

Rapport du projet
Développement d'Applications
Réticulaires
« Meet me at the station »

2014

Sommaire

I. Introduction

II. Technologies utilisées

III. Structure du projet

a) Côté serveur

b) Côté client

c) Base de données

IV. Fonctionnalités

a) Frontend

b) Backend

V. Problèmes rencontrés

VI. Tests de validation

VII. Manuel d'utilisation

VIII. Conclusion

I. Introduction

Dans le cadre de notre projet de Développement d'Applications Réticulaires, nous avons pour objectif de réaliser une application web, utilisant une API de Transports publics, et dont la finalité de l'application est libre. Cependant quelques exigences techniques ont été imposé, comme l'architecture client / serveur web ou l'utilisation de servlets Java obligatoire pour la partie serveur.

Notre groupe a créé une application web, nommée « Meet me at the station », que vous trouverez sur <http://meet-meat.appspot.com>, dont les détails du développement sont décrits par la suite, nous verrons tout d'abord les choix des technologies, ainsi qu'une présentation de la structure du projet, ses fonctionnalités, puis nous parlerons des problèmes rencontrés. Pour compléter cela, nous avons également effectué des tests de validations et rédigé un manuel d'utilisation.

II. Technologies utilisées

Pour le développement du projet, nous avons eu recours aux langages HTML, CSS, JavaScript (incluant JQuery et AJAX) comme demandé, ainsi que Java, XML, JSP, et de certaines API, décrites ci-dessous.

L'API de Transports publics que nous avons choisi est Navitia, une API de type REST retournant les résultats sous format JSON. Elle nous permet de récupérer la liste des lignes et stations de métro, ainsi que les prochains départs du réseau RATP.

Pour héberger notre application web, Google App Engine a été retenu pour ses services complets et gratuits, acceptants les projets en Java. De plus, nous utilisons l'API Cron fournit par Google et les servlets Java, pour gérer la partie métier.

L'application nécessite une base de données pour les comptes utilisateur, les messages ainsi que pour récupérer la liste des stations, nous avons donc opté pour le datastore de Google App Engine, cependant son API bas niveau est difficile à manipuler. C'est pourquoi nous avons inclus Objectify qui est une API d'accès aux données Java, spécialement conçu pour Google App Engine datastore, mais créé par un développeur externe. Elle permet de faciliter la manipulation de la base de données.

Nous avons choisi, afin d'inclure une carte dans notre application web, l'API GoogleMap. Son interface est intuitive et offre la possibilité de personnaliser la carte. Dans l'application, elle permet de faciliter le repérage d'une station ou d'une ligne, et montrer visuellement un itinéraire à prendre.

III. Structure du projet

a) Côté serveur

La partie serveur de notre application, implémentée en Java, est structurée de la façon suivante :

Nous avons séparé chaque partie du serveur afin de mieux nous retrouver :

Le package « com.navitia », est le package contenant la servlet principale MeetMeAtTheStation.java, appelée lorsqu'un client accède à notre serveur.

Le package « com.navitia.datastore », contient les objets correspondant aux tables de la base de données.

Dans « com.navitia.servlet », nous avons mis les servlets permettant d'exécuter les différentes fonctionnalités de l'application.

Ensuite dans « com.navitia.servlet.backend », il y a les servlets du backend, celles qui sont appelées pour exécuter le travail côté métier.

Enfin le package « com.navitia.uti », contient les autres classes utilisées par l'application, aucune servlet n'est présente ici, il n'y a que les classes d'objets qui nous permet de nous faciliter le codage.

Un dernier dossier situé dans « war/jsp » liste tous les fichiers de type jsp.

Chaque servlet a sa propre fonction dans l'application, celles du package com.navitia.servlet servent à interagir avec le client :

- InscriptionServlet vérifie, lors de l'inscription du client, tous les champs du formulaire que le client a rempli et la servlet renvoie une chaîne de caractère (String) au client. Lorsque le client se connecte, cette même servlet renvoie du JSON.
- TchatServlet récupère les derniers messages postés et renvoie le résultat en HTML.
- LineServlet récupère les lignes de métro de la base de données et renvoie le résultat en JSON.
- StopAreaServlet récupère les stations de métro de la base de données et renvoie le résultat en JSON.

