramètres nécessaires au bon fonctionnement de guacamole. Mais également, la difficulté à créer une page interactive avec des boutons pour effectuer des commandes vers la base de données.

Identifiant: montech
Mot de passe: *****
Répéter mot de passe:

PROFIL

Nom: Technicien
Adresse Mail: montech@poitiers.fr
Organisation:
Rôle: Technicien

RESTRICTIONS DE COMPTE

Connexion désactivée: ☐
Mot de passe expiré: ☐
Autoriser l'accès après: 15 : 15 : 15
Ne pas autoriser l'accès après: 18 : 00 : 00
Activer le compte après: 14 / 02 / 2024
Désactiver le compte après: 14 / 02 / 2024
Fuseau horaire utilisateur: Europe Paris

II-Solution numéro 2. Création d'une application WEB à l'aide de l'API Guacamole

Pour commencer cette solution, j'ai dû tout d'abord me documenter sur l'API de guacamole pour comprendre son fonctionnement.

Je n'étais pas à l'aise avec les requêtes GET, PUT, PATCH, POST, DELETE. Donc j'ai pris du temps avant de comprendre le fonctionnement de l'API.

Je me suis ensuite renseigné auprès de l'équipe informatique pour savoir comment je pouvais effectuer ces requêtes afin de faire des tests. On m'a alors conseillé une extension de Firefox appelé RESTClient.



Voici comment se présente RESTClient.

Nous avons le choix de la méthode :

- **GET** : Cette requête est utilisée pour récupérer des données à partir d'une ressource spécifiée. Elle est utilisée lorsqu'on souhaite simplement récupérer des informations du serveur sans effectuer de modifications sur la ressource.
- **PUT** : Cette requête est utilisée pour mettre à jour une ressource existante sur le serveur. On fournit les nouvelles données complètes de la ressource dans le corps de la requête, et elle remplace entièrement la ressource existante.
- **PATCH** : Cette requête est similaire à PUT, mais au lieu de remplacer entièrement la ressource, elle est utilisée pour apporter des modifications partielles à une ressource existante. On fournit uniquement les données à modifier dans le corps de la requête.
- **POST** : Cette requête est utilisée pour envoyer des données au serveur pour créer une nouvelle ressource. Contrairement à PUT, qui est utilisé pour mettre à jour une ressource existante, POST est utilisé pour créer une nouvelle ressource sur le serveur.
- **DELETE** : Cette requête est utilisée pour supprimer une ressource spécifiée sur le serveur.

On a également le **corps** de requête :

Le corps (body) dans une requête HTTP contient les données envoyées au serveur dans le cadre de la requête. Cela s'applique principalement aux requêtes telles que POST, PUT et PATCH, où des données supplémentaires doivent être envoyées au serveur pour créer, mettre à jour ou modifier une ressource.

L'URL :

L'URL (Uniform Resource Locator) dans une requête HTTP (Hypertext Transfer Protocol) sert à identifier la ressource sur le serveur avec laquelle vous souhaitez interagir. Il fournit une adresse spécifique qui permet au serveur de localiser la ressource demandée.

Les en-têtes :

Les en-têtes (headers) dans une requête HTTP (Hypertext Transfer Protocol) servent à fournir des informations supplémentaires sur la requête elle-même et sur les données qui y sont associées. Ils fournissent des métadonnées importantes pour le serveur afin de comprendre et de traiter correctement la requête. Voici un des rôles clés des en-têtes dans une requête HTTP que je vais utiliser pour pouvoir échanger avec guacamole :

Type de contenu (Content-Type) : L'en-tête Content-Type indique le type de données envoyées dans le corps de la requête. Par exemple, si le contenu est JSON, l'en-tête Content-Type serait `application/json`. Il y a également Content-Type : `x-www-form-urlencoded` qui est utilisé pour spécifier le format des données envoyées dans le corps d'une requête HTTP lorsqu'un formulaire HTML est soumis. Dans ce format, les données sont encodées sous forme de paires clé-valeur, où les valeurs sont encodées en utilisant la convention "percent-encoding" (aussi connue sous le nom d'encodage URL).

Nous pouvons aussi voir la réponse aux requêtes, afin de savoir si le serveur a bien reçu et accepté les requêtes qu'on lui fait.

1^{re} - Partie. Générer des requêtes vers le serveur

Dans cette première partie de mon travail de recherche sur le fonctionnement de l'API, je vais commencer par récupérer les requêtes intéressantes qui vont me permettre de modifier les paramètres des utilisateurs. Mais avant de pouvoir utiliser ces requêtes, il faut que je m'authentifie auprès du serveur, afin d'avoir la permission d'interagir avec lui.

Comme documentation de l'API, j'ai utilisé un répertoire GIT d'un utilisateur qui a référencé la liste des commandes. [Documentation](#).

Avant de commencer, rappelons-nous que nous avons deux options pour réaliser ce projet. La première étant de créer un bouton avec un état on/off qui active ou désactive une connexion. La deuxième option, elle, est de paramétrer une plage horaire de l'utilisateur, afin qu'il puisse s'authentifier que pendant la période autorisée.

Que ce soit pour la première ou deuxième option, la première commande sera la même. Il va falloir qu'on commence tout d'abord par générer un Token en s'authentifiant auprès du serveur.

Générer un jeton d'authentification pour un utilisateur :

Method : POST

Headers : Content-Type:application/x-www-form-urlencoded

URL : <http://ipserveur/guacamole/api/tokens>

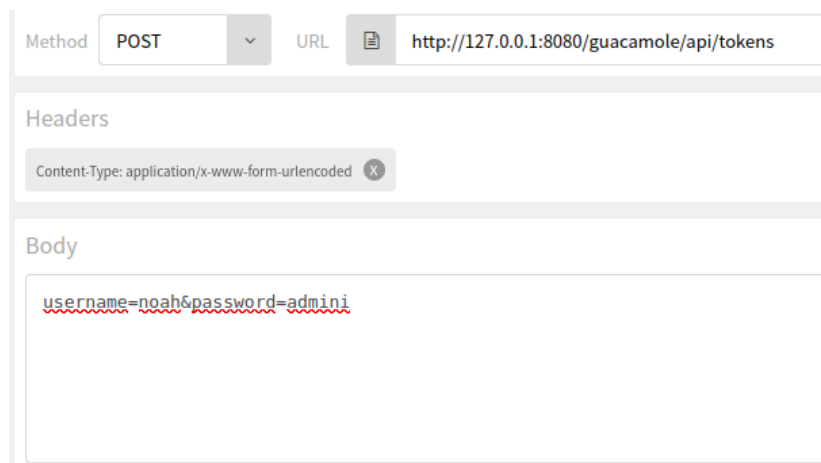
Body : username=exemple&password=motdepasseutilisateur

Utilisation de la méthode POST pour créer une ressource, le token en occurrence.

