IUT Villetaneuse
Université Sorbonne Paris Nord

Suivi par Mme.Graba

SAE23 - Mettre en place une solution informatique pour l'entreprise

Pendichev Andon
Rigoulet Rayan
Gr4

BUT1 R&T

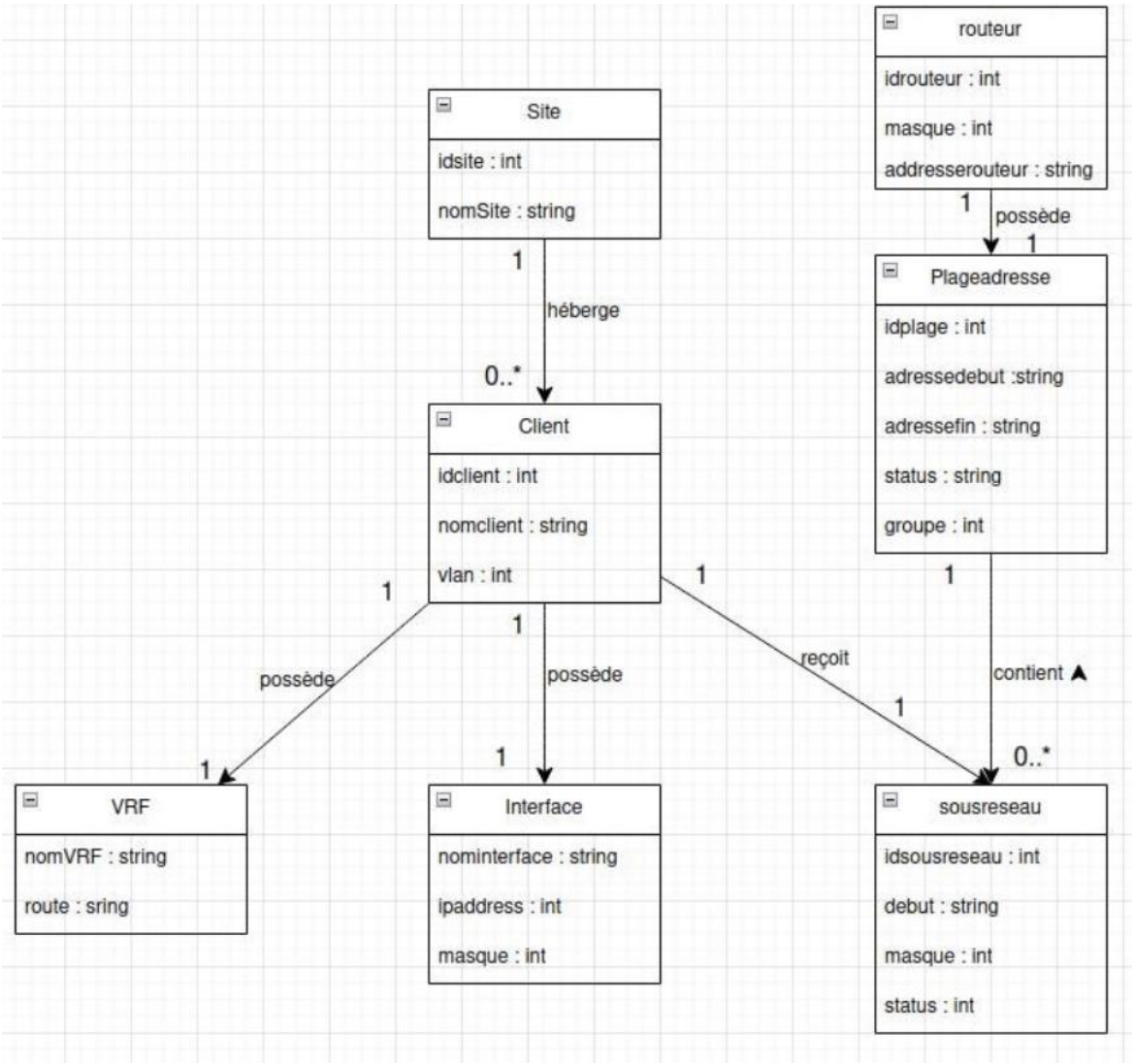
2024/2025 1

Table des Matières

Introduction

Dans le cadre de notre formation en Réseaux et Télécommunications, nous avons été amenés à développer une solution informatique pour une start-up spécialisée dans la commercialisation de connexions Internet. Ce projet vise à concevoir et implémenter un outil de gestion d'adresses IP (IPAM) permettant d'automatiser l'attribution d'adresses publiques aux clients, tout en centralisant les informations réseau dans une base de données.

La solution devra intégrer une interface utilisateur intuitive accessible via un navigateur web, ainsi que des fonctionnalités d'automatisation pour la création de sous-réseaux et la configuration des équipements réseau. Ce projet met en œuvre nos compétences en développement web, gestion de bases de données et algorithmique, tout en répondant aux besoins spécifiques de l'entreprise en matière de gestion décentralisée et d'efficacité opérationnelle.



Conception de la base de données

Nous avons commencé par le diagramme de classes :

Table	Attributs	Clé primaire	Clés étrangères
Site	IdSite (INT), nomSite (VARCHAR)	idSite	___
Client	IdClient (INT), nomClient (VARCHAR), vlan (INT), idSite (INT)	idClient	idSite → Site.idSite
Routeur	Idrouteur (INT) masque (INT) addresserouteur (VARCHAR)		
PlageAdresse	idPlage (INT), adresseDebut (VARCHAR), adresseFin (VARCHAR), statut (VARCHAR), groupe (INT)	idPlage	___
SousReseau	idSousReseau (INT), debut (VARCHAR), masque (INT), statut (VARCHAR), idPlage (INT), idClient (INT)	idSousReseau	idPlage → PlageAdresse.idPlage idClient → Client.idClient
VRF	nomVRF (VARCHAR), routeDistinguisher (VARCHAR), idClient (INT)	nomVRF	idClient → Client.idClient
Interface	nomInterface (VARCHAR), ipAddress (VARCHAR), masque (INT), idClient (INT)	nomInterface	idClient → Client.idClient

