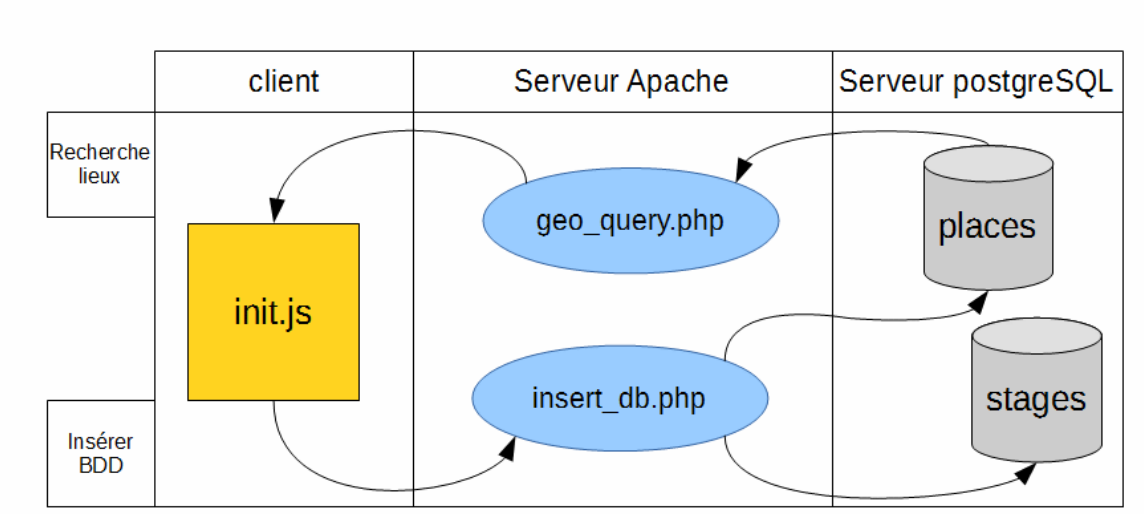
DOCUMENTATION PROGRAMMEUR

CARTE DES STAGES – GESTION DES STAGES AU MEME LIEU

FONCTIONNEMENT SCHEMATIQUE



PARTIE BASE DE DONNEES

Le serveur de base de données est de type postgreSQL, muni de postGIS.

La base de données (nommée par défaut « internships », cf. paragraphe PHP) comporte une table (par défaut « stages ») qui correspond aux stages, et une autre (par défaut « internship_places ») qui correspond aux lieux de stages.

PARTIE PHP

MODIFICATION DE L'ACCES A LA BASE DE DONNEES

```php
// Lien vers la base de données postGIS en local
$link = pg_connect("host=localhost port=5432 dbname=internships user=postgres password=postgres");
```

On peut modifier les paramètres d'accès à la base de données (adresse, port, nom de la base, utilisateur, mot de passe) dans les 2 fichiers PHP : à la **ligne 9** dans *geo_query.php*, et à la **ligne 11** dans *insert_db.php*.

```php
// Nom de la table dans la BDD
$table_name = "internships";
```

Pour modifier le nom des tables auxquelles on accède, il suffit de modifier la variable « table_name », situé à la **ligne 6** dans *geo_query.php*, et les variables « internship_table » et « place_table » aux **lignes 3 et 4** dans *insert_db.php*.

GEO_QUERY.PHP

Il s'agit de rechercher les lieux de stages proches du marqueur rentré par l'utilisateur, afin, le cas échéant, de les lui proposer.

Une fois la connexion avec la BDD faite et les noms de tables renseignés, on récupère la latitude et la longitude envoyées par le client. On réalise une requête spatiale pour trouver tous les stages ans un rayon de 28km du marqueur. S'il n'y a rien, on revoie une chaîne vide au client, sinon, on renvoie au format JSON le tableau des résultats.

INSERT_DB.PHP

On récupère le JSON envoyé par le client : un tableau associatif de tous les champs renseignés dans le HTML. La nature de l'attribut « location » du JSON dépend de si le lieu existe déjà ou non. Si le lieu existe déjà, il est caractérise uniquement par son ID dans la base de données. Sinon, il est caractérisé par son nom, une latitude et une longitude, qu'il faut insérer à la table correspondant aux lieux de stages. On va ensuite rechercher son ID (par construction, le dernier existant) afin de renseigner la clef étrangère de la table des stages.

Cela étant fait, on construit la requête permettant l'insertion dans la base de données du stage en parcourant le JSON que l'on a récupéré, puis on l'exécute.

PARTIE CLIENT

On utilise l'API cartographique GoogleMaps pour le positionnement du marqueur.

INITIALISATION DE LA PAGE

La fonction *window.onload*, qui est donc lancée une fois la page chargée, est la fonction *init().* Elle fait d'abord appel à la fonction initMap(), qui initialise la GoogleMap, puis initialise les éléments DOM et leurs écouteurs d'évènements, et ajoute le marqueur sur la carte.

On peut modifier le centre de la carte par défaut dans l'attribut *center* de la map, avec des coordonnées au format latitude, longitude.

VALIDATION DU FORMULAIRE

À la validation du formulaire, on récupère la latitude et longitude du marqueur, et on envoie une requête AJAX au serveur via le fichier *geo_query.php* pour obtenir les lieux de stages éventuels proches du marqueur. S'il n'y a rien, on met directement à jour la BDD. Sinon, on récupère le JSON envoyé par le PHP afin de peupler un select avec les noms des stages trouvés (avec comme valeur de champ l'ID du lieu dans la table des lieux de stages), et une div qui s'affiche par-dessus la page. Si l'un des lieux est reconnu par l'utilisateur comme son lieu de stage, il confirme et on met à jour la BDD. Sinon, il peut indiquer que son lieu de stage est un nouveau lieu, et il sera considéré comme tel dans la mise à jour de BDD.

MISE A JOUR DE LA BDD

À la confirmation de lieu de stage le cas échéant, ou sinon à la validation du formulaire, on crée un objet JavaScript que l'on peuple à l'aide du formulaire et du marqueur (ou de l'ID dans la table du lieu de stage le cas échéant). On envoie les données au fichier *insert_db.php* qui mettra à jour la base de données. À la réception de la réponse positive du serveur, on cache l'overlay s'il était visible.

Il existe aussi une fonction closeOverlay(), qui permet à l'appui sur l'icône de fermeture de l'overlay de le fermer.