Utilisation de l'en-tête Content-type:application/x-www-form-urlencoded pour indiquer quel type de données on va lui envoyer. x-www-form-urlencoded est utilisé pour spécifier le format des données envoyées dans le corps d'une requête HTTP.

URL : On spécifie quelle ressource est demandée.

Le body va être les informations qui vont nous permettre de s'authentifier et de générer le Token.



The screenshot shows a REST client interface with the following fields:

- Method:** POST
- URL:** http://127.0.0.1:8080/guacamole/api/tokens
- Headers:** Content-Type: application/x-www-form-urlencoded
- Body:** username=noah&password=admini

Voici la réponse du serveur :

```
{
  "authToken": "B89706F078DA60A29A0E42D326E77562C1BE7D353A9DB57610C57CC193A65561",
  "username": "noah",
  "dataSource": "mysql",
  "availableDataSources": ["mysql", "mysql-shared"]
}
```

Nous pouvons observer que le serveur a généré un token et nous la retourné.

On voit également quel utilisateur est lié à ce token.

L'origine de la source et la liste des sources de données disponibles pour l'utilisateur cité.

Ce token va nous être utile pour la suite de nos tests, donc il va falloir le stocker sur un bloc-note ou autre support.

Solution A : Créer un bouton qui active ou désactive l'autorisation d'un utilisateur à s'authentifier.

Voici la liste des étapes et des commandes qui vont permettre d'activer ou de désactiver une connexion grâce aux requêtes.

Voir les paramètres actuels d'un utilisateur :

Method : GET

URL

<http://ipserveur/guacamole/api/session/data/mysql/users/nomutilisateur/?token=<token>>

```
{
  "username": "montech",
  "attributes": {
    "guac-email-address": "montech@poitiers.fr",
    "guac-organizational-role": "Technicien",
    "guac-full-name": "Technicien",
    "expired": null,
    "timezone": "Europe/Paris",
    "access-window-start": null,
    "guac-organization": null,
    "access-window-end": null,
    "disabled": "true",
    "valid-until": null,
    "valid-from": null
  },
  "lastActive": 1707903203000
}
```

Voici la réponse de la requête, nous avons les informations liées à l'utilisateur. L'attribut disabled étant sur "true" cela signifie que la connexion à ce compte est désactivée.

Ce compte utilisateur n'est pas valide pour le moment.

Suite à cela, voyons maintenant comment autoriser la connexion de cet utilisateur.



APACHE GUACAMOLE

Se connecter

Autoriser un utilisateur à se connecter à guacamole :

Method : PUT

Header : Content-Type:application/json;charset=utf-8

URL:

<http://ipserveur/guacamole/api/session/data/mysql/users/nomutilisateur?token=<token>>

Body :

```
{
  "attributes": {
    "disabled": "false"
  }
}
```

Utilisation de la méthode PUT pour mettre à jour les paramètres de l'utilisateur cible.

L'en-tête utilisé nous permet d'encoder le corps sous un format JSON.

L'URL pour spécifier la ressource demandée. Il faut, en plus de l'URL, rajouter le paramètre “?token=”, qui nous permet de spécifier le token de l'utilisateur qu'on a généré précédemment, afin d'avoir la permission d'effectuer ces requêtes au serveur.

Le Body qui nous permet de modifier les paramètres au format JSON de l'utilisateur défini. On modifie dans ce paramètre son autorisation à se connecter qui est stocké dans la base donnée. On modifie le tableau [attributes] et la colonne =>[disabled].

Si nous consultons maintenant les permissions de l'utilisateur, le champ “disabled” est sur “null”

Réponse du serveur :

```
"disabled": null,
```



Nous pouvons nous authentifier avec l'utilisateur Montech et son mot de passe.

Désactiver un utilisateur à se connecter à guacamole :

Method : PUT

Header : Content-Type:application/json;charset=utf-8

URL:

<http://ipserveur/guacamole/api/session/data/mysql/users/nomutilisateur?token=<token>>

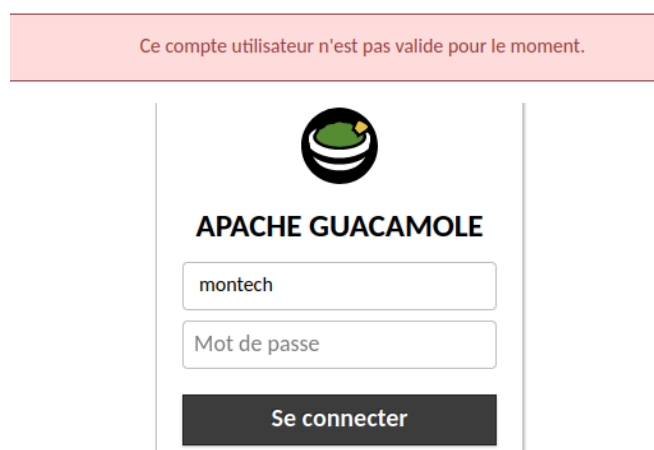
Body :

```
{
  "attributes": {
    "disabled": "true"
  }
}
```

Si on consulte les permissions de l'utilisateur, le champ "disabled" est sur "true"

Réponse du serveur :

"disabled": "true",



Voici les requêtes pour la solution A, qui nous permettent d'activer ou désactiver une connexion.

Pour conclure sur cette solution, nous avons besoin seulement de générer un token avec les identifiants d'un utilisateur administrateur. Consulter au besoin les paramètres actuels de l'utilisateur, et enfin, activer ou désactiver sa possibilité de se connecter.

Solution B : Créer un formulaire dans lequel on peut définir une plage horaire pendant laquelle l'utilisateur peut s'authentifier.

Voici la liste des étapes et des commandes qui vont permettre de paramétrer des plages horaires grâce aux requêtes.

Générer un jeton d'authentification pour un utilisateur :

Method : POST

Headers : Content-Type:application/x-www-form-urlencoded

URL : <http://ipserveur/guacamole/api/tokens>

Body : username=exemple&password=motdepasseutilisateur

Voir les paramètres actuels d'un utilisateur :

Method : GET

URL

<http://ipserveur/guacamole/api/session/data/mysql/users/nomutilisateur?token=<token>>

Cette requête peut nous servir à savoir si une plage horaire est déjà définie.

```
{
  "username": "montech",
  "attributes": {
    "guac-email-address": "montech@poitiers.fr",
    "guac-organizational-role": "Technicien",
    "guac-full-name": "Technicien",
    "expired": null,
    "timezone": "Europe/Paris",
    "access-window-start": "17:01:00",
    "guac-organization": null,
    "access-window-end": "17:02:00",
    "disabled": null,
    "valid-until": "2025-03-17",
    "valid-from": "2026-03-17"
  },
  "access-window-start": "17:01:00",
  "access-window-end": "17:02:00",
  "valid-until": "2025-03-17",
  "valid-from": "2026-03-17"
}
```

