

PROJET PWEBC

Une carte interactive avec géolocalisation

Étudiante : Chloé Hamilcaro 201

Professeur : J.-M. Illié

Présentation

Ce projet est réalisé dans le cadre du cours de PWEBC en deuxième année de DUT Informatique, et consiste à mettre en place une carte interactive en utilisant jQuery et JavaScript.

Mon projet est une **application web MVC** développée avec le framework **Symfony**, qui se base sur les données des **emplacements des gares en Île-de-France** fournies par IDF Mobilités. Les données sont utilisées sous forme d'un jeu : l'utilisateur doit retrouver l'emplacement des gares d'une ligne donnée en déposant ses réponses dans des boîtes. L'utilisateur peut également décider de ne pas jouer et d'afficher la carte seule : dans ce cas, il peut afficher la ligne dont il souhaite connaître le trajet en glissant sur la carte de nom de celle-ci, et définir des chemins entre deux points en cliquant sur un point de la carte puis sur un autre. Le développement back-end a été effectué en **JavaScript**, à l'aide de la bibliothèque **jQuery**, tandis que la mise-en-page a été effectuée grâce au framework **Bootstrap**.

L'application est composée de trois pages : une pour se connecter, une pour sélectionner la ligne de transport avec laquelle l'utilisateur souhaite jouer et une pour afficher la carte. L'accès au service se fait par un **mot de passe**. Puis, l'utilisateur peut choisir la ligne de transport avec laquelle il souhaite jouer : les **données privées** relatives aux gares sont contenues dans un fichier **GeoJson**, extraites grâce à une requête **Ajax**. Une fois la sélection effectuée, une carte **Leaflet** s'affiche, avec à droite le nom des stations de la ligne choisie, et en-dessous de la carte des boîtes qui permettent à l'utilisateur de déposer ses réponses. La carte fait apparaître les notions de **chemins**, **de couchés** et d'**événements**.

Le fonctionnement de cette application et les fonctionnalités développées sont détaillés dans ce rapport.

Mot de passe

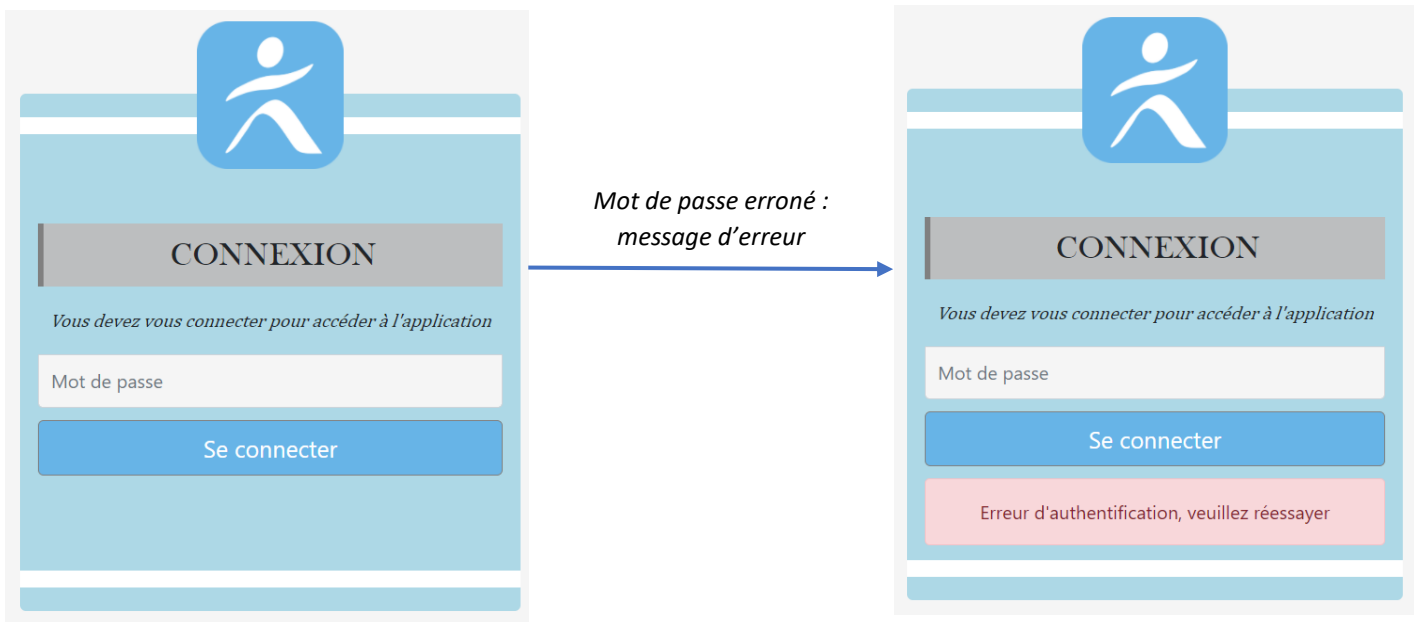
Le **mot de passe** pour accéder au service est : MAP123