Voici notre base de données :

```sql
--  Nettoyage de l'existant
DROP TABLE IF EXISTS client, vrf, interface, sousreseau, site, routeur, plageadresse CASCADE;

-- Création de la table site
CREATE TABLE site (
    id_site SERIAL PRIMARY KEY,
    nom TEXT NOT NULL
);

-- Création de la table sousreseau
CREATE TABLE sousreseau (
    id_sousreseau SERIAL PRIMARY KEY,
    plage_ip CIDR NOT NULL
);

-- Création de la table vrf
CREATE TABLE vrf (
    id_vrf SERIAL PRIMARY KEY,
    nom TEXT NOT NULL,
    rd TEXT NOT NULL
);

-- Création de la table interface
CREATE TABLE interface (
    id_interface SERIAL PRIMARY KEY,
    ip_interface INET NOT NULL
);
```

```sql
-- Création de la table client (avec email)
CREATE TABLE client (
    id_client SERIAL PRIMARY KEY,
    nom TEXT NOT NULL,
    email TEXT NOT NULL,
    id_site INT NOT NULL REFERENCES site(id_site) ON DELETE CASCADE,
    id_sousreseau INT NOT NULL REFERENCES sousreseau(id_sousreseau) ON DELETE CASCADE,
    vlan_id INT NOT NULL,
    rd TEXT NOT NULL,
    id_vrf INT NOT NULL REFERENCES vrf(id_vrf) ON DELETE CASCADE,
    id_interface INT NOT NULL REFERENCES interface(id_interface) ON DELETE CASCADE
);

-- Création de la table routeur
CREATE TABLE routeur (
    id_routeur SERIAL PRIMARY KEY,
    nom TEXT NOT NULL,
    ip INET NOT NULL,
    id_site INT NOT NULL REFERENCES site(id_site) ON DELETE CASCADE
);

-- Création de la table plageadresse
CREATE TABLE plageadresse (
    id_plage SERIAL PRIMARY KEY,
    plage CIDR NOT NULL
);

-- Insertion d'un site par défaut
INSERT INTO site (nom) VALUES ('Radon');
```

Implementation de la base de données

L'objectif de cette première partie était d'implementer la base de données et d'écrire le programme qui permet sa création via une IHM

Par la suite nous avons écrit un programme qui permet d'implémenter cette base de données via une IHM qui permet l'ajout et la suppression d'un client.

Une autre IHM a été crée pour lors de la création d'un nouveau client sur le site, celle-ci permet de voir toutes les informations génerées dans la base de données concernant le client.

Cette création, ainsi que l'adressage (IP et VLAN) est automatisée. Nos clients souhaitant aussi avoir une visualisation simplifiée des clients sur chaque site. Ils nous demandent donc de définir une IHM qui permet de consulter les clients de notre site, de visualiser automatiquement la configuration de chaque client.

Voici notre fonction pour supprimer un client:

supprimer_client.php

```php
<?php
require 'config.php';

if (isset($_GET['id'])) {
    $id = intval($_GET['id']);

    // Supprimer le client
    $stmt = $pdo->prepare("DELETE FROM client WHERE id_client = ?");
    $stmt->execute([$id]);

    // Redirection vers la liste
    header("Location: voir_clients.php");
    exit;
} else {
    echo "Aucun ID client fourni.";
}
?>
```

Ce script PHP établit une connexion à une base de données pSQL en utilisant PDO (PHP Data Objects). Il stocke les informations de connexion (host, nom de la BDD, identifiants) et gère les erreurs potentielles :

config.php

```php
<?php
$host = "aquabdd";
$dbname = "etudiants";
$user = "12402035"; // mon numéro étudiant
$password = "090444260KC";

try {
    $pdo = new PDO("pgsql:host=$host;dbname=$dbname", $user, $password);
} catch (PDOException $e) {
    die("Erreur de connexion : " . $e->getMessage());
}
?>

```

Voici quelques parties clés de notre script principal (fichier ajout_client.php), une IHM qui permet l'ajout et la suppression d'un client ainsi que la création d'un nouveau client sur le site, celle-ci permet de voir toutes les informations générées dans la base de données concernant le client :

ajouter_client.php

```php
<?php
require 'config.php';

// Activer le mode debug
ini_set('display_errors', 1);
error_reporting(E_ALL);

// Suppression d'un client
if (isset($_GET['supprimer'])) {
    $id = intval($_GET['supprimer']);

    $stmt = $pdo->prepare("
        SELECT client.nom AS nom_client, id_vrf, id_interface, id_sousreseau
        FROM client
        WHERE id_client = ?
    ");
    $stmt->execute([$id]);
    $client = $stmt->fetch();

    if ($client) {
        $fichier = 'configs/' . strtolower(str_replace(' ', '_', $client['nom_client'])) . '.txt';
        if (file_exists($fichier)) unlink($fichier);

        $pdo->prepare("DELETE FROM client WHERE id_client = ?")->execute([$id]);
        $pdo->prepare("DELETE FROM vrf WHERE id_vrf = ?")->execute([$client['id_vrf']]);
        $pdo->prepare("DELETE FROM interface WHERE id_interface = ?")->execute([$client['id_interface']]);
        $pdo->prepare("DELETE FROM sousreseau WHERE id_sousreseau = ?")->execute([$client['id_sousreseau']]);

        $_SESSION['message'] = [
            'type' => 'success',
            'text' => 'Client supprimé avec succès!'
        ];
    }

    header("Location: ajouter_client.php");
    exit;
}
```

ajouter_client.php

```php
// Génère les plages IP
function getSubnets() {
    $subnets = [];
    for ($i = 0; $i < 256; $i += 8) {
        $subnets[] = "164.166.1.$i/29";
    }
    return $subnets;
}

$all = getSubnets();
$used_stmt = $pdo->query("SELECT plage_ip FROM sousreseau");
$used_all = $used_stmt->fetchAll(PDO::FETCH_COLUMN);
$used = array_filter($used_all, fn($ip) => str_starts_with($ip, '164.166.1.'));
$free = array_diff($all, $used);
$no_ip = empty($free);

// Ajout d'un client
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $nom = $_POST['nom'];
    $email = $_POST['email'];
    $site_nom = "Randon";

    try {
        $pdo->beginTransaction();

        // Vérification/Création du site
        $stmt = $pdo->prepare("SELECT id_site FROM site WHERE nom = ?");
        $stmt->execute([$site_nom]);
        $id_site = $stmt->fetchColumn();

        if (!$id_site) {
            $pdo->prepare("INSERT INTO site (nom) VALUES (?)")->execute([$site_nom]);
            $id_site = $pdo->lastInsertId();
        }

        // Génération des identifiants
        $res = $pdo->query("SELECT MAX(id_client) FROM client");
        $new_id = ($res->fetchColumn() ?? 0) + 1;
        $vlan_id = $new_id;
        $rd = "65556:$new_id";
```

