Projet web service client consumer

|  |  |
| --- | --- |
| Auteur | Valentin Schneeberger |
| Date de début de projet | 19/12/16 |
| Date de fin de projet | 6/2/17 |
| Date de reddition du rapport | 19/12/16 - 6/2/17 |

Table des matières

[1 Introduction 3](#_Toc474152923)

[2 Documentation de développement 3](#_Toc474152924)

[2.1 Cahier des charges 3](#_Toc474152925)

[2.2 Définition des conventions applicables 3](#_Toc474152926)

[2.3 Planning de livraison global 3](#_Toc474152927)

[3 La phase de programmation 4](#_Toc474152928)

[3.1 La phase de test 4](#_Toc474152929)

[3.2 Un peu de code 6](#_Toc474152930)

[3.2.1 Code PHP concernant l’ajout d’une nouvelle ville dans la liste déroulante: 6](#_Toc474152931)

[3.2.2 C# 7](#_Toc474152932)

[4 Mode d'emploi utilisateur 8](#_Toc474152933)

[5 Conclusions 9](#_Toc474152934)

[6 Annexes 10](#_Toc474152935)

[6.1 Journal de bord 10](#_Toc474152936)

[6.2 Références 11](#_Toc474152937)

# Introduction

Le client a besoin de deux applications qui lui affiche de données météo, la première sera un client lourd et la deuxième un client léger.

# Documentation de développement

## Cahier des charges

Créer une application PHP et une deuxième application soit en c# soit en Android.

En l’occurrence la deuxième application a été créé en c#.

Les deux applications doivent afficher la météo pour une ville que l’utilisateur choisi dans une liste déroulante, l’utilisateur peut choisir également dans une liste déroulante combien de jours il veut afficher (entre 1 et 5).

L’utilisateur peut ajouter une nouvelle ville dans la liste déroulante pour autant qu’elle existe, par défaut il y a « Neuchâtel »,  « La Chaux-de-Fond », « Berne » et « Lausanne ».

Les localités que l’utilisateur ajoute sont en mémoire pour la prochaine utilisation.

Temps à disposition 70-75 périodes. A rendre le 6.2.2017

## Définition des conventions applicables

Les conventions de nommage des variables sont celle du CPLN. Aucune autre convention n’a été mise en place.

## Planning de livraison global



# La phase de programmation

## La phase de test

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test à effectuer** | | **Résultat**  **escompté** | **Résultat**  **obtenu** | **Constatation** |
| Généralités | | | | |
| PHP | Lancer l’application et regardez la liste déroulante « localité » | Les villes « Neuchâtel »  « La Chaux-de-Fonds »  « Berne »  Et « Lausanne » uniquement doivent être dans la liste déroulante |  |  |
| Regardez dans la liste nombre de jours à afficher | Les nombres doivent être listés de 1 à 5 |  |  |
| Changer la localité et sélectionnez Lausanne,  Changer le nombre de jours et inscrivez 4  Presser sur valider | La météo s’affiche pour aujourd’hui et les 3 prochaine jours pour la ville de Lausanne |  |  |
| Redimensionner la page en plus grand et plus petit | Le texte s’adapte au maximum à la grandeur de la page |  |  |
| Inscrivez « bonsoir » comme nouvelle ville à ajouter dans la liste. Presser sur valider | Bonsoir ne s‘ajoute pas dans la liste, un mot d’erreur s’affiche en haut à gauche de l’écran |  |  |
| Inscrivez « Genève » comme nouvelle ville à ajouter dans la liste. Presser sur valider | « Genève » s’ajout dans la liste déroulante |  |  |
|  | Inscrivez Neuchâtel comme nouvelle ville à ajouter dans la liste. Presser sur valider | Neuchâtel ne s’ajout pas dans la liste car il y est déjà |  |  |
| quitter le navigateur et relancer le site | « Genève » doit se trouver dans la liste déroulante avec les 4 autres villes précédentes |  |  |
| C# | Lancer l’application et aller dans la liste déroulante localité | Les villes « Neuchâtel »  « La Chaux-de-Fonds »  « Berne »  Et « Lausanne » uniquement doivent être dans la liste déroulante |  |  |
| Aller dans la liste déroulante des jours | Les jours doivent être listés de 1 à 5 |  |  |
| Changez la localité et sélectionnez Berne.  Changez le nombre de jours et inscrivez 2 Pressez sur valider | La météo s’affiche bien pour Berne pour les aujourd’hui et demain |  |  |
| Inscrivez « bonjour » comme nouvelle ville à ajouter dans la liste. Et cliquez sur ajouter | Un message d’erreur apparaît car la ville n’existe pas |  |  |
| Inscrivez « Neuchâtel » comme nouvelle ville à ajouter dans la liste. Et cliquez sur ajouter | Un message d’erreur apparaît car vous ne pouvez pas ajouter une ville qui est déjà là |  |  |
| Inscrivez « Basel » comme nouvelle ville à ajouter dans la liste et cliquez sur ajouter | « Basel » s’ajoute à la liste déroulante |  |  |
| Quittez l’application et relancez là | « Basel » doit être dans la liste déroulante en plus des autres villes |  |  |