Accéder au service : mot de passe

L'accès au service se fait donc via un **mot de passe**. Symfony permet de crypter le mot de passe avant que celui-ci ne parvienne au serveur, la connexion est donc entièrement sécurisée.

La mise en page de ce formulaire a été facilitée grâce à l'utilisation de **Bootstrap**, qui permet de faire des formulaires web-responsive.

Si le mot de passe rentré par l'utilisateur est faux, un message d'erreur apparaît :



Fonctionnalités JavaScript / jQuery

Sur cette page, j'ai utilisé une **animation jQuery** : le formulaire est un élément jQuery, tout d'abord caché. Lorsque la page est chargée, celui-ci s'affiche, avec une animation « clip ».

```
...
<script>
    $(document).ready(function() {
        $("#connexion").hide();
        $("#connexion").show("clip", {}, 1000);
    });
</script>
...
<form id="connexion" class="form-connexion text-center" method="post">
...
</form>
```

Extrait du code, avec le script et une partie du code HTML du formulaire

Sélection d'une ligne

Une fois authentifié, l'utilisateur peut **sélectionner la ligne de transports** avec laquelle il souhaite jouer. Les informations des lignes de transport et des gares se trouvant dans un fichier **GeoJson**, une requête **jQuery** et non Ajax a été effectuée pour pouvoir les extraire. J'ai choisi de procéder ainsi car, les données étant stockées sur une machine privée, aucune requête n'était applicable sur ce fichier et la structure jQuery était la plus simple.

```
$.getJSON("données/emplacement-des-gares-idf.geojson",  
    function(data) {  
        ...  
    }  
);
```

Extrait du code, avec utilisation de la fonction jQuery getJSON

L'utilisateur peut alors faire deux choix : sélectionner une ligne parmi les différents moyens de transport et cliquer sur le bouton Jouer, ou alors cliquer sur le bouton en haut à gauche qui permet d'afficher une carte vide, qui se remplira dynamiquement selon la volonté de l'utilisateur.

Fonctionnalités JavaScript / jQuery : onglets et sélecteur

La **navigation** entre les différents moyens de transports (Tramway, RER, Métros...) se fait au moyen d'**onglets jQuery**, qui sont une façon pratique et ergonomique de présenter un menu de navigation.

```
<script>  
    $( function() {  
        $( "#tabs" ).tabs().addClass( "ui-tabs-vertical ui-helper-clearfix" );  
    });  
    $( "#tabs li" ).removeClass( "ui-corner-top" ).addClass( "ui-corner-left" );  
    } );  
</script>  
...  
<div id="tabs">  
    <ul class="nav nav-tabs nav-fill mb-3">  
        <li class="nav-item"><a class="nav-link" href="#tabs-1">TRAMWAYS</a></li>  
        <li class="nav-item"><a class="nav-link" href="#tabs-2">MÉTROS</a></li>  
        ...  
    </ul>  
    <div id="tabs-1"><label>Tramways : </label></div>  
    <div id="tabs-2"><label>Métros : </label></div>  
    ...  
</div>
```

Extrait du code, avec le script et une partie du code HTML des onglets

Une fois toutes les lignes de transport extraites du fichier Json, celles-ci sont placées dans des sélecteurs correspondant à leur réseau de transport.