Nous avons défini un fichier XML cron.xml situé dans « war/WEB-INF/ » pour le backend, dans lequel nous indiquons les servlets à appeler, en précisant l'intervalle entre chaque appel. Ces servlets de backend sont défini dans le package com.navitia.servlet.backend :

- UpdateBackendServlet, récupère les prochains départs de chaque station toutes les minutes et les enregistre dans la base de données,
- TchatBackendServlet, supprime les messages des chats de la base de données, une fois par jour à minuit.

b) Côté client

La partie client est implémentée en HTML, CSS et JavaScript, dans le dossier war du projet, et est composé d'une seule page HTML.

Le style de la page a été séparé en trois parties, dans un dossier css :

- MeetMeAtTheStation.css, qui gère le style général de la page.
- partie1.css, principalement pour le bandeau, la carte et le menu.
- partie2.css, spécialement dédié à la partie chat.

De la même façon, nous avons un dossier js pour les scripts :

- script.js, est le script principale qui définit les variables globales et les fonctions appelées au chargement de la page.
- inscription.js, définit les fonctions qui gèrent l'inscription et la connexion.
- tchat.js, définit les fonctions pour générer les chats.
- carte.js, définit les fonctions d'affichage de la carte et de mise à jour de celle-ci
- menu.js, gère l'affichage dynamique du menu.

c) Base de données

L'application manipule quatre données principales : les lignes et les stations de métro, les comptes utilisateurs et les messages. La base de données de l'application contient donc ces quatre tables :

- Line

<i>Id</i>	<i>Name</i>	<i>Code</i>
line:RTP:1038770	Nation – Porte Dauphine	2

- StopArea

<i>Id</i>	<i>Name</i>	<i>Coord</i>	<i>LinesCode</i> *	<i>NextDeparture</i> *
stop_area:RTP:SA:1684	Ternes	48.878228/ 2.298121	[2]	["2/Porte Dauphine (Maréchal de Lattre de Tassigny) /20141030T152600", "2/Nation/20141030T152700"]

* LinesCode contient la liste des lignes qui passe par la stopArea.

* NextDeparture contient les prochains départs, sous la forme :
codeligne/direction/date.

- Utilisateur

<i>Id</i>	<i>Nom</i>	<i>Prénom</i>	<i>Pseudo</i>	<i>Email</i>	<i>Mot de passe</i>	<i>Amis</i> *
570015471	Sacquet	Frodon	Frofro	frodon.sacquet @gmail.com	sdasda	[-1L]

* Le champ Amis contient les id d'utilisateurs amis, ici l'utilisateur n'a pas encore d'amis.

- Message

<i>Id</i>	<i>Tchat</i> *	<i>Pseudo</i>	<i>Message</i>	<i>Date</i>
5684592452315	1	Frofro	Salut !	16:17:27

* Le champ Tchat correspond au code de la ligne de métro du chat.

IV. Fonctionnalités

a) Frontend

L'application dispose d'un système de comptes utilisateurs, nous pouvons donc distinguer deux types de clients :

Un utilisateur non connecté, peut visiter le site en tant que simple visiteur. Il peut donc :

- consulter la carte
- rechercher une station et l'afficher sur la carte
- connaître les prochains départs d'une station
- rechercher un itinéraire
- s'inscrire

Un utilisateur connecté peut profiter de toutes les fonctionnalités dont dispose un visiteur, et de plus accéder aux chats en ligne :

- sélectionner le chat de la ligne souhaitée
- consulter les dix derniers messages envoyés par les utilisateurs
- envoyer des messages
- se déconnecter

Nous disposons d'un chat par ligne de métro, qui serviront aux utilisateurs, empruntant la même ligne, de discuter entre eux, sur l'état de la ligne, ou pour se rejoindre et faire le trajet ensemble par exemple.

b) Backend

Pour le backend, nous utilisons Cron qui nous permet de mettre à jour la base de données de manière automatique. Il se charge plus précisément de la mise à jour des prochains départs toutes les minutes, ainsi que la suppression des messages de chats, une fois par jour.