## Un peu de code

### Code PHP concernant l’ajout d’une nouvelle ville dans la liste déroulante:

if (isset($\_GET['EnvoieVille'])) {

$reponse= $client->get($\_GET['Nouvelle\_Ville']);

$data = json\_decode(($reponse), true);

if (isset($data['errors'])) {

echo 'erreur';

} else {

$b = true;

foreach ($array as $value) {

if ($\_GET['Nouvelle\_Ville'] == $value) {

$b = false;

}

}

if ($b) {

setcookie($\_GET['Nouvelle\_Ville'], $\_GET['Nouvelle\_Ville'], time() + (86400 \* 365), "/"); //86400 = 1 day

header("Refresh:0");

}

}

}

Explication du code :

1. Au début nous regardons si l’utilisateur à pressé sur le bouton d’ajout d’une nouvelle ville.
2. S’il ne l’a pas fait la suite n’est pas appliquée, s’il a bien pressé le bouton nous allons commencer par chercher sur le site <http://www.prevision-meteo.ch/services/json> les informations concernant notre ville. Nous récupérons le résultat dans une variable qui s’appelle « $reponse ». Toute la connexion se fait au moyen de la méthode get qui se trouve ailleurs et que nous n’allons pas détailler ici.
3. Nous allons maintenant décoder la réponse, le résultat sera sous forme de tableau dans la variable $data.
4. Nous testons maintenant si il y a une erreur dans $data ce qui voudrait dire que l’utilisateur a mal rentré la ville qu’il voulait ajouter, si il y a une erreur nous affichons le mot errer à l’écran, une gestion différente de cette erreur peut être mise en place,  
   si il n’y a pas d’erreur nous pouvons passer à la suite du code.
5. Nous commençons ici par définir une variable temporaire à vrai, elle nous sera utilisée plus tard.
6. Nous allons maintenant parcourir un tableau initialisé ailleurs qui contient ce qui doit être affiché dans la liste déroulante. Si la valeur que l’utilisateur a rentrée se trouve déjà dans notre liste nous initialisons notre variable temporaire à faux pour qu’il n’y ait pas de doublons dans la liste.
7. Nous passons maintenant au dernier test, si notre variable temporaire est vrai nous créons un cookie d’une durée d’une année qui contient la valeur à ajouter dans la liste déroulante.
8. Pour terminer nous rafraichissons la page pour que la liste se mette directement à jour avec ce nouveau cookie.

Dans ce cas-là il a été choisi de travailler avec des cookies mais il y a bien d’autre solution, pour n’en citer qu’une nous aurions pu mettre en place une base de données avec les valeurs à afficher dans la liste déroulante et simplement ajouter dans la base une nouvelle valeur quand l’utilisateur veut en ajouter une.

Les avantages de la base de données par rapport aux cookies sont :

1. C’est très stable en comparaison des cookies qui peuvent être désactivé par l’utilisateur par exemple.

Les inconvénients de cette méthode sont :

1. Chaque utilisateur verra ce que les autres ont ajoutés dans la base à la différence des cookies qui sont personnalisés par l’utilisateur.
2. C’est très gros à mettre en place et à administrer pour finalement peu de données.

### C#

En c# il n’y a pas de bout de code spécifique à montrer. La méthode utilisée concernant l’ajout dans la liste déroulante vaut par contre la peine d’être expliquée.