“access-window-start” Heure à partir à laquelle il peut s'authentifier
“access-window-end” Heure jusqu'à laquelle il peut s'authentifier
“valid-until” La date d'expiration
“valid-from” La date à partir de laquelle il peut s'authentifier

Paramétrer la plage horaire des utilisateurs :

Method : PUT

Header : application/json;charset=utf-8

URL

<http://ipserveur/guacamole/api/session/data/mysql/users/nomutilisateur?token=<token>> :

Body :

```
{
  "attributes": {
    "access-window-end": "10:00:00",
    "access-window-start": "09:00:00",
    "valid-from": "2024-02-19",
    "valid-until": "2024-02-19"
  }
}
```

Si on lui redemande les paramètres actuels de l'utilisateur montech, on peut voir que l'heure et la date ont bien été pris en compte

```
{
  "username": "montech",
  "attributes": {
    "guac-email-address": "montech@poitiers.fr",
    "guac-organizational-role": "Technicien",
    "guac-full-name": "Technicien",
    "expired": null,
    "timezone": "Europe/Paris",
    "access-window-start": "09:00:00",
    "guac-organization": null,
    "access-window-end": "10:00:00",
    "disabled": null,
    "valid-until": "2024-02-19",
    "valid-from": "2024-02-19"
  },
  "lastActive": 1707903203000
}
```

CONNEXIONS RÉCENTES



SSH-SRV1

TOUTES LES CONNEXIONS

SSH-SRV1

SSH-SRV2


Filtre

Nous pouvons nous authentifier avec l'utilisateur Montech et son mot de passe.

Après date d'expiration :

À partir de 10H, comme on lui a spécifié précédemment, on ne peut plus désormais se connecter.

L'accès à ce compte n'est pas autorisé pour le moment. Veuillez réessayer plus tard.



APACHE GUACAMOLE

montech

Mot de passe

Se connecter

Voici les requêtes pour la solution B, qui nous permettent d'autoriser l'authentification d'un utilisateur selon une plage horaire définie.

Pour conclure sur cette solution, nous avons besoin seulement de générer un token avec les identifiants d'un utilisateur administrateur. Consulter au besoin les paramètres actuels de l'utilisateur, et enfin, lui paramétrer sa plage horaire.

Conclusion :

Ces deux solutions qui peuvent permettre toutes les deux, différentes méthodes de régularisation d'accès, fonctionnent. J'ai donc proposé l'une de ces deux solutions à mon tuteur qui a choisi l'option de paramétrer une plage horaire, car elle apporte une fonctionnalité en plus que la première solution. Cette fonctionnalité permet de désactiver automatiquement, sans action de l'utilisateur au moment précis, la possibilité à un utilisateur de s'authentifier. En effet, parce que grâce à la date d'expiration à paramétrer au début, cela enlève une action à effectuer pour l'utilisateur plus tard. Ainsi, on évite les oublis de désactivation en fin d'intervention.

Structuration et application de la solution 2. B

Création d'un formulaire.

Liste des pré-requis

- Serveur Apache2
- Serveur Guacamole
- Extension Firefox RESTClient
- Visual Studio Code

Mon projet sera hébergé sur mon propre poste, sur un serveur Apache2.

Tables des matières

-
-
-
-
-
-
-

1 - Création du formulaire Version 1.

Comme précédemment expliqué, je vais créer un formulaire en HTML qui va me permettre de paramétrer les plages horaires des utilisateurs. J'utilise donc mon serveur Apache qui va venir héberger mon formulaire appelé index.php.

J'utilise comme extension de fichier .php, car nous allons avoir besoin de PHP pour développer mon outil.

Pour le créer, je vais créer un fichier avec Visual Studio Code dans le répertoire var/www/html/.

1^{re} étape. Initialisation de CURL avec les différentes méthodes :

Afin de pouvoir envoyer des requêtes depuis mon formulaire vers le serveur, je vais utiliser la bibliothèque CURL.

```
<?php
//-----

//paramètre de la fonction get
function get(string $url,array $headers=array()) {
    $h = curl_init($url);
    curl_setopt($h, CURLOPT_TIMEOUT, 10);
    curl_setopt($h, CURLOPT_URL, $url);
    curl_setopt($h, CURLOPT_HEADER, 0);
    curl_setopt($h, CURLOPT_POST, 0);
    curl_setopt($h, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($h, CURLOPT_HTTPHEADER, $headers);
    curl_setopt($h, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($h, CURLOPT_SSL_VERIFYHOST, 0);
    curl_setopt($h, CURLOPT_SSL_VERIFYSTATUS, 0);
    curl_setopt($h, CURLOPT_FOLLOWLOCATION, 1);
    //curl_setopt($h, CURLOPT_PROXY, $this->proxy_url);
    //curl_setopt($h, CURLOPT_PROXYPORT, $this->proxy_port);
    $reponse = curl_exec($h);
    if ($reponse) {
        $reponse = json_decode($reponse, true);
    } else {
        $this->error = curl_error($h);
    }
    curl_close($h);
    return $reponse;
}

//-----

//paramètre de la fonction post
function post(string $url, array $headers, string $body) {
    $h = curl_init($url);
    curl_setopt($h, CURLOPT_TIMEOUT, 10);
    curl_setopt($h, CURLOPT_URL, $url);
    curl_setopt($h, CURLOPT_HEADER, 0);
    curl_setopt($h, CURLOPT_POST, 1);
    curl_setopt($h, CURLOPT_CUSTOMREQUEST, "POST");
    curl_setopt($h, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($h, CURLOPT_HTTPHEADER, $headers);
    curl_setopt($h, CURLOPT_POSTFIELDS, $body);
    curl_setopt($h, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($h, CURLOPT_SSL_VERIFYHOST, 0);
    curl_setopt($h, CURLOPT_SSL_VERIFYSTATUS, 0);
    curl_setopt($h, CURLOPT_FOLLOWLOCATION, 1);
    //curl_setopt($h, CURLOPT_PROXY, $this->proxy_url);
    //curl_setopt($h, CURLOPT_PROXYPORT, $this->proxy_port);

    $reponse = curl_exec($h);

    if ($reponse) {
        $reponse = json_decode($reponse, true);
    } else {
        $this->error = curl_error($h);
    }
    curl_close($h);
    return $reponse;
}
```

Ces différents paramètres ont été partagés par le développeur du service informatique.