ajouter_client.php

```php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    try {

        // Attribution plage IP
        $plage = array_values($free)[0];
        $pdo->prepare("INSERT INTO sousreseau (plage_ip) VALUES (?)")->execute([$plage]);
        $id_sousreseau = $pdo->lastInsertId();

        // Calcul IP interface
        $ip_base = explode('/', $plage)[0];
        $ip_parts = explode('.', $ip_base);
        $ip_parts[3] += 1;
        $ip_interface = implode('.', $ip_parts);

        $pdo->prepare("INSERT INTO interface (ip_interface) VALUES (?)")->execute([$ip_interface]);
        $id_interface = $pdo->lastInsertId();

        // Création VRF
        $pdo->prepare("INSERT INTO vrf (nom, rd) VALUES (?, ?)")->execute([$nom, $rd]);
        $id_vrf = $pdo->lastInsertId();

        // Création client
        $sql = "INSERT INTO client (nom, email, id_site, id_sousreseau, vlan_id, rd, id_vrf, id_interface)
                VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
        $pdo->prepare($sql)->execute([$nom, $email, $id_site, $id_sousreseau, $vlan_id, $rd, $id_vrf, $id_interface]);

        // Génération configuration
        $config = "interface Vlan$vlan_id\n";
        $config .= " description Client $nom\n";
        $config .= " ip address $ip_interface 255.255.255.248\n!\n";
        $config .= "ip vrf $nom\n";
        $config .= " rd $rd\n";
        $config .= " route-target export $rd\n";
        $config .= " route-target import $rd\n!\n";
```

```php
        // Sauvegarde fichier
        if (!file_exists("configs")) {
            mkdir("configs", 0777, true);
        }

        $fichier_nom = strtolower(str_replace(' ', '_', $nom)) . ".txt";
        $chemin = "configs/" . $fichier_nom;

        if (file_put_contents($chemin, $config) !== false) {
            $pdo->commit();
            $_SESSION['message'] = [
                'type' => 'success',
                'text' => "Client $nom créé avec succès!",
                'details' => [
                    'Plage IP' => $plage,
                    'Interface IP' => $ip_interface,
                    'Route Distinguisher' => $rd
                ]
            ];
            header("Location: ajouter_client.php");
            exit;
        }
    } catch (Exception $e) {
        $pdo->rollBack();
        $_SESSION['message'] = [
            'type' => 'danger',
            'text' => "Erreur lors de la création du client: " . $e->getMessage()
        ];
    }
}

// Récupération des clients
try {
    $clients = $pdo->query("
        SELECT client.id_client, client.nom AS nom_client, client.email,
               site.nom AS nom_site, vlan_id, plage_ip, ip_interface, rd
        FROM client
        JOIN site ON client.id_site = site.id_site
        JOIN sousreseau ON client.id_sousreseau = sousreseau.id_sousreseau
        JOIN interface ON client.id_interface = interface.id_interface
        ORDER BY client.id_client DESC
    ")->fetchAll(PDO::FETCH_ASSOC);
} catch (PDOException $e) {
    $clients = [];
    $_SESSION['message'] = [
        'type' => 'danger',
        'text' => 'Erreur de base de données: ' . $e->getMessage()
    ];
}
?>
```

La mise en œuvre de notre plateforme de gestion des informations

La présente partie vise à poursuivre la mise en œuvre de notre plateforme de gestion des informations d'un réseau pour une entreprise fictive.

Nous avons automatisé la génération de configurations réseau à partir des données contenues dans la base de données, et enrichi l'interface utilisateur web afin d'offrir des fonctions de recherche, de visualisation détaillée et de documentation automatisée.

Génération automatique d'un fichier de configuration

À partir des informations stockées dans la base de données nous avons créé un script qui génère automatiquement un fichier de configuration pour un routeur PE lors de l'ajout d'un client. Ce fichier contient la création de la sous-interface client à partir de l'interface physique du routeur Cisco, l'adresse IP de l'interface, l'attribution du VLAN etl a configuration de la VRF (nom du client).

```php
// Génération configuration
$config = "interface Vlan$vlan_id\n";
$config .= " description Client $nom\n";
$config .= " ip address $ip_interface 255.255.255.248\n!\n";
$config .= "ip vrf $nom\n";
$config .= " rd $rd\n";
$config .= " route-target export $rd\n";
$config .= " route-target import $rd\n!\n";

// Sauvegarde fichier
if (!file_exists("configs")) {
    mkdir("configs", 0777, true);
}

$fichier_nom = strtolower(str_replace(' ', '_', $nom)) . ".txt";
$chemin = "configs/" . $fichier_nom;

if (file_put_contents($chemin, $config) !== false) {
    $pdo->commit();
    $_SESSION['message'] = [
        'type' => 'success',
        'text' => "Client $nom créé avec succès!",
        'details' => [
            'Plage IP' => $plage,
            'Interface IP' => $ip_interface,
            'Route Distinguisher' => $rd
        ]
    ];
    header("Location: ajouter_client.php");
    exit;
}
```

Amélioration des IHMs

Nous avons aussi fait plusieurs améliorations dans notre plateforme avec le but de la rendre plus efficace, attirant et

fonctionnelle. Exemples de ces améliorations sont l'intégration de la fonction de recherche de client par nom, identifiant ou adresse IP ; le filtrage par site de production ou par statut IP (libre / réservé / alloué) et la vue détaillée pour chaque client, avec sa configuration réseau affichée. Nous allons maintenant vous démontrer les parties du script qui font ces améliorations.

La fonction de recherche de client par nom, identifiant ou adresse IP :

```
// Gestion de la recherche
document.getElementById('search').addEventListener('keyup', function() {
    const filter = this.value.toLowerCase();
    const rows = document.querySelectorAll('#clientTable tbody tr');

    rows.forEach(row => {
        let found = false;
        row.querySelectorAll('td').forEach(td => {
            if (td.textContent.toLowerCase().includes(filter)) found = true;
        });
        row.style.display = found ? '' : 'none';
    });
});
```

```
<!-- Onglet Liste Clients -->
<div id="client-list" class="tab-content">
    <div class="search-box">
        <i class="fas fa-search"></i>
        <input type="text" id="search" class="form-control" placeholder="Rechercher un client...">
    </div>
```