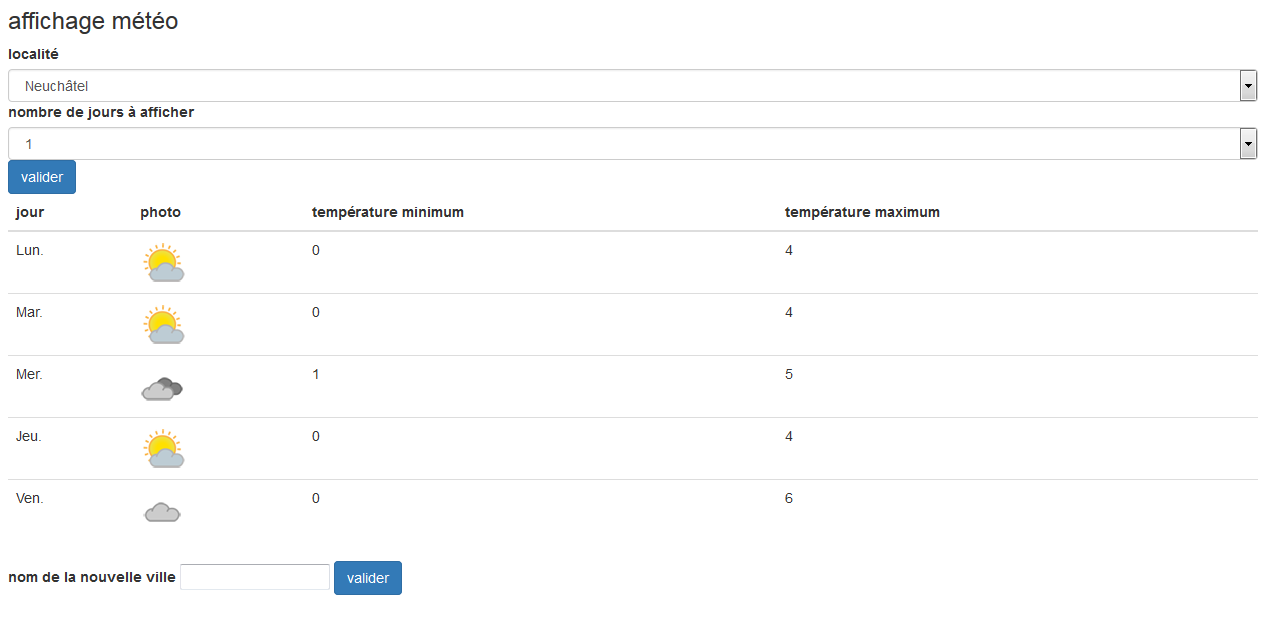
Quand l’utilisateur veut ajouter une nouvelle ville le programme effectue une série de test notamment pour savoir si elle existe bien ou pour savoir si elle est déjà dans la liste.

Si les tests sont positifs nous ajoutons la valeur dans un fichier texte.

Le fichier texte en question est, au lancement de l’application, lut et chargé dans une liste déroulante, ce fichier est aussi lut après qu’une ville lui ait été ajoutée pour que la liste se mette dynamiquement à jour.

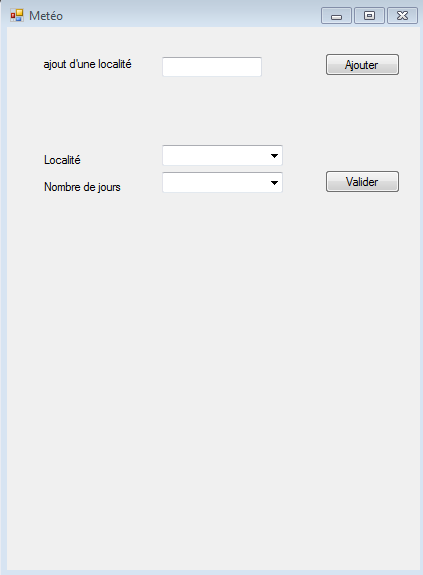
# Mode d'emploi utilisateur

En PHP,



1. L’utilisateur peut changer la localité et le nombre de jours qu’il souhaite afficher, dès qu’il aura pressé sur valider l’affichage se mettra à jour et montrera la météo concernant la ville choisie et le nombre de jours choisis.
2. Si l’utilisateur veut ajouter une nouvelle ville, c’est très simple il suffit de la rentré dans le champ « nom de la nouvelle ville » et de presser sur valider, attention il faut que la ville corresponde exactement avec la ville de l’url du site, par exemple pour La Chaux-de-Fonds on devra écrire la-chaux-de-fonds, ce format un peu contraignant sera une des prochaines mises à jour du projet.