Pour la fonction GET

1. Initialise une session cURL avec `curl_init`.
2. Définit un délai d'attente de 10 secondes avec `curl_setopt` pour éviter les timeouts.
3. Définit l'URL de la requête avec `curl_setopt`.
4. Configure la requête pour ne pas inclure les en-têtes de réponse avec `curl_setopt`.
5. Configure la requête pour ne pas être une requête POST avec `curl_setopt`.
6. Configure la requête pour récupérer le résultat en tant que chaîne de caractères avec `curl_setopt`.
7. Ajoute les en-têtes spécifiés avec `curl_setopt`.
8. Désactive la vérification SSL avec `curl_setopt` (ce qui peut être un risque pour la sécurité dans certains cas, donc utilisez-le avec prudence).
9. Exécute la requête avec `curl_exec`.
10. Si la requête réussit, décode la réponse JSON en tableau associatif avec `json_decode`.
11. Sinon, stocke l'erreur dans la propriété `error` de l'objet (qui semble être une référence à une classe, mais je ne vois pas la définition de cette classe dans le code que vous avez fourni).
12. Ferme la session cURL avec `curl_close`.
13. Retourne la réponse de la requête.

La fonction POST est similaire à GET, mais elle prend en plus un corps de requête, appelé body.

1. Configure la requête pour être une requête POST avec `curl_setopt`.
2. Définit le corps de la requête avec `curl_setopt`.

2^e étape. Construction d'une requête pour générer un token :
username = l'utilisateur admin et password = son mot de passe

```
$r=POST('http://127.0.0.1:8080/guacamole/api/tokens',  
array('Content-Type:application/x-www-form-urlencoded'),  
'username=noah&password=admini');
```

Création du variable appelé \$r qui sera la réponse au POST de la requête.

On va ensuite extraire le token du tableau \$r.

```
$token=$r['authToken'];
```

Pour vérifier que nous avons bien réussi à extraire le token et que nous l'avons stocker, il faut l'afficher

```
var_dump($token);
```

```
string(64) "DCA7FE752AC43046DF91BEC0E76A98E4E1B168DEEAC8200273687FEF4785EE48"
```

Le token à bel est bien était stocké dans \$token

3^e étape. Réutilisation du token :

Maintenant que nous avons notre token, il va falloir l'utiliser dans nos prochaines requêtes afin d'avoir les permissions d'interagir avec le serveur.

Exemple de requête avec le token utilisé :

```
$reponse=GET('http://127.0.0.1:8080/guacamole/api/session/data/mysql/users/montech/?token='.urlencode($token));
```

4^e étape. Création du formulaire :

Pour créer le formulaire, je vais simplement utiliser du html avec la balise <form> en utilisant la méthode post pour envoyer le formulaire.

Pour les noms des différents labels, je vais utiliser les mêmes termes utilisés pour envoyer des requêtes au serveur. Ces noms vont nous servir lors de l'envoi des données à Guacamole.

```
<form method="post" action="">  
  <label for="access-window-start">Autoriser l'accès après:</label>  
  <input type="time" name="access-window-start" id="access-window-start" required>  
  <label for="access-window-end">Ne pas autoriser l'accès après:</label>  
  <input type="time" name="access-window-end" id="access-window-end" required>  
  <label for="valid-from">Activer le compte après:</label>  
  <input type="date" name="valid-from" id="valid-from" required>  
  <label for="valid-until">Désactiver le compte après:</label>  
  <input type="date" name="valid-until" id="valid-until" required>  
  <input type="submit" value="Envoyer">  
</form>
```

5^e étape. Création de la requête :

Une étape très importante, qui est de récupérer les données du formulaire et de les transformer en variable.

Pour cela, je vais utiliser une condition qui va s'actionner, si elle reçoit une requête post d'elle-même. Lorsqu'elle s'active, elle récupère les données du formulaire et en fait des variables.

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {  
    // Récupérer les valeurs du formulaire  
    $accessWindowStart = ($_POST['access-window-start'].':00');  
    $accessWindowEnd = ($_POST['access-window-end'].':00');  
    $validFrom = $_POST['valid-from'];  
    $validUntil = $_POST['valid-until'];  
}
```

Nous allons créer un tableau appelé \$data à partir des différentes variables récupérées du formulaire. Ce tableau va être le corps de la requête, il faut donc l'encoder en JSON.

```
// Construire le tableau $data  
$data = array(  
    "attributes" => array(  
        "access-window-end" => $accessWindowEnd,  
        "access-window-start" => $accessWindowStart,  
        "valid-from" => $validFrom,  
        "valid-until" => $validUntil  
    )  
);  
  
// Convertir le tableau en format JSON  
$jsonData = json_encode($data);
```


Après cela, on peut créer la requête qui nous sert à paramétrer la plage horaire de l'utilisateur montech.


```
// Appeler la fonction put avec les données du formulaire  
$url =  
'http://127.0.0.1:8080/guacamole/api/session/data/mysql/users/montech?token='  
urlencode($token);  
$headers = array('Content-Type: application/json;charset=utf-8');  
  
$response = put($url, $headers, $jsonData);  
  
// Traitement de la réponse  
//var_dump($response);  
}
```

À partir de ce moment-là, j'ai la possibilité avec un formulaire de paramétrer la plage horaire de l'utilisateur "montech".

Autoriser l'accès après: Test du fonctionnement du formulaire

Ne pas autoriser l'accès après:

Activer le compte après: 

Désactiver le compte après: 


RESTRICTIONS DE COMPTE


Connexion désactivée: ☐



Mot de passe expiré: ☐

Autoriser l'accès après:

Ne pas autoriser l'accès après:

Activer le compte après: 

Désactiver le compte après: 

Fuseau horaire utilisateur:  

Valeur, consulté dans Guacamole pour l'utilisateur montech.

En résumé de ce premier test, nous sommes maintenant capables de définir la plage horaire spécifique pour l'utilisateur "montech". Cependant, nous devons trouver une méthode pour configurer les plages horaires de chaque utilisateur individuellement.

Initialement, j'ai commencé à créer un formulaire manuellement pour chaque utilisateur, mais cette approche n'est pas pratique lorsque nous avons un grand nombre d'utilisateurs. De plus, elle nécessite une intervention humaine dans l'application web à chaque fois qu'une mise des utilisateurs est effectuée dans Guacamole.

2 - Formulaire Version 2.

Dans cette Version 2, nous devons pouvoir avoir un formulaire par utilisateur, généré automatiquement selon la liste des utilisateurs de Guacamole.

Paramètres pour testforeach

Activer le compte après: 

Désactiver le compte après: 

J'ai ajouté du CSS, afin d'avoir au minimum un formulaire plus lisible.

1^{re} étape. Récupération de la liste des utilisateurs

Au chargement de la page, l'application web va interroger le serveur sur sa liste d'utilisateurs.

Voici comment j'ai procédé :

```
userData= GET('http://localhost:8080/guacamole/api/session/data/mysql/users?token='  
. urlencode($token));
```

J'ai fait une boucle foreach qui pour le nombre d'utilisateurs de \$userData va s'exécuter. Dans cette boucle, on va remplir la variable \$intervenant à chaque itération de la boucle les données que nous avons récupérées précédemment dans la requête userData.