Le filtrage par site de production ou par statut IP (libre / réservé / alloué) :

```php
<!-- Onglet Gestion IP -->
<div id="ip-management" class="tab-content">
    <div class="ip-status available">
        <i class="fas fa-check-circle"></i>
        <div>
            <h4>Statut des plages IP</h4>
            <p><?= count($free) ?> plages disponibles sur <?= count($all) ?></p>
        </div>
    </div>

    <div class="table-responsive">
        <table class="table">
            <thead>
                <tr>
                    <th>Plage IP</th>
                    <th>Statut</th>
                    <th>Client</th>
                    <th>VLAN</th>
                </tr>
            </thead>
            <tbody>
                <?php foreach ($all as $plage): ?>
                    <tr>
                        <td><?= $plage ?></td>
                        <td>
                            <?php if (in_array($plage, $used)): ?>
                                <span class="badge badge-danger"><i class="fas fa-times"></i> Utilisée</span>
                            <?php else: ?>
                                <span class="badge badge-success"><i class="fas fa-check"></i> Libre</span>
                            <?php endif; ?>
                        </td>
```

```php
                        <td>
                            <?php if (in_array($plage, $used)): ?>
                                <?php
                                $client_stmt = $pdo->prepare("
                                    SELECT client.nom
                                    FROM client
                                    JOIN sousreseau ON client.id_sousreseau = sousreseau.id_sousreseau
                                    WHERE sousreseau.plage_ip = ?
                                ");
                                $client_stmt->execute([$plage]);
                                $client_name = $client_stmt->fetchColumn();
                                echo $client_name ? htmlspecialchars($client_name) : 'Inconnu';
                                ?>
                            <?php else: ?>
                                -
                            <?php endif; ?>
                        </td>
                        <td>
                            <?php if (in_array($plage, $used)): ?>
                                <?php
                                $vlan_stmt = $pdo->prepare("
                                    SELECT vlan_id
                                    FROM client
                                    JOIN sousreseau ON client.id_sousreseau = sousreseau.id_sousreseau
                                    WHERE sousreseau.plage_ip = ?
                                ");
                                $vlan_stmt->execute([$plage]);
                                echo $vlan_stmt->fetchColumn();
                                ?>
                            <?php else: ?>
                                -
                            <?php endif; ?>
                        </td>
                    </tr>
                <?php endforeach; ?>
            </tbody>
        </table>
```

La vue détaillée pour chaque client, avec sa configuration réseau affichée :

```
<!-- Onglet Télécharger Configuration -->
<div id="download-config" class="tab-content">
    <h3><i class="fas fa-download"></i> Télécharger la configuration d'un client</h3>
    <div class="search-box">
        <i class="fas fa-search"></i>
        <input type="text" id="search-download" class="form-control" placeholder="Rechercher un client...">
    </div>
    <div class="table-responsive">
        <table class="table" id="downloadTable">
            <thead>
                <tr>
                    <th>Nom</th>
                    <th>Fichier de configuration</th>
                    <th>Actions</th>
                </tr>
            </thead>
            <tbody>
                <?php foreach ($clients as $client):
                    $filename = strtolower(str_replace(' ', '_', $client['nom_client'])) . '.txt';
                    $filepath = 'configs/' . $filename;
                ?>
                    <tr>
                        <td><?= htmlspecialchars($client['nom_client']) ?></td>
                        <td><code><?= $filename ?></code></td>
                        <td>
                            <?php if (file_exists($filepath)): ?>
                                <a href="<?= $filepath ?>" download class="btn btn-primary btn-sm">
                                    <i class="fas fa-download"></i> Télécharger
                                </a>
                            <?php else: ?>
                                <span class="badge badge-danger">Fichier non trouvé</span>
                            <?php endif; ?>
                        </td>
                    </tr>
                <?php endforeach; ?>
            </tbody>
        </table>
```

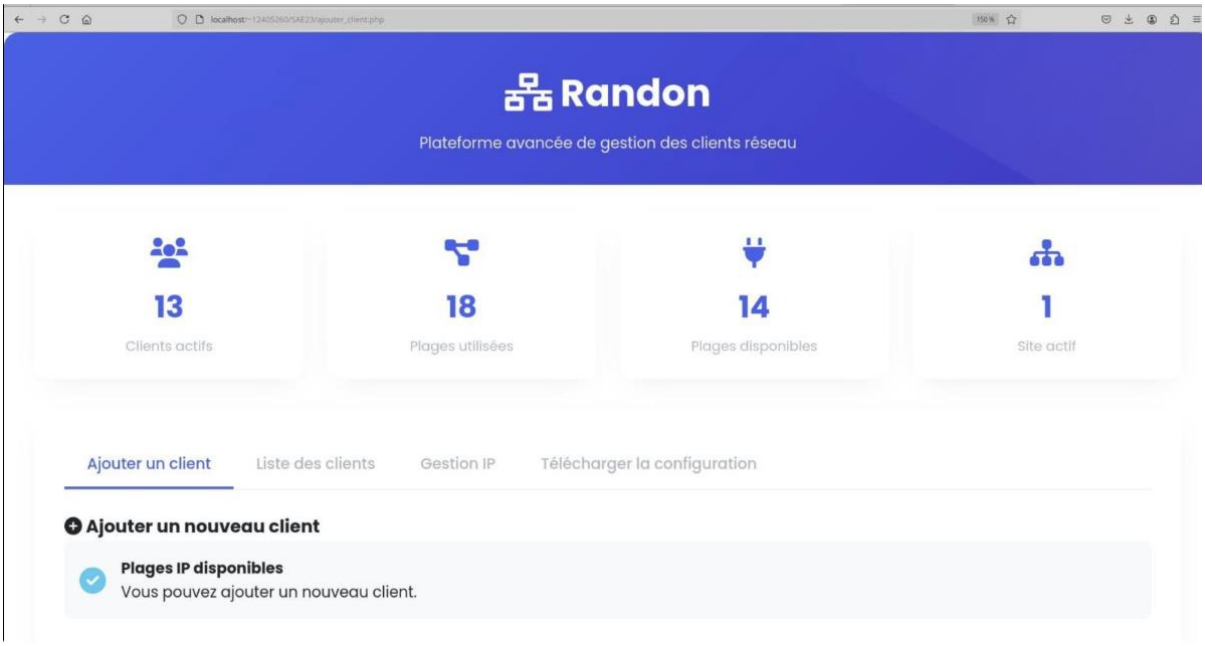
Notre plateforme

Nous avons choisi pour nom de notre site le nom Randon (Rayan + Andon), chaque client ajouté devient un client de ce site.