V. Problèmes rencontrés

Lors de la réalisation du projet, nous avons rencontré certains problèmes, que nous n'avons pas pu remédier :

Notre application étant hébergé sur Google App Engine gratuitement, nous sommes limités sur le nombre de requêtes à la base de données, à 50 000 par jour. Ce qui n'est pas suffisant pour supporter le fonctionnement de notre application en temps normal.

IMPORTANT : Afin de pouvoir déployer une application fonctionnelle nous avons fait le choix de ne mettre à jour que les prochains départs de la station « Jussieu », mais le code permettant de mettre à jour toute les stations est présent dans le fichier InscriptionServlet.java en commentaire.

De plus, les données récupérées de l'API Navitia ne sont pas toujours correctes. En effet, lorsque nous récupérons les prochains départs d'une station de la ligne 7, il n'y a aucun train pour la direction Villejuif, car l'API ne fait pas la différence entre les deux directions du sud : Mairie d'Ivry et Villejuif, soit un train sur deux dans la direction Mairie d'Ivry est en vérité en direction de Villejuif et nous ne disposons d'aucun moyens de les différencier.

Enfin nous avons eu un souci au niveau des heures de départs rendu par navitia, en effet l'heure de départ obtenu est en retard d'une heure par rapport à l'heure réelle, nous ne savons si cela est du au changement d'heure.

VI. Tests de validation

Pour s'assurer du bon fonctionnement de notre application, nous avons effectué les tests de validation suivants :

Titre	Connexion - utilisateur enregistré
Contexte	L'utilisateur est sur la page d'accueil du client. L'utilisateur Frodon Sacquet est enregistré dans la base de données.
Entrée	login : " frodon0@gmail.com ", password : "sda".
Scénario	1. L'utilisateur rentre ces informations dans le panel de connexion. 2. Il valide le formulaire.
Résultat attendu	L'utilisateur est connecté à l'application et un cookie est créé avec son identifiant.
Vérification	Visuel, l'utilisateur voit s'afficher le message de bienvenue de l'application et peut accéder aux chats.

Titre	Déconnexion - utilisateur enregistré
Contexte	L'utilisateur est sur la page d'accueil du client. L'utilisateur est connecté dans l'application.
Entrée	
Scénario	1. L'utilisateur clique sur le lien de déconnexion.
Résultat attendu	L'utilisateur est déconnecté de l'application et le cookie contenant son identifiant est supprimé.
Vérification	Visuel, l'utilisateur ne peut plus accéder aux chats.

Titre	Chat - utilisateur n'est pas connecté
Contexte	L'utilisateur est sur la page d'accueil du client.
Entrée	
Scénario	1. L'utilisateur sélectionne un champs de chat.
Résultat attendu	L'utilisateur ne voit pas les messages du chat et ne peut pas envoyer de message.
Vérification	Visuel, un message de demande de connexion est afficher dans le champ de chat.

Titre	Chat - utilisateur est connecté
Contexte	L'utilisateur est sur la page d'accueil du client.
Entrée	
Scénario	1. l'utilisateur sélectionne un champs de chat. 2. l'utilisateur saisi un message et clique sur envoyer.
Résultat attendu	L'utilisateur voie les dix derniers messages du chat et peut envoyer des messages
Vérification	Visuel, les dix derniers messages du chat son afficher et son message aussi affiché.

Titre	Recherche - station non existante
Contexte	L'utilisateur est sur la page d'accueil du client.
Entrée	Station : "aa"
Scénario	1. l'utilisateur clique sur "Recherche" dans le menu. 2. l'utilisateur entre le nom de station "aa".
Résultat attendu	Aucune station du nom de "aa" n'est trouvé.
Vérification	Visuel, un message d'erreur est afficher à l'utilisateur sous le bouton de recherche.