```
// Parcourir chaque utilisateur et extraire le nom d'utilisateur  
foreach ($userData as $intervenant) {
```

Nous avons maintenant un tableau avec la liste de tous les utilisateurs de Guacamole.

Si nous souhaitons vérifier le contenu du tableau, nous devons faire une autre boucle foreach, similaire à celle-ci, qui va créer une liste avec les noms des utilisateurs.

Pour ce faire, on va initialiser un tableau vide pour stocker les utilisateurs. Ensuite pour le nombre d'utilisateurs récupéré dans \$userData on va extraire la donnée 'username' et l'enregistrer dans le tableau \$usernames.

```
$usernames = array();  
// Parcourir chaque utilisateur et extraire le nom d'utilisateur  
foreach ($userData as $user) {  
    $usernames[] = $user['username'];  
}  
print_r($usernames); // Afficher la liste $usernames
```

Affichons le tableau avec print_r (\$usernames);

```
Array ( [0] => Technicien [1] => tech-test [2] => testforeach [3] => Intervenant-serveur [4] => Tech-guacamole [5] =>  
tectest [6] => Technicien [7] => test123 [8] => montech [9] => noah [10] => Utilisateur [11] => intervenant [12] =>  
Intertech )
```

Ce tableau nous sert seulement à vérifier, si on récupère bien les noms des utilisateurs.

2^e étape. Génération des formulaires

Reprenons notre boucle foreach.

```
// Parcourir chaque utilisateur et extraire le nom d'utilisateur  
foreach ($userData as $intervenant) {
```

Cette boucle va être exécutée pour chaque utilisateur et à chaque itération la variable \$intervenant sera égale au nom d'un utilisateur. Donc, nous allons utiliser cette variable pour nos différents paramètres.

Le contenu à l'intérieur de la boucle :

```
<form id="tableau" method="post" action="">
  <h1>Paramètres pour <?php echo $intervenant['username'] ?></h1>
  <label for="valid-from">Activer le compte après:</label>
  <input type="date" name="valid-from" id="valid-from"required>
  <input type="time" name="access-window-start" id="access-window-start"required>
  <label for="valid-from">Désactiver le compte après:</label>
  <input type="date" name="valid-until" id="valid-until"required>
  <input type="time" name="access-window-end" id="access-window-end">
    <input type="hidden" value="<?php echo $intervenant['username'] ?>"
name="intervenant">
    <input type="submit" name="submit" value="Envoyer">
</form>
<?php>
}
?>
```

Grâce à la variable `$intervenant` on va pouvoir afficher chaque formulaire accompagné du nom de l'utilisateur associé.

De plus, à la fin du formulaire, il y a la ligne `<input type="submit" name="submit" value="Envoyer">` qui va permettre de définir à la requête quel est l'utilisateur ciblé.

Nous devons modifier la condition qui envoie les données vers Guacamole lorsqu'elle reçoit une requête POST. Il faut ajouter la variable `$intervenant` qui stockera le nom de l'utilisateur envoyé dans le formulaire par `<input type="hidden" value="<?php echo $intervenant['username'] ?>" name="intervenant">`.

On doit également modifier l'URL, afin que l'utilisateur ne soit plus "montech", il faut le remplacer par `.$intervenant`.

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    ... //parametres precedents
    $intervenant = $_POST['intervenant'];

    ... //autre parametres
    $jsonData = json_encode($data);

    $url = 'http://127.0.0.1:8080/guacamole/api/session/data/mysql/users/' .
    $intervenant . '?token=' . urlencode($token);
    ... //autre parametres
}

?>
```

Paramètres pour testforeach

Activer le compte après:

Désactiver le compte après:

Envoyer

Paramètres pour montech

Activer le compte après:

Désactiver le compte après:

Envoyer

Paramètres pour intervenant

Activer le compte après:

Désactiver le compte après:

Envoyer

Pour résumer, les formulaires sont désormais générés automatiquement grâce à la requête qui interroge la liste des utilisateurs, ainsi que la boucle foreach. Ils sont nominatifs et propres à chacun de leurs utilisateurs. Les données envoyées d'un formulaire appartenant à l'utilisateur A ne peuvent pas modifier les paramètres appartenant à l'utilisateur B.

J'ai ajouté à cela du CSS, pour le faire correspondre à la charte graphique du CHL.



Gestionnaire des connexions

Paramètres pour Technicien

Activer le compte après:

Désactiver le compte après:

Envoyer

Paramètres pour tech-test

Activer le compte après:

Désactiver le compte après:

Envoyer

Paramètres pour testforeach

Activer le compte après:

Désactiver le compte après:

Envoyer

Paramètres pour Intervenant-serveur

Activer le compte après:

Désactiver le compte après:

Envoyer

Paramètres pour Tech-guacamole

Activer le compte après:

Désactiver le compte après:

Envoyer

Paramètres pour tecttest

Activer le compte après:

Désactiver le compte après:

Envoyer

Le point.

Que permet de faire l'application web ?

- Dans cette version 2 de cette application web, nous avons des formulaires par utilisateur qui se génèrent au chargement de la page. Ces formulaires sont générés dynamiquement, selon la base des utilisateurs de Guacamole.
Grâce à ces formulaires, nous pouvons paramétrer la plage horaire de tous les utilisateurs de Guacamole.

Ce que j'ai constaté.

Dans cette version-là, j'ai observé deux problèmes.

- Le premier étant que les utilisateurs administrateurs apparaissent dans les formulaires affichés.
- Le second est que si un utilisateur renseigne une date d'expiration inférieure à la date d'activation, la permission de connexion de ce compte reste continuellement active.
-

Par conséquent, je vais devoir trouver des solutions à ces différents problèmes.

Ajout de fonctionnalités.

J'ai échangé avec mon tuteur sur le projet, il souhaiterait ajouter des fonctionnalités.

- Afficher les plages horaires actuelles des utilisateurs s'ils en ont une.
- Dans les champs input pour activer le compte, récupérer la date et heure actuelle et les provisionner. Pour les champs input de date expiration, provisionner la date actuelle et ajouter +2 à l'heure actuelle.

Résolution des problèmes

1^{er} Problème. Exclure les administrateurs des formulaires.

Ce problème a été identifié quand j'ai vu le formulaire de l'utilisateur "noah" qui est un utilisateur aux privilèges d'administrateurs.

Pour résoudre ce problème, nous allons devoir interroger les permissions des utilisateurs, afin de savoir s'ils sont administrateurs.

Pour vérifier auprès de Guacamole si un utilisateur est administrateur, on doit effectuer une requête.