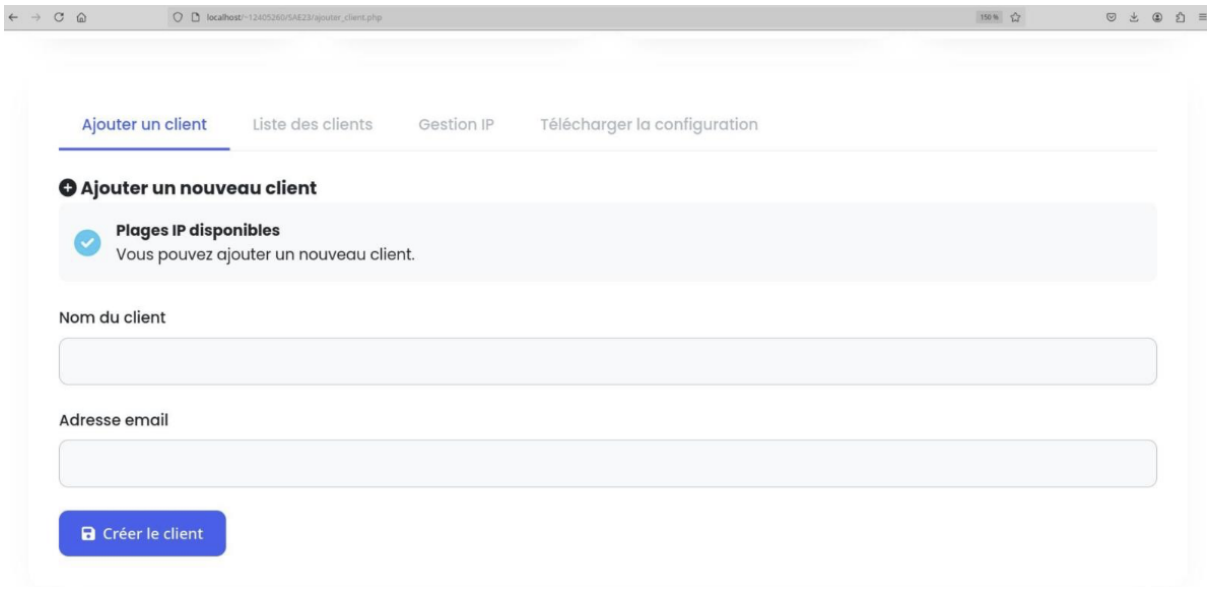
À propos de notre script. Ce script PHP constitue une plateforme complète de gestion réseau pour l'entreprise Randon, combinant back-end et front-end. Coté serveur, il gère l'ajout/suppression de clients avec attribution automatique d'adresses IP (plages /29 dans 164.166.1.0/24), de VLANs (ID = ID client) et de VRFs, tout en générant des fichiers de configuration Cisco (interfaces, VRF, route-targets). Les opérations s'appuient sur une base de données PostgreSQL avec des transactions sécurisées (beginTransaction()/commit()). Coté client, une IHM responsive (HTML/CSS/JS) propose des onglets dynamiques pour ajouter des clients, consulter la liste (avec recherche en temps réel), visualiser les plages IP, et télécharger les configurations. Le design utilise des animations CSS (fadeIn), des grilles flexibles, et s'adapte aux mobiles. Les fonctionnalités clés de notre plateforme incluent automatisation réseau, dont la génération de configurations routeur ; gestion des erreurs, dont les messages clairs (succès/échec) avec détails techniques ; sécurité, dont protection anti-injection SQL via PDO, suppression en cascade propre ; expérience utilisateur, dont Interface intuitive avec feedback visuel (badges colorés, notifications).

Démonstration d’utilisation

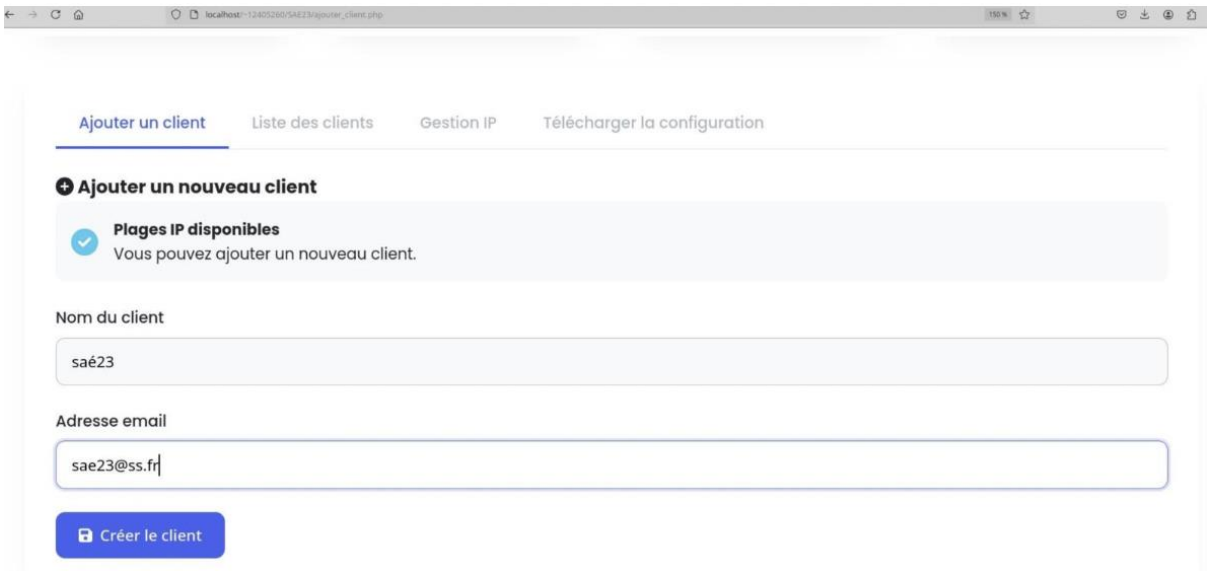
Commençons par la page d'accueil de notre plateforme, nous voyions le nom du site, le nembre de clients actifs, les plages disponibles, les plages utilisées et le site actif (Randon). Puis nous avons 4 onglets, dont ajouter client, liste des clients, gestion des IPs et téléchargement de la configuration.