Comme vous pouvez le constater, les sélecteurs (balise `<select>`) n'apparaissent pas dans le code HTML initial, seul l'intitulé du moyen de transport apparaît, ce qui est normal car le squelette de l'application n'est pas censé connaître en avance combien de lignes chaque moyen de transport comptera. Les sélecteurs et les options sont ajoutées dynamiquement après le chargement des données Json, en jQuery.

```
if (type === "TRAIN") {
    for (I = 0; I < tablignes.length; i++) {
        txt += "<option value='" + tablignes[i] + "'>" + tablignes[i] +
"</option>";
    }
    divdonneesTrain.html(divdonneesTrain.html() + dForm + txt + fForm);
}
```

Extrait du script, exemple des trains : ajout du sélecteur et des options via jQuery

JOUER AVEC...

Vous pouvez jouer avec différents moyens de transport

TRAMWAYS

MÉTROS

RER

TRAINS

NAVETTES

GRANDES LIGNES

FUNICULAIRES

Trains :

LIGNE H

Jouer

Aspect final du menu de navigation par onglets

Certaines lignes de transport ne sont pas très intéressantes, comme les funiculaires : il n'y en a qu'un en Île-de-France, celui de Montmartre. Ce choix est donc affiché mais **désactivé**, pour que l'utilisateur ne puisse pas jouer avec ces lignes.

JOUER AVEC...

Vous pouvez jouer avec différents moyens de transport

TRAMWAYS

MÉTROS

RER

TRAINS

NAVETTES

GRANDES LIGNES

FUNICULAIRES

Funiculaires :

FUNICULAIRE MONTMARTRE

Jouer

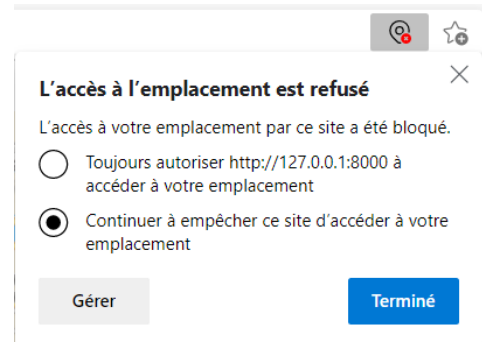
Funiculaires : un choix grisé

La carte Leaflet

La carte s'affiche grâce à **Leaflet**, une bibliothèque JavaScript qui permet d'afficher des cartes interactives : les éléments de cette carte sont tous des éléments JavaScript, ce qui permet de développer de nombreuses fonctionnalités.

J'ai utilisé le fond de carte de **Maptiler**, ainsi que **MapQuest** pour la définition de routes et de chemins.

Un des premiers défis de la mise en place de cette carte a été la **géolocalisation**. En effet, trois cas de figure se présentent : la géolocalisation n'est pas possible sur le navigateur utilisé par l'utilisateur, la géolocalisation est possible mais n'est pas autorisée par l'utilisateur, et enfin la géolocalisation est possible et autorisée. Il a fallu gérer ces trois cas de figure-ci.



Cas de figure : accès à l'emplacement refusé

La carte Leaflet est donc chargée de trois manières différentes. Si la localisation n'est pas possible ou n'est pas autorisée, la carte se **positionnera par défaut** sur les coordonnées de Paris. Si la géolocalisation est autorisée, elle se positionnera sur la géolocalisation de l'utilisateur.

Comme l'affichage de la carte peut prendre du temps, surtout si la géolocalisation est activée, un message s'affiche pour dire à l'utilisateur de patienter. Une fois la carte chargée, ce message disparaît.

```
divstatus.html("<i>Affichage de la carte en cours</i>");
```

Script qui affiche le statut de l'affichage avant le chargement complet de la carte.

```
divstatus.html("");
```

Script qui vide supprime le message indiquant à l'utilisateur de patienter.