Titre	Recherche - station existante
Contexte	L'utilisateur est sur la page d'accueil du client. La station "Jussieu" est enregistrée dans la base de donnée.
Entrée	station : "Jussieu"
Scénario	1. l'utilisateur clique sur "Recherche" dans le menu. 2. l'utilisateur entre le nom de station "Jussieu".
Résultat attendu	Le marqueur de la station "Jussieu" est zoomé sur la carte est son nom est affiché pendant 5 secondes
Vérification	Visuel, la carte est zoomé sur le marqueur de "Jussieu" et son nom est affiché

Titre	Info Station
Contexte	L'utilisateur est sur la page d'accueil du client. La station "Jussieu" est enregistrée dans la base de donnée.
Entrée	
Scénario	1. l'utilisateur clique sur "Info Station" dans le menu. 2. l'utilisateur sélectionne une station de métro sur la carte.
Résultat attendu	L'utilisateur voie les informations de la station sélectionné.
Vérification	Visuel, liste des prochains départ, direction et heure de départ.

Titre	Itinéraire - sans départ et arrivée
Contexte	L'utilisateur est sur la page d'accueil du client.
Entrée	
Scénario	1. l'utilisateur clique sur "Itinéraire" dans le menu. 2. l'utilisateur valide sa recherche.
Résultat attendu	Aucun.
Vérification	Aucun.

Titre	Itinéraire - sans départ
Contexte	L'utilisateur est sur la page d'accueil du client.
Entrée	
Scénario	1. l'utilisateur clique sur "Itinéraire" dans le menu. 2. l'utilisateur sélectionne une station d'arrivée. 3. l'utilisateur valide sa recherche.
Résultat attendu	Aucun.
Vérification	Aucun.

Titre	Itinéraire - sans arrivée
Contexte	L'utilisateur est sur la page d'accueil du client.
Entrée	
Scénario	1. l'utilisateur clique sur "Itinéraire" dans le menu. 2. l'utilisateur sélectionne une station de départ. 3. l'utilisateur valide sa recherche.
Résultat attendu	Aucun.
Vérification	Aucun.

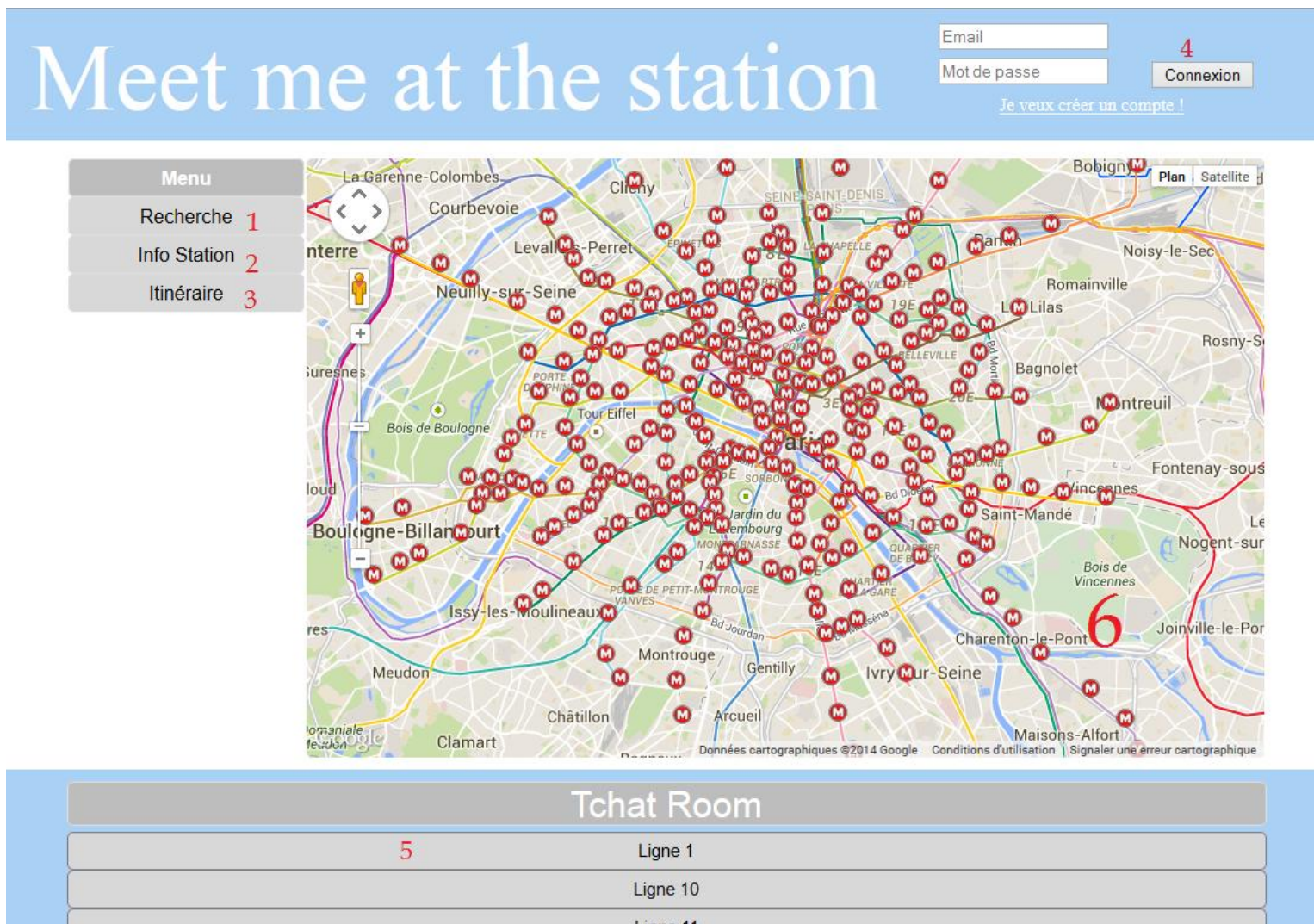
Titre	Itinéraire - avec départ et arrivée
Contexte	L'utilisateur est sur la page d'accueil du client.
Entrée	
Scénario	1. l'utilisateur clique sur "Itinéraire" dans le menu. 2. l'utilisateur sélectionne une station départ. 3. l'utilisateur sélectionne une station de d'arrivée. 4. l'utilisateur valide sa recherche.
Résultat attendu	On trace le trajet entre les deux stations sur la carte.
Vérification	Visuel, l'utilisateur voit afficher le trajet entre les deux stations sur la carte.