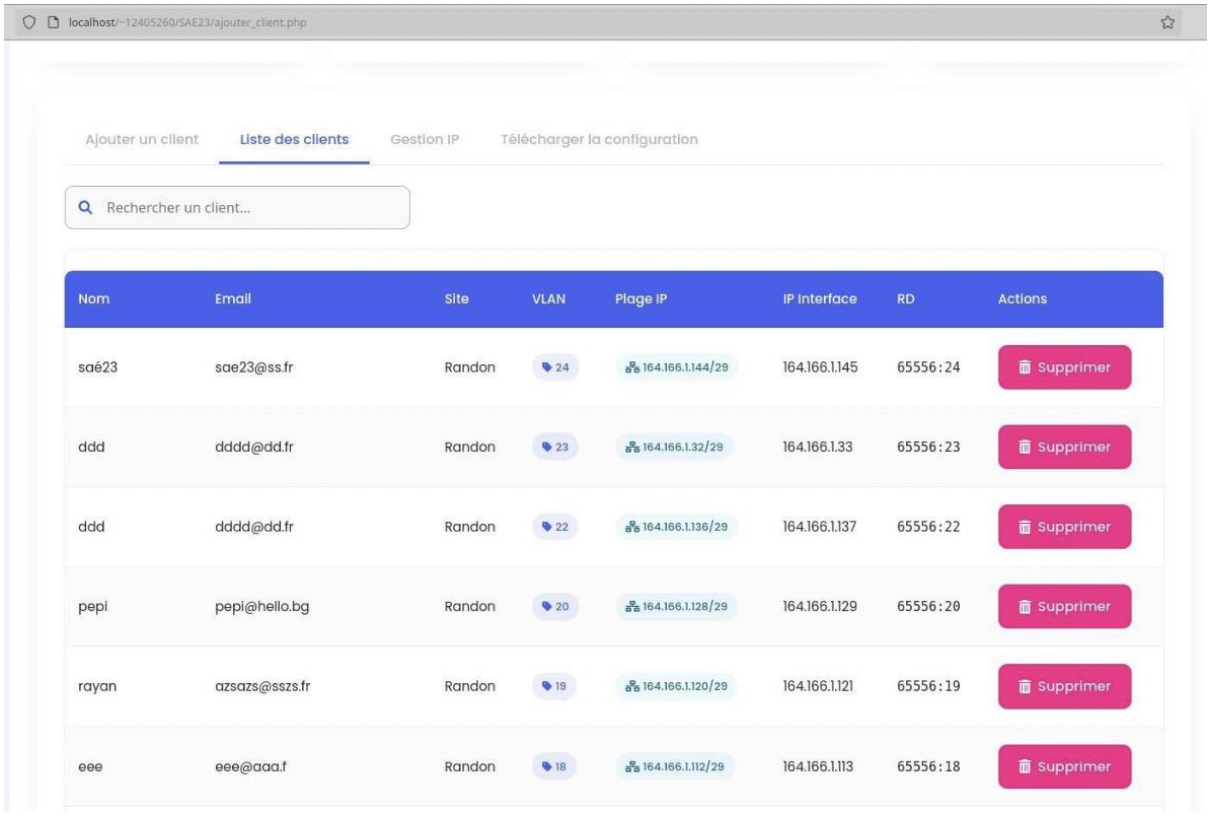
Si nous appuyons sur le bouton ajouter client, nous allons être demandé de choisir le nom du client et son mail.



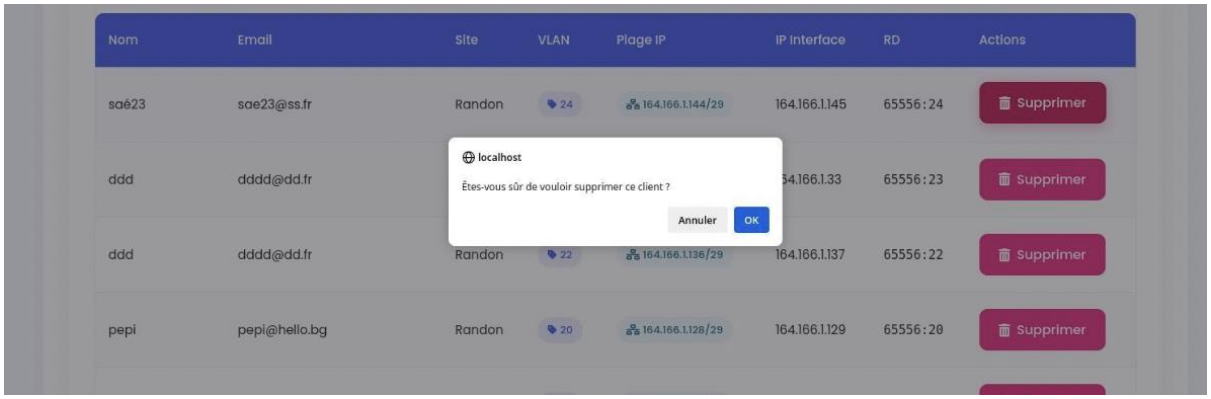
Nous allons créer un client :



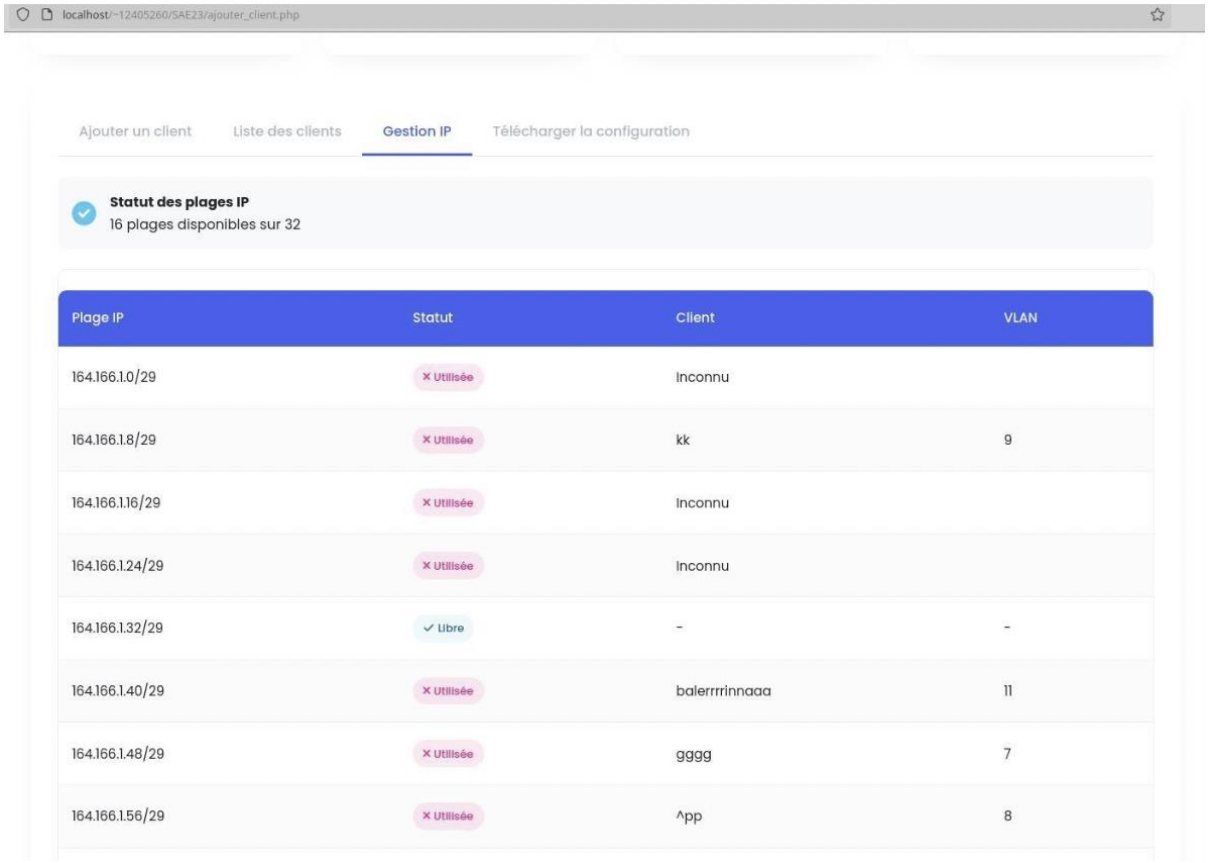
Une fois avoir appuyé sur le bouton “Créer le client” nous pouvons aller consulter la liste des clients pour vérifier que ceci a bien été créé.