Paramètres pour noah

Activer le compte après:

Désactiver le compte après:

Voir les permissions d'un utilisateur :

Method : GET

URL :

<http://ipserveur/guacamole/api/session/data/mysql/users/utilisateur/effectivePermissions?token=<token>>

Une partie de la réponse à la requête dans le cas où, l'utilisateur est administrateur

"systemPermissions": ["ADMINISTER"]

S'il ne l'est pas "systemPermissions": []

Tournons-nous vers la boucle foreach que nous avons créées pour afficher les formulaires

```
foreach ($userData as $intervenant) {  
    ?> <form>  
        ...  
        ...  
    </form>  
  
    <?php  
        }  
    ?>
```

Ajoutons à la boucle, la requête qui nous permet de consulter les permissions d'un utilisateur. Dans l'URL

```
foreach ($userData as $intervenant) {  
    $reponse=get('http://localhost:8080/guacamole/api/session/data/mysql/users/'.$intervenant['username'] . '/effectivePermissions?token=' . urlencode($token));
```

Ensuite, on crée une condition qui va vérifier si l'utilisateur n'est pas administrateur, s'il ne l'est pas, on exécute le formulaire.

```
    if (!in_array('ADMINISTER', $reponse['systemPermissions'])) {  
  
    ?> <form>  
        ...  
        ...  
    </form>  
    <?php  
        }  
    }  
    ?>
```

Grâce à cela, les formulaires des administrateurs sont désormais exclus.

2^e Problème. Date d'expiration inférieure à la date d'accès.

Pour résoudre ce problème qui laisse la possibilité à un utilisateur Guacamole de s'authentifier quand il veut, il faut ajouter une condition qui, avant d'envoyer la requête à Guacamole, teste s'il la date d'expiration n'est pas inférieure à la date d'accès.

Il faut placer cette condition, dans la condition qui s'active, si l'application WEB reçoit une requête POST d'elle-même.

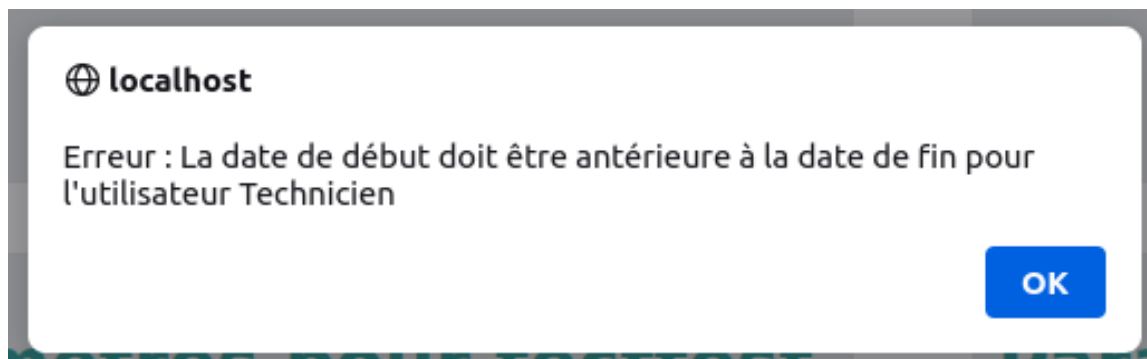
Définition des variables \$start et \$end qui sont les données de dates d'accès et de date d'expiration. Ces données sont converties en heure et en date avec la fonctionnalité 'strtotime'

Puis, on vérifie si la date d'accès est supérieure à la date d'expiration, si elle l'est alors, on affiche un message d'erreur. Sinon, on exécute la suite.

```
$start = strtotime($validFrom . ' ' . $accessWindowStart);
$end = strtotime($validUntil . ' ' . $accessWindowEnd);

if ($start > $end) {
    echo "<script>alert('Erreur : La date de début doit être antérieure à la
date de fin pour l'utilisateur $intervenant');</script>,";
} else {
    ...
    ...
}
);
```

Affichage du message d'erreur s'il la date d'accès est supérieur à la date de fin.



Les problèmes sont maintenant résolus, je vais alors pouvoir me pencher sur l'ajout des fonctionnalités demandées.

Ajout des fonctionnalités

1^{re} Fonctionnalité. Afficher les plages horaires actuelles des utilisateurs.

Pour faire ajouter cette fonctionnalité, je vais modifier mon formulaire afin de lui ajouter du PHP ou j'ajoute la date que j'ai récupérée avec la requête :

Method : GET

URL : [http://ipserveur/guacamole/api/session/data/mysql/users/\\$intervenant/?token=\\$token](http://ipserveur/guacamole/api/session/data/mysql/users/$intervenant/?token=$token)

```
<div>
  <label for="access-window-start">Paramètres actuels: </label>
  <!-- vérifier si la date est nul -->
  <span id="dateactu"> <?php if (!is_null($intervenant['attributes']['valid-from'])) {
    echo date("d/m/Y", strtotime($intervenant['attributes']['valid-from']));
  } else {
    echo 'aucune date';
  }
  ?> </span>
  <span id="dateactu2"> <?php if (!is_null($intervenant['attributes']['access-window-start'])) {
    echo " ", $intervenant['attributes']['access-window-start'];
  } else {
    echo 'aucune heure';
  }
  ?> </span>
</div>
<input type="hidden" value="<?php echo $intervenant['username'] ?>" name="intervenant">
```

Il a été nécessaire de reformuler la date afin qu'elle s'affiche correctement

Cette partie était la partie de la date d'accès, maintenant, il faut faire la même chose pour la date de fin.

```
<div>
  <label for="access-window-until">Paramètres actuels: </label>
  <span id="dateactu"> <?php if (!is_null($intervenant['attributes']['valid-until'])) {
    echo date("d/m/Y", strtotime($intervenant['attributes']['valid-until']));
  } else {
    echo 'aucune date';
  }
  ?> </span>
  <span id="dateactu2"> <?php if (!is_null($intervenant['attributes']['access-window-end'])) {
    echo " ", $intervenant['attributes']['access-window-end'];
  } else {
    echo 'aucune heure';
  }
  ?> </span>
</div>
```

Voici le résultat.

Nous pouvons observer les dernières plages horaires attribuées aux utilisateurs.

Paramètres pour Technicien

Activer le compte après:

jj / mm / aaaa -- : --

Paramètres actuels:

Désactiver le compte après:

jj / mm / aaaa -- : --

Paramètres actuels:

Envoyer

2^e Fonctionnalité. Récupérer l'heure et la date du jour et les provisionner par défaut dans les entrées.

Pour répondre à cette demande, j'ai modifié le formulaire afin que dans la valeur par défaut des champs `<input>`. Afin qu'on écrive à l'intérieur avec l'aide de PHP la date et heure actuel.

```
<input type="time" name="access-window-start" id="access-window-start" value=<?php
$heure = date('H:i');

echo $heure; ?> required>
```

Pour écrire l'heure du futur avec 2H en plus, voici comment j'ai procédé.

```
<input type="time" name="access-window-end" id="access-window-end" value=<?php
$heureFutur = date('H:i', strtotime(' +2 hours'));

echo $heureFutur; ?> required>
```

Voici l'heure à laquelle j'ai chargé la page :

22 févr. 12:17

Résultat affiché dans le formulaire :

Paramètres pour Technicien			
Activer le compte après:	22 / 02 / 2024	12 : 17	
	Paramètres actuels: 23/05/2027 18:57:00		
Désactiver le compte après:	22 / 02 / 2024	14 : 17	
	Paramètres actuels: 24/04/2026 18:58:00		
<button>Envoyer</button>			

La date du jour et l'heure est bien affichée et on ajoute 2h pour la date de fin.