Nous avons de plus testé notre application sur les navigateurs et systèmes d'exploitation suivants :

- Google Chrome 38.0.2125.104 m, Windows 8
- Google Chrome 35.0.1916.114 m, Windows 7
- Firefox 32.0.3, Windows 8
- Firefox 26.0, Debian (machine de la PPTI)
- Safari 5.1.7(7534.57.2), Windows 8

VII. Manuel d'utilisation

Notre application web a été pensée afin que chacune des ses fonctionnalités soit visible et accessible le plus rapidement et le plus facilement possible. Voici donc comment naviguer sur notre application :



1. Cette fonctionnalité vous permet de rechercher une station :

Il vous suffit d'entrer un nom de station et une liste de stations vous sera proposé, Sélectionnez la station recherchée et cliquez sur « Rechercher », vous verrez alors la carte se centrer et zoomer sur la station que vous recherchez

Menu

Recherche

Rechercher une station

Rechercher

Info Station

Itinéraire

Attention si vous entrer un nom de station non existante vous ne pourrez pas la retrouver sur la carte...

2. Cette fonctionnalité vous permet d'obtenir des informations sur une station donnée, notamment ses prochains départs.

Menu

Recherche

Info Station

Cliquez sur une station de la carte

Itinéraire

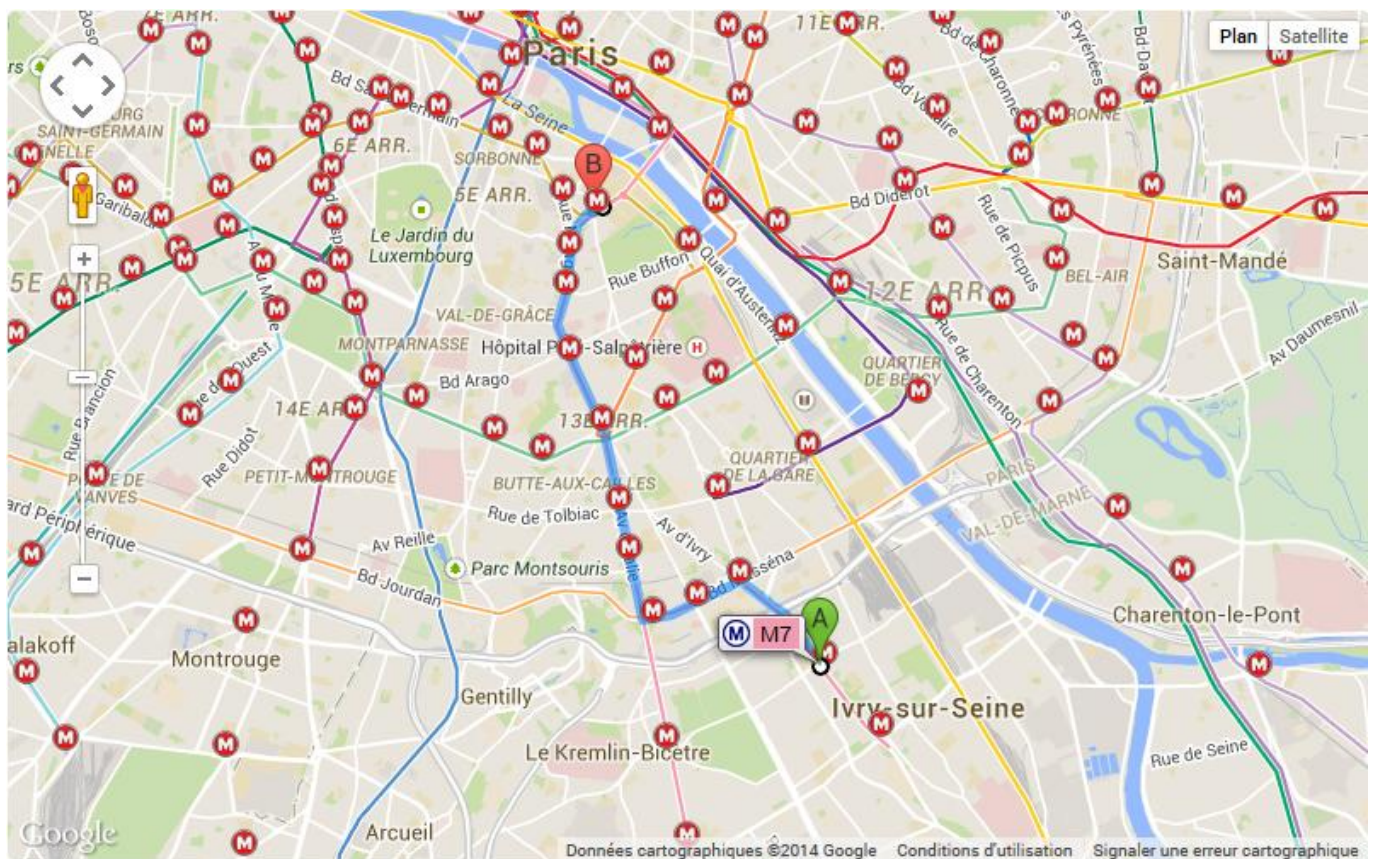