En c#



Le fonctionnement est presque le même qu’en PHP,

1. Vous pouvez afficher la météo pour une localité choisie et un nombre de jours choisi en pressant sur valider.
2. Si vous voulez ajouter une nouvelle ville, faite le à l’aide du champ « ajout d’une nouvelle ville » et cliquer sur ajouter.
3. Comme en PHP il faut que vous ajoutiez exactement la même valeur que celle à inscrire dans l’url et comme en PHP ceci constituera une des prochaines améliorations du projet.

# Conclusions

Les deux applications correspondent au cahier des charges et sont rendus dans les temps, elles permettent entre autre d’ajouter de nouvelle localité et de consulter la météo pour les localités existantes, les données ajoutées sont encore présentes au prochains lancement de l’application.

Amélioration possible :

La gestion d’erreur est faite en c# comme en PHP mais les messages d’erreurs ne sont pas très claire, la limitation dans le temps a fait que j’ai donné une faible priorité à cet aspect.

Enter autre si je devais continuer ce projet je rendrais l’interface pour l’utilisateur plus agréables en rajoutant des messages quand les localités sont rentrées et quand il rentre des valeurs erronées.

Remarque

J’ai été limité dans le temps pour ce projet, en effet nous avons dû pendant la période de développement faire pas mal de choses qui n’avaient rien à voir avec le projet mais comme dit plus haut, le cahier des charges correspond à la demande.

Ce projet m’a permis de m’affiner dans le développement et la récupération de données en provenance de web service.

Il m’a permis aussi de me mettre en condition de TPI ce qui est une bonne chose.

# Annexes

## Journal de bord

|  |  |  |  |
| --- | --- | --- | --- |
| **Semaine N° : 17** | **Du : 19.12.2016** | **Au : 25.12.2016** | |
| Genre de travail | | | Durée |
| *Lundi: (9 périodes)* | | |  |
| Explication théorique cahier des charges projet météo | | | 2p |
| Mise en place trello/github et écriture planning projet météo | | | 3p |
|  | | |  |
| Implémentation des premières fonctionnalités | | | 2p |

|  |  |  |  |
| --- | --- | --- | --- |
| **Semaine N° : 20** | **Du : 16.1.2017** | **Au : 22.1.2017** | |
| Genre de travail | | | Durée |
| *Lundi: (9 périodes)* | | |  |
| Problème de sauvegarde des fichiers git le projet a dû être recommencé 4 périodes perdues. Recréation des répertoires git et de la configuration initial | | | 1p |
| Affichage en web responsive de la météo de Neuchâtel pour 5 jours | | | 3p |
| Création d’une liste déroulante qui récupère les villes sur le serveur | | | 1p |
|  | | |  |
| Adaptation de la liste déroulante qui ne répondait pas au cahier des charges | | | 2p |
| Recherche sur les cookies | | | 2p |
|  | | |  |
|  | | |  |
|  | | |  |
| Mardi (7 périodes) | | |  |
|  | | |  |
| Finition de l’application PHP, implémentation des cookies gestion des erreurs. Application PHP terminé | | | 5p |
| Recherche et développement de l’application C# | | | 2p |
|  | | |  |
| *Mercredi: (4 périodes)* | | |  |
|  | | |  |
| Problème c# concernant la récupération json du web service recherche et documentation, problème non résolu | | | 4p |

|  |  |  |  |
| --- | --- | --- | --- |
| **Semaine N° : 21** | **Du : 30.1.2017** | **Au : 5.2.2017** | |
| Genre de travail | | | Durée |
| *Lundi: (9 périodes)* | | |  |
|  | | |  |
| Absent | | | 9p |
| Mardi (9 périodes) | | |  |
| Finition de l’application c# consultant la météo,  Gestion de l’affichage  Gestion des erreurs  Gestion de l’ajout d’une nouvelle localité | | | 7p |
| Commencement du rapport concernant le projet météo | | | 2p |
|  | | |  |
|  | | |  |
| *Mercredi: (4 périodes)* | | |  |
|  | | |  |
| Rédaction rapport  Rédaction de commentaire c# | | | 4p |
|  | | |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Semaine N° : 22** | **Du : 6.2.2017** | **Au : 12.2.2017** | |
| Genre de travail | | | Durée |
| *Lundi: (9 périodes)* | | |  |
| Rédaction du rapport concernant le projet météo | | | 8p |

## Références

Page de documentation PHP et c# fournis par M. Sacchetti