Les dernières plages horaires des utilisateurs sont bien affichées.

Résultat final de cette version 2.



Paramètres pour Technicien	Paramètres pour tech-test	Paramètres pour Intervenant-serveur
Activer le compte après: 22 / 02 / 2024 12 : 17 Paramètres actuels: 23/05/2027 18:57:00 Désactiver le compte après: 22 / 02 / 2024 14 : 17 Paramètres actuels: 24/04/2026 18:58:00 <button>Envoyer</button>	Activer le compte après: 22 / 02 / 2024 12 : 17 Paramètres actuels: 20/02/2024 11:27:00 Désactiver le compte après: 22 / 02 / 2024 14 : 17 Paramètres actuels: 20/02/2024 13:27:00 <button>Envoyer</button>	Activer le compte après: 22 / 02 / 2024 12 : 17 Paramètres actuels: 06/02/2024 16:30:00 Désactiver le compte après: 22 / 02 / 2024 14 : 17 Paramètres actuels: 06/02/2024 18:30:00 <button>Envoyer</button>
Paramètres pour Tech-guacamole	Paramètres pour tecttest	Paramètres pour Technioien
Activer le compte après: 22 / 02 / 2024 12 : 17 Paramètres actuels: 14/02/2024 08:43:00 Désactiver le compte après: 22 / 02 / 2024 14 : 17 Paramètres actuels: 14/02/2024 10:43:00 <button>Envoyer</button>	Activer le compte après: 22 / 02 / 2024 12 : 17 Paramètres actuels: 29/01/2024 16:33:00 Désactiver le compte après: 22 / 02 / 2024 14 : 17 Paramètres actuels: 29/01/2024 18:33:00 <button>Envoyer</button>	Activer le compte après: 22 / 02 / 2024 12 : 17 Paramètres actuels: 13/02/2024 14:29:00 Désactiver le compte après: 22 / 02 / 2024 14 : 17 Paramètres actuels: 13/02/2024 16:29:00 <button>Envoyer</button>

Nous disposons maintenant d'une application web qui affiche les formulaires horaires des utilisateurs de Guacamole, avec leur date d'accès et date de fin déjà provisionnée automatiquement et modifiable, leur dernière plage horaire paramétrée.

Dans la demande qui était de faire un POC d'outil, j'ai rempli la totalité de la mission qui m'avait été confiée.

J'ai présenté cette version à mon tuteur qui était satisfait du résultat. Cependant, j'ai voulu aller plus loin dans une 3^e version.

En effet, l'affichage automatique des formulaires au chargement de la page, peut générer un grand nombre de formulaires selon la liste des utilisateurs de guacamole. J'ai donc voulu ajouter à cette 3^e Version, la possibilité de rechercher à partir d'une liste et en même un champ de recherche, l'utilisateur qu'on souhaite administrer. Je souhaite aussi lister les groupes et pouvoir ensuite les parcourir pour connaître les utilisateurs qui leur appartiennent et afficher leur formulaire.

3 - Formulaire Version 3.

Intégration des nouvelles fonctionnalités.

1^{re} Étape. Création de la liste et du champ de recherche.

Je souhaite intégrer ces deux fonctionnalités en une seule. Mon idée est lorsqu'on appuie sur le champ de recherche, la liste des utilisateurs se déroule, mais qu'on puisse également rechercher un utilisateur en tapant son nom.

Pour réaliser le champ de recherche, je l'ai récupéré du portail d'application du CHL, afin que cela corresponde à la charte graphique.

Voici donc le résultat :



Voici le code HTML utilisé pour générer le champ de recherche.

```
<div id="liste">
  <div id="listUser">
    <div id="align-bandeau">
      <div id="bandeau">
        <input type="text" id="search" list="user-list" placeholder="Rechercher un utilisateur...">
        <span id="search_zon_btn" class="input-group-addon" onclick="btn_recherche()"> <!-- onclick
          <span class="glyphicon glyphicon-search">
        </span>
      </div>
    </div>
  </div>
  <datalist id="user-list">
    <?php
      foreach ($usernames as $username) {
        echo "<option value=\"{$username}\" label=\"{$username}\" selected hidden></option>";
      }
    <?>
  </datalist>
</div>
```

Je voudrais maintenant que lorsqu'on clique dessus, une liste se déroule avec la liste des utilisateurs. Pour faire la liste, j'ai utilisé le tableau \$usernames qu'on avait créé au début. J'utilise ce tableau afin d'exclure les administrateurs du tableau.

```
foreach ($userData as $user) {
    $permission =
    get('http://localhost:8080/guacamole/api/session/data/mysql/users/'
    $user['username'] . '/effectivePermissions?token=' . urlencode($token));
    if (!in_array('ADMINISTER', $permission['systemPermissions'])) {
        $usernames[] = $user['username'];
    }
}
```

J'ajoute dans le div "listUser" de mon champ de recherche une balise <datalist> qui me permet de faire une liste déroulante. Ensuite, j'utilise du PHP pour renseigner les informations de la liste avec la liste des utilisateurs du tableau \$usernames.

Maintenant, pour que la liste affiche un formulaire lorsqu'on sélectionne un utilisateur, il va falloir utiliser du JavaScript qui va écouter les événements sur la page.

```
<script>
    // Récupérer la valeur de la zone de recherche
    document.getElementById("search").addEventListener("change", function() {
        // Récupérer la valeur de la zone de recherche
        var selectedIntervenant = this.value;

        // Masquer tous les formulaires d'intervenants
        var intervenantForms =
document.getElementsByClassName("intervenant-form");
        for (var i = 0; i < intervenantForms.length; i++) {
            intervenantForms[i].style.display = "none";
        }

        // Afficher le formulaire correspondant à l'intervenant sélectionné
        var selectedForm = document.getElementById("form-" +
selectedIntervenant);
        if (selectedForm) {
            selectedForm.style.display = "block";
        }
    });
});
```

Ce script va masquer tous les formulaires qu'ils ne soient pas affichés, il va chercher la valeur qui est présente dans la zone "search" qui est notre champ de recherche. Ensuite, avec la valeur qu'il va récupérer, il va afficher le formulaire de l'intervenant.



Voici le résultat, nous avons un champ de recherche avec une liste déroulante pour parcourir la liste des utilisateurs.

Lorsqu'on sélectionne un utilisateur, son formulaire apparaît.

J'ai trouvé un problème dans cette fonctionnalité, c'est que si on tape le nom d'un utilisateur admin, il nous affiche son formulaire.