Une fois avoir sélectionné la fonctionnalité il vous suffit de cliquer sur l'une des stations présente sur la carte (représenté par un M dans un cercle rouge), vous aurez alors affiché dans le menu tout les prochains trains de la station que vous venez de sélectionner.

Menu	
Recherche	
Info Station	
Station: Jussieu	
Direction	Heure
[M10]Boulogne Pont de Saint-Cloud	15:26:00
[M10]Gare d'Austerlitz	15:27:00
[M10]Gare d'Austerlitz	16:21:00
[M7]La Courneuve-8-Mai-1945	15:27:00
[M7]La Courneuve-8-Mai-1945	15:30:00
[M7]La Courneuve-8-Mai-1945	00:50:00
[M7]Mairie d'Ivry	15:27:00
[M7]Mairie d'Ivry	15:31:00
[M7]Mairie d'Ivry	15:48:00
[M7]Mairie d'Ivry	15:59:00
Itinéraire	

3. Cette fonctionnalité vous permet de rechercher un itinéraire entre deux stations :

Menu	
Recherche	
Info Station	
Itinéraire	
Départ:	<i>clique gauche sur une station</i>
Arrivée:	<i>clique droit sur une station</i>
<input type="button" value="Rechercher"/>	
réinitialiser la carte	

Pour trouver un itinéraire il vous suffit de faire un clique gauche sur une station présente sur la carte afin de définir une station de départ et de faire un clique droit sur une station présente sur la carte afin de définir une station d'arrivée. Validez vos choix et la carte tracera automatiquement l'itinéraire le plus court entre ces deux stations.