Les coordonnées de Paris sont extraites d'une **requête Ajax** sur la base de données **Nominatim (données publiques)**.

```
$.ajax({
  url : "https://nominatim.openstreetmap.org/search",
  type : "GET",
  dataType : "json",
  data : {format: "json", limit: 1, country: "france", city: "paris"},
  success : function(data) {
    ...
  }
  ...
})
```

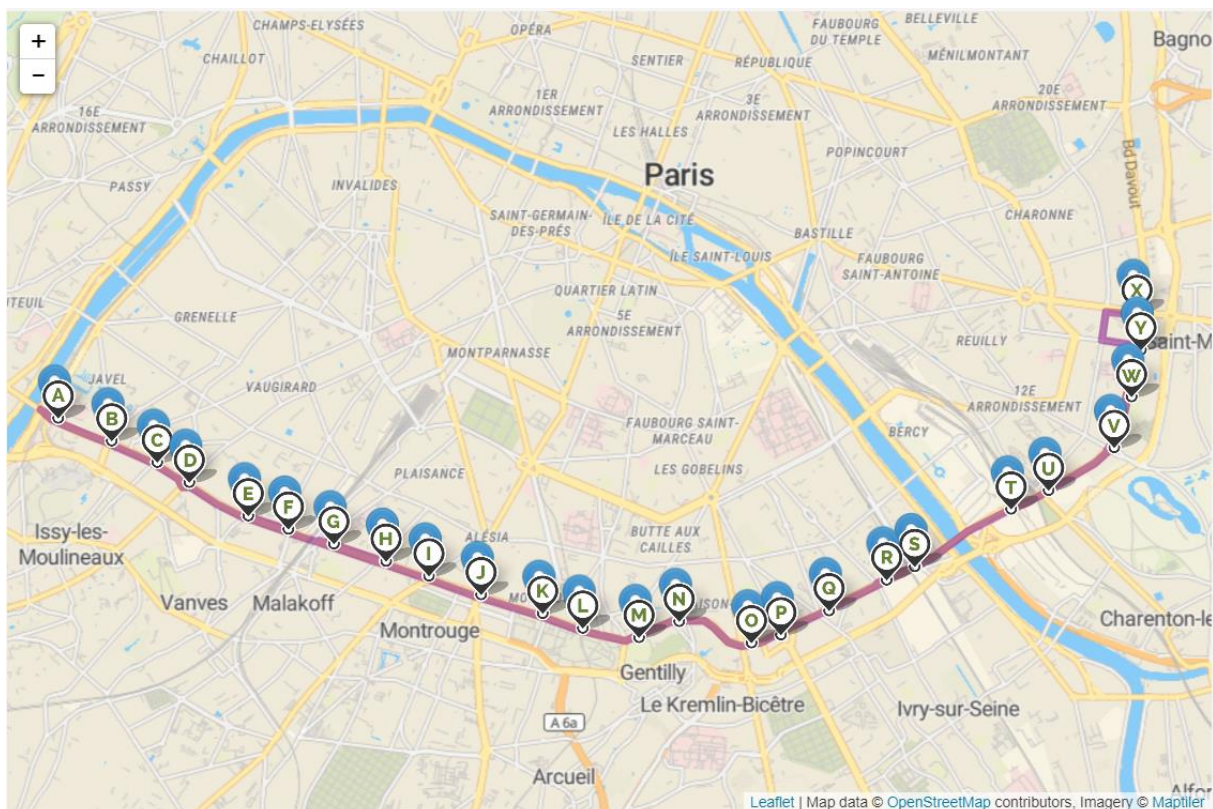
Extrait de la requête Ajax

La carte Leaflet : jouer avec les lignes

La carte Leaflet sert donc dans un premier temps à **jouer avec les lignes de transport**. Une **requête Ajax** sur le même fichier de données privées que celui utilisé pour l’affichage du sélecteur des lignes de transport permet de récupérer toutes les stations correspondant à la ligne passée en paramètre de l’URL (méthode GET).

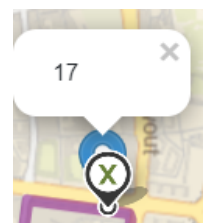
127.0.0.1:8000/map?num_ligne=T3A Exemple d’URL

La carte est donc chargée avec des points représentant les stations de la ligne choisie. Si celles-ci ne sont pas trop nombreuses, le **chemin entre ces stations** est affiché grâce à **MapQuest**. Comme MapQuest utilise le tracé des routes et que le métro, les trains et certaines navettes utilisent des autres voies, les chemins sont parfois erronés car ces voies particulières n’apparaissent pas sur une carte.