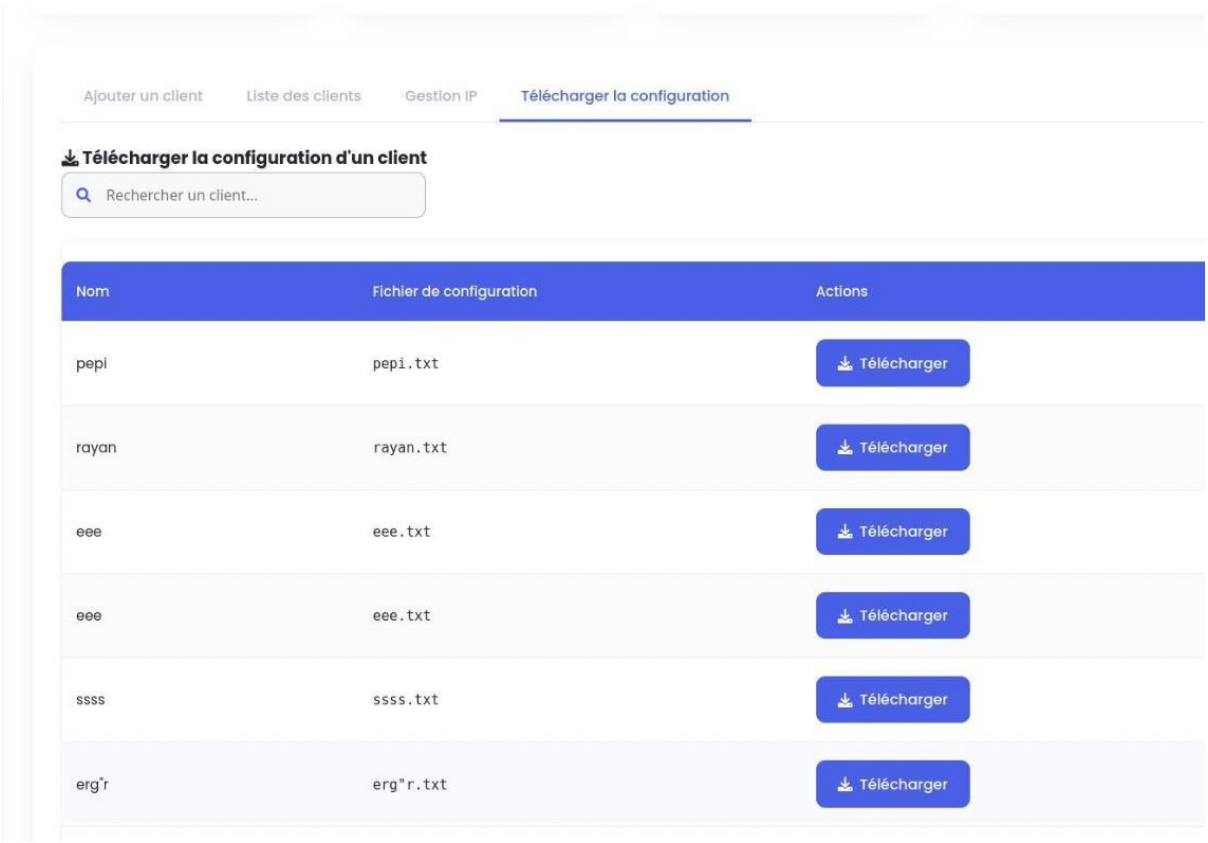
Nous avons la possibilité de supprimer n'importe quel client juste en appuyant sur le bouton “Supprimer” qui est située dans la colonne toute à droite dans l'onglet liste des clients.



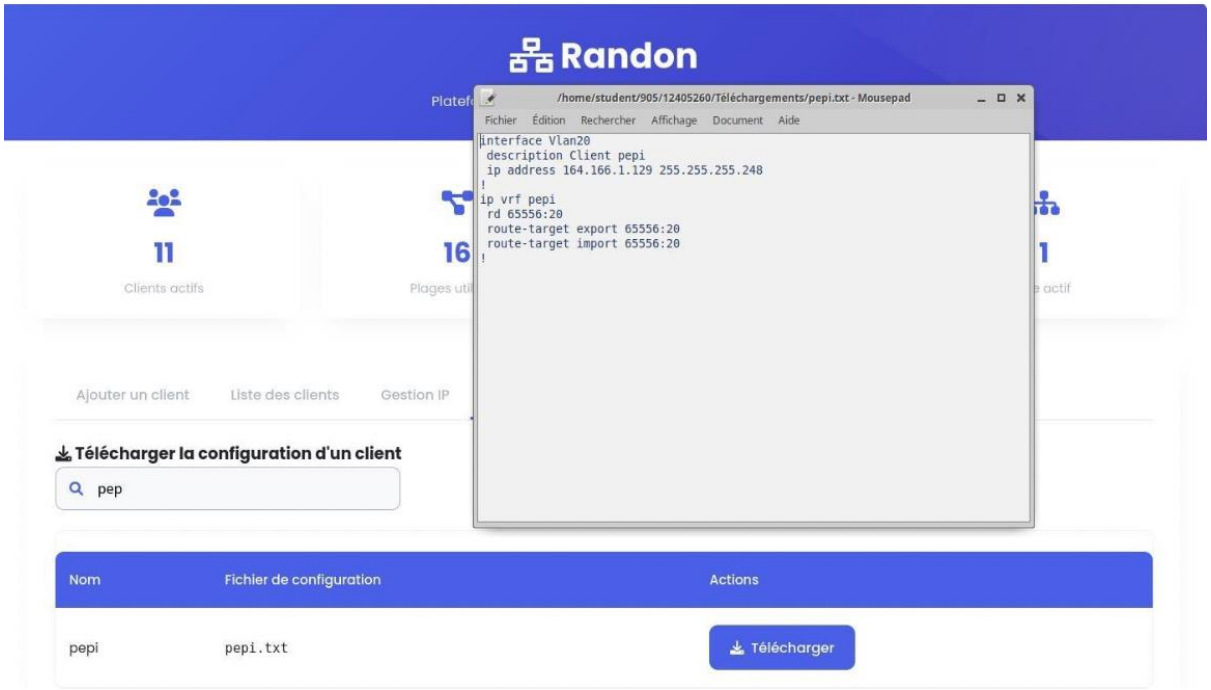
Juste après l’onglet liste des clients se trouve l’onglet “Gestion IP”, dans cet onglet nous pouvons voir quelles adresses IP sont disponibles et aussi les adresses IP qui sont déjà allouées.