4. Cette fonctionnalité vous permet de vous connecter aussi site, malgré le fait que chacune des fonctionnalités précédentes ne nécessite pas un compte utilisateur, si vous souhaitez participer à la communauté du site à travers les différents de chats vous devrez vous connecter :

Tout d'abord vous devez vous enregistrer, cliquer sur « Je veux créer un compte », vous verrez alors apparaître un formulaire d'inscription, remplir tout les champs et valider

A registration form titled "Inscription" on a light gray background. It contains six input fields for "Nom", "Prenom", "Pseudo", "Email", "Mot de passe", and "Confirmer le mot de passe". Below the fields are two buttons: "Je m'inscris !" and a link "Annuler".

Inscription

Nom

Prenom

Pseudo

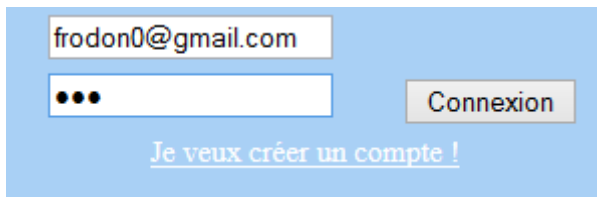
Email

Mot de passe

Confirmer le mot de passe

[Annuler](#)

Ensuite vous pourrez vous connecter en utilisant le formulaire de connexion de la page d'accueil, une fois connecté vous verrez le message de bienvenue suivant :

A login form on a light blue background. It has two input fields: one for the email "frodon0@gmail.com" and one for the password with three dots indicating it is hidden. There is a "Connexion" button and a link "Je veux créer un compte !".

frodon0@gmail.com

●●●

[Je veux créer un compte !](#)

A welcome message box on a light blue background. It displays the user's name "Bienvenue Frodon Sacquet0", their chat pseudo "userFrodon", and a "déconnexion" link.

Bienvenue Frodon Sacquet0

Ton pseudo pour les tchats: [userFrodon](#)

[déconnexion](#)

5. Cette fonctionnalité vous permet de participer à la communauté du site, ainsi vous pourrez discuter et rencontre des millions d'utilisateurs prenant chaque jour le même métro que vous. Vous pourrez discuter de l'état des lignes, vous tenir au courant des dernières actualités sur votre ligne favorite. Evidemment pour profiter de cette fonctionnalité vous devrez tout d'abord avoir un compte (cf.4) :

The image shows a web interface for a 'Tchat Room'. At the top, there is a header bar with the title 'Tchat Room'. Below this, there is a section for 'Ligne 1'. Inside this section, there is a chat window showing a message: '[15:16:25]userFrodon: Salut!'. Below the chat window, there is a text input field and a button labeled 'Envoyer'. At the bottom of the interface, there are two more sections labeled 'Ligne 10' and 'Ligne 11', each with its own text input field.

6. Enfin nous avons la carte sur laquelle vous pouvez zoomer dé-zoomer, cliquer sur une station de métro de votre choix.

VIII. Conclusion

Notre application « Meet me at the station » répond aux objectifs du projet et nous intégrons en plus une gestion d'utilisateur et un moyen de communication pour les utilisateurs connecté.

Notre équipe n'étant pas familiarisé avec le développement d'application web avant ce projet, ceci nous à permis de mieux comprendre le fonctionnement de ceux-ci et de découvrir de nouvelle technologie jusqu'alors jamais vu.

Finalement nous n'avons pas eu suffisamment de temps afin d'implémenter toute nos idée mais nous aurions voulu améliorer notre application en ajoutant une liste d'amis aux utilisateurs et ainsi intégrer une fonctionnalité de chat privé. De plus nous aurions souhaité étendre nos stations sur tout le réseau parisien sans nous limiter au métro.