Exemple de la ligne T3A

En cliquant sur un point de la carte, un **pop-up** s’ouvre et l’utilisateur obtient le numéro de la boîte de réponse à remplir avec la station qu’il pense être la bonne. Par exemple, la station X porte le numéro 17 : si l’utilisateur connaît le nom de cette station, il doit déposer sa réponse dans la boîte numéro 17.



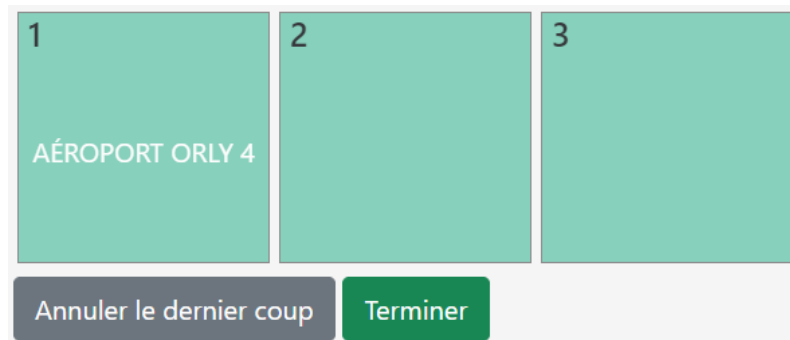
Fonctionnalités JavaScript / jQuery : draggable, droppable, barre de progression, alertes et redirections

Les réponses possibles sont affichées à gauche de la carte. Ce sont des éléments **jQuery draggable** : l'utilisateur peut les déplacer à sa guise sur la carte.

ANTONY
AÉROPORT ORLY 1-2-3
AÉROPORT ORLY 4

La case « Aéroport Orly 4 » a été déplacée par l'utilisateur

Les **boîtes de réponses** sont quant à elles **droppable** : l'utilisateur peut venir y déposer la réponse qu'il juge la plus pertinente. Lorsque la réponse est déposée sur la boîte de bonne réponse, celle-ci est enregistrée par une fonction JavaScript et la réponse disparaît du choix des réponses possibles. Le texte de la réponse devient alors celui de la balise <p> de la boîte de réponse.




L'utilisateur vient de la station « Aéroport Orly 4 » déposer sur la boîte 1

```
$(".droppable").droppable({
  drop: function (event, ui) {
    $(this).find("p").html(ui.draggable.find("p").html());
    var s = ui.draggable.find('p').html(); // Texte de la
station
    var indice = parseInt($(this).find('h5').html()); // Indice de
la boîte de réponse
    ui.draggable.empty(); // La réponse n'est plus disponible, on
vide sa div
    sauvegarderReponse(s, indice);
  }
});
```

Le code des boîtes de réponses de classe droppable

J'ai également incorporé une **barre de progression jQuery**, qui se remplit au fur et à mesure que l'utilisateur indique ses réponses. Ces barres ont comme valeur maximum le nombre de stations que comporte la ligne, et leur valeur s'affiche sous la forme d'un pourcentage de remplissage.

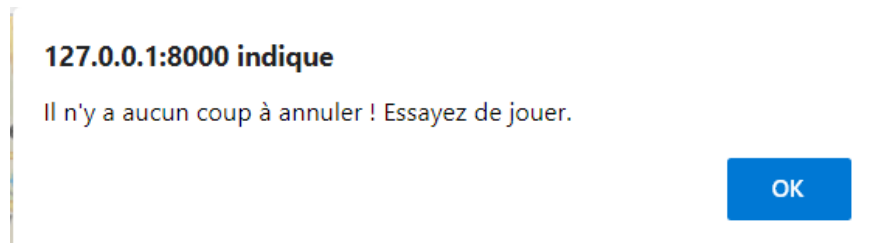


The image shows a user interface for a quiz. At the top, there is a purple progress bar that is approximately one-third full. To its right, the text "Progression : 33%" is displayed. Below the progress bar, the instruction "Glissez vos réponses dans les boîtes ci-dessous" is centered. Underneath this instruction, there are three green rectangular boxes labeled 1, 2, and 3. Box 1 contains the text "AÉROPORT ORLY 4". Boxes 2 and 3 are empty.