Pour résoudre ce problème, j'ai créé une condition qui va vérifier avant de générer un formulaire si un utilisateur est administrateur. S'il est, son formulaire n'est pas généré.

2^e Étape. Création de la liste de groupes parcourables.

Pour réaliser cette idée, je vais commencer par demander à l'application web d'interroger le serveur pour obtenir la liste de tous les groupes.

```
// Récupérer la liste des groupes
$groupData = get('http://localhost:8080/guacamole/api/session/data/mysql/userGroups?token=' . urlencode($token));
$groupnames = array();
foreach ($groupData as $group) {
    $groupnames[] = $group['identifiant'];
}
```

J'ai fait une requête qui va récupérer les permissions de chacun des groupes. Puis je vérifie les permissions des groupes que l'on a enregistré précédemment avec une requête.

Je fais une condition pour vérifier si les groupes ont des permissions administrateurs. S'ils ne le sont pas alors, on peut exécuter le code suivant. Le code suivant va interroger Guacamole sur la liste des utilisateurs appartenant à chaque groupe grâce à la variable `$group[identifiant]`.

Afin de vérifier si l'utilisateur du groupe est administrateur ou non, je vais comparer le précédent tableau avec la liste des administrateurs avec la liste d'utilisateur de chaque groupe. Si un utilisateur admin est trouvé dans la liste de groupe, alors, on le supprime de la liste du tableau `$groupUsersFiltered`. J'ai maintenant grâce à cette méthode la liste des utilisateurs filtrée pour chaque groupe sans administrateurs. Ensuite, je vais afficher le groupe avec son nom et ses utilisateurs lui appartenant avec les balises `<détails>` et `<summary>`.

```
<div id="listGroup">
  <label for="groupe"> Groupes de connexions </label>
  <nav id="group-nav">
    <?php
    $admins = array();
    foreach ($userData as $user) {
        $permissions = get('http://localhost:8080/guacamole/api/session/data/mysql/users/' . $user['username'] . '/effectivePermissions?token=' . urlencode($token));
        if (in_array('ADMINISTER', $permissions['systemPermissions'])) {
            $admins[] = $user['username'];
        }
    }

    foreach ($groupData as $group) {
        $permissionGroup = get('http://localhost:8080/guacamole/api/session/data/mysql/userGroups/' . $group['identifiant'] . '/permissions?token=' . urlencode($token));
        if (in_array('ADMINISTER', $permissionGroup['systemPermissions'])) {
            $groupUsers = get('http://127.0.0.1:8080/guacamole/api/session/data/mysql/userGroups/' . $group['identifiant'] . '/memberUsers?token=' . urlencode($token));
            $groupUsersFiltered = array_diff($groupUsers, $admins);
        }
        <détails>
          <summary id="group"><?php echo $group['identifiant']; ?></summary>
          <ul class="summary-content">
            <?php
            if (!empty($groupUsersFiltered)) {
                foreach ($groupUsersFiltered as $user): ?>
                  <li id="listUser">
                    <button id="listUser" onclick="showUserForm(' . $user['username'] . ')"><?php echo $user; ?></button>
                  </li>
                <?php endforeach;
            } else {
                echo "<li>Aucun utilisateur trouvé</li>";
            }
            ?>
          </ul>
        </détails>
      </?php
    }
  </nav>
</div>
```

J'ai créé un bouton, qui lorsqu'on appuie dessus affiche le formulaire de l'utilisateur. Pour cela, j'ai utilisé du JavaScript.

Si aucun utilisateur n'est trouvé dans un groupe, alors l'application web affiche "Aucun utilisateur trouvé".


```
//script pour les groupes
// Fonction pour afficher le formulaire d'un utilisateur spécifique
function showUserForm(username) {
    // Masquer tous les formulaires
    var userForms = document.querySelectorAll('.intervenant-form');
    userForms.forEach(function(form) {
        form.style.display = 'none';
    });

    // Afficher le formulaire correspondant à l'utilisateur sélectionné
    var selectedUserForm = document.getElementById('form-' + username);
    if (selectedUserForm) {
        selectedUserForm.style.display = 'block';
    }
}
```

Présentation du résultat :

Affichage simple

Groupes de connexions

Tech

montech

testdegroupe

Affichage lorsqu'on clique dans un groupe vide.

Groupes de connexions

Tech

• Aucun utilisateur trouvé

montech

testdegroupe

Affichage lorsqu' groupe à des utilisateurs et affichage d'un formulaire.

The screenshot shows a web application titled "Gestionnaire des connexions" with a logo of a stylized tree. The interface is divided into two main sections. On the left, under the heading "Groupes de connexions", there are three expandable group cards: "Tech", "montech", and "testdegroupe". The "montech" group is currently expanded, showing a list of four users: "Technicien", "test123", "montech", and "Intertech". Above this list is a search bar labeled "Rechercher un utilisateur...". On the right, a modal window titled "Paramètres pour montech" is displayed. It contains two sections: "Activer le compte après :" with a date-time picker set to "23 / 02 / 2024" at "11 : 58", and "Désactiver le compte après :" with a date-time picker set to "23 / 02 / 2024" at "13 : 58". Each section also includes "Paramètres actuels :" with a date-time picker set to "22/02/2024" at "15:45:00". At the bottom of the modal is a green "Envoyer" button.

Le point sur cette version 3.

J'ai rencontré beaucoup de difficultés, concernant ces fonctionnalités, comme les utilisateurs administrateurs à exclure, des utilisateurs qui n'apparaissaient pas, ou encore l'utilisation du JavaScript.

Pour conclure sur cette 3^e version, je suis plutôt satisfait du résultat. J'ai bien réussi à intégrer mes idées dont la recherche d'utilisateurs par groupes ou par champs de recherche. Cependant, cette finalité n'est pas une solution optimale, car il s'agit d'un Proof of Concept, donc plus, une première démonstration de comment cela pourrait être fait.

Conclusion du projet.

J'ai trouvé ce projet intéressant et passionnant. Je suis satisfait d'avoir réussi à répondre à la demande de l'entreprise.

Pour résumer ce projet, il m'était demandé de produire un outil facilitant l'administration des plages horaires des comptes des intervenants de Guacamole, afin que l'outil soit le plus simple possible pour les utilisateurs du CHL.

J'ai alors produit un outil permettant, via, la recherche d'utilisateurs par nom ou par leur appartenance aux groupes de Guacamole, un gestionnaire de plage horaire à l'aide de formulaires générés.

Pour conclure, ce Proof of Concept m'a permis de développer et d'améliorer différentes compétences comme l'utilisation de PHP, l'étude de système et de base de données, l'étude des méthodes de requête HTTP, ou bien l'intégration WEB.

Vous trouverez ci-joint le répertoire avec les codes sources du projet : [RÉPERTOIRE PROJET](#)