À partir de l'onglet "Télécharger la configuration" nous pouvons trouver la configuration de chacun de nos clients.

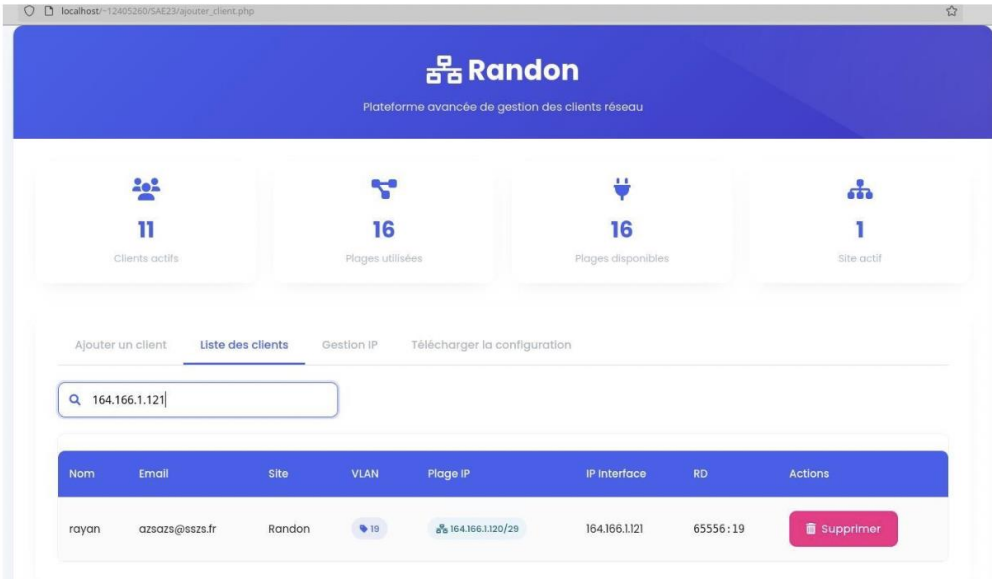


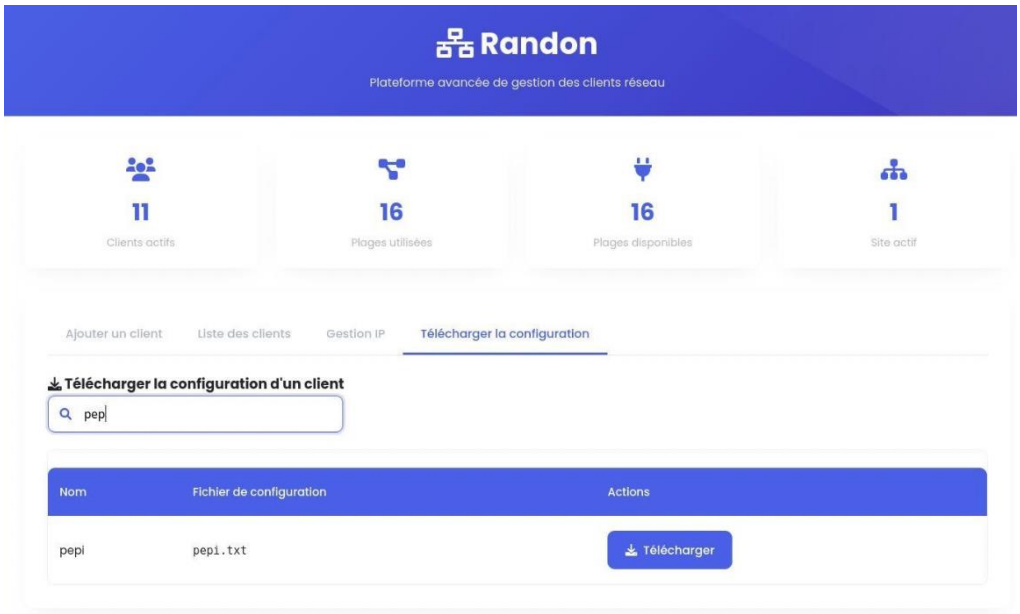
Une fois avoir appuyé sur le bouton “Télécharger” nous téléchargeons la configuration complète du client choisi.



Nous avons mis en place une barre de recherche dans l'onglet liste des clients et aussi une dans l'onglet télécharger la configuration. À partir de celle de la liste des clients nous pouvons trouver un client juste en

tapant soit son nom, soit son identifiant ou bien son adresse IP. Dans l'onglet télécharger la configuration nous pouvons trouver n'importe quelle configuration à partir de nom du client.





Conclusion

Ce projet a été l'occasion de mettre en pratique nos connaissances théoriques dans un contexte concret, en développant une solution complète allant de la modélisation des données à la création d'une interface utilisateur fonctionnelle. Grâce à cette plateforme, la start-up pourra gérer de manière automatisée et centralisée ses plages d'adresses IP, simplifiant ainsi l'administration réseau et améliorant la qualité de service pour ses clients.
Les défis techniques rencontrés, notamment l'automatisation des configurations et la synchronisation des données entre les sites, ont enrichi notre expérience et renforcé nos compétences en développement et en gestion de projet. Enfin, ce travail collaboratif a souligné l'importance d'une documentation claire et d'une méthodologie rigoureuse pour assurer la pérennité et l'évolutivité de la solution.

Nous sommes fiers du résultat obtenu et confiants que cet outil répondra aux attentes de l'entreprise, tout en offrant une base solide pour de futures améliorations.

IUT Villetaneuse
Université Sorbonne Paris Nord

2024/2025