Une case sur 3 de remplie : progression à 33%

Il se peut cependant que l'utilisateur souhaite **annuler sa réponse** : un clic sur le bouton « annuler le dernier coup » déclenche une **fonction JavaScript** qui permet d'annuler le dernier élément déposé sur un bloc de réponse. Celui-ci devient alors vide, la jauge de remplissage diminue et l'élément réapparaît dans la liste des réponses.

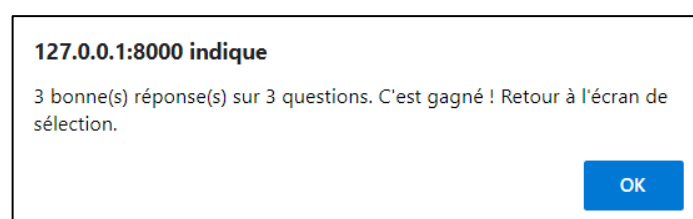
Si tous les blocs sont déjà vides, qu'aucun coup n'a été joué, une **alerte JavaScript** apparaît à l'écran :



The image shows a JavaScript alert box. The title bar reads "127.0.0.1:8000 indique". The main text inside the box says "Il n'y a aucun coup à annuler ! Essayez de jouer." In the bottom right corner, there is a blue button with the text "OK".

Le joueur clique sur « annuler le dernier coup » alors que toutes les boîtes sont vides : il y a erreur

Enfin, lorsque l'utilisateur décide de terminer de jouer, qu'il ait ou non rempli toutes les cases, il peut cliquer sur le bouton « Terminer ». Une fonction JavaScript s'enclenche, et si l'utilisateur a trouvé toutes les bonnes réponses, un message s'affiche pour le notifier de sa victoire et **l'utilisateur est redirigé** vers la page de sélection d'une ligne.



The image shows a JavaScript alert box. The title bar reads "127.0.0.1:8000 indique". The main text inside the box says "3 bonne(s) réponse(s) sur 3 questions. C'est gagné ! Retour à l'écran de sélection." In the bottom right corner, there is a blue button with the text "OK".

En revanche, si l'utilisateur n'a pas trouvé toutes les bonnes réponses, un message le notifie de son échec, et les boîtes de réponses **se remplissent des bonnes réponses**.

127.0.0.1:8000 indique

1 bonne(s) réponse(s) sur 3 questions. Perdu ! Les bonnes réponses vont s'afficher.

OK

La carte Leaflet : afficher la carte seule

La deuxième fonctionnalité développée est l'affichage de la carte seule. En effet, il se peut que l'utilisateur n'ait pas envie de jouer avec les lignes mais simplement de parcourir le tracé de celle-ci.

Fonctionnalités JavaScript / jQuery : draggable, droppable, couches et écouteurs d'événements

Dans ce cas-là, la barre de progression et les boutons permettant de jouer ne s'affichent pas, de même que les boîtes de réponses et de stations. Cependant, des boîtes semblables aux boîtes de réponses s'affichent : elles sont **draggable**, et contiennent le nom des lignes de transport en commun d'Île-de-France.

La **carte est droppable** : lorsque l'utilisateur dépose une ligne sur celle-ci, les stations et leurs noms apparaissent sur la carte. Pour faciliter la lisibilité de la carte, les lignes **repartent à leur position** une fois déposées sur un élément droppable grâce à l'ajout de l'option `{ revert: "valid" }`.



Affichage des stations de la ligne 10

Lorsqu'une autre ligne est déposée sur la carte, la **précédente couche est retirée** et une nouvelle ajoutée pour ne garder qu'une seule ligne sur la carte.

Enfin, l'utilisateur peut avoir envie de connaître le chemin pour aller d'une station à une autre : pour cela, un **écouteur d'événements** a été défini sur les points de la carte. Lorsque l'utilisateur clique sur un point, puis sur un autre, un chemin se dessine entre les deux.

