DOCUMENTATION TECHNIQUE – RENDEV

MARINHO Johnny,
JAUCH Walter,
JEANRENAUD Nelson, GOUVEIA André, MACHADO Jorge

Projet de gestion d'évènements, ajout, modification et suppression via application c# avec une interface web

09.03.2020

Version 1.0

DOCUMENTATION TECHNIQUE – RENDEV

Table des matières

DOCUMENTATION TECHNIQUE – RENDEV

Introduction

Ce rapport est une documentation technique d'une application faite, en groupe, dans le cadre du module 306 supervisé par Mme Terrier. Il documente le code et l'ensemble des fonctions, méthodes et algorithmes du projet Rendev, développé par Jauch Walter, Gouveia André et Jeanrenaud Nelson pour la partie C# ainsi que Marinho Johnny et Machado Jorge.

Le projet est un gestionnaire d'évènements qui sont affichés sur la carte à l'aide d'une interface web. Les évènements sont donc ajoutés, modifiées et supprimées via l'application de bureau et ensuite, elles sont visibles, de manière synchronisée sur le site web.

Cahier des charges

Matériel et logiciels à disposition

- Visual Studio 2017
- Mysql
- Visual Studio Code
- OpenStreetMap (api WEB)
- OpenLayers (librairie JS)
- GMap (nuGet C#)

Descriptif complet du projet

Explication :

Un site web avec une carte sur laquelle on voit des évènements. Ces évènements peuvent être crée sur une application C#.

Côté Web :

Un site avec une seule page qui permet de visualiser les multiples évènements sur une carte dynamique utilisant la technologie api OpenStreetMap. Le visiteur du site peut rechercher les évènements selon différent critères tel que le nom, la date ou encore la catégorie.

Côté C# :

Une page d'accueil avec une carte qui permet à l'utilisateur de sélectionner une location l'utilisateur peut ensuite renseigner des champs, comme description, nom, date… sur un formulaire classique. Pour finir, il peut ensuite envoyer les points qu'il a créé sur le serveur pour qu'il puisse être affiché sur le site web.

Livrables

- Planning
- Rapport de projet
- Manuel utilisateur (si applicable)
- Journal de travail

Analyse de l'existant

Analyse des programmes similaires

- Le site McDonald's permet de trouver un emploi dans un périmètre donné et utilise un système d'objets sur une carte similaire à celle utilisée dans ce projet
- https://www.ge-soif.ch/ aussi permet de trouver des fontaines placées sur une carte ou l'on peut ajouter, modifier et supprimer des objets

Technologies à utiliser

OpenStreetMap (api WEB)

Selon Wikipédia : « OpenStreetMap est un projet Service collaboratif de cartographie en ligne qui vise à constituer une base de données géographiques libre du monde, en utilisant le système GPS et d'autres données libres. ».

C'est donc une api parfaite pour le client WEB. Elle est largement utilisée et permet une implémentation facile et logique dans notre projet.

OpenLayers (librairie JavaScript)

Selon Wikipédia : « OpenLayers est un logiciel libre, publié sous licence BSD. Il constitue une bibliothèque de fonctions JavaScript assurant un noyau de fonctionnalités orienté vers la mise en place d'applications clientes Web cartographiques fluides. »

Cette dernière va permettre aux événements d'apparaître sur la carte aux endroits où ils ont été créés via l'application C#. Elle permettra aussi d'afficher les informations de l'événement en question quand l'utilisateur interagit avec.

GMap

GMap (ou Great Map) est un contrôle .NET open source, puissante et, surtout, gratuit. On peut utiliser des cartes de Google, Bing, OpenStreetMap, Yahoo!, Pergo, WikiMapia et bien d'autres.

C'est ce que nous avons d'utiliser, sous conseils de notre enseignante (Mme. Anne Terrier), pour l'application C#. C'est grâce à GMap que nous afficherons les points déjà ajoutés et nous pourrons également récupérer l'adresse d'un point.

Analyse fonctionnelle

Méthodologie

Pour planifier ce projet nous nous sommes basés sur la méthode en 6 étapes que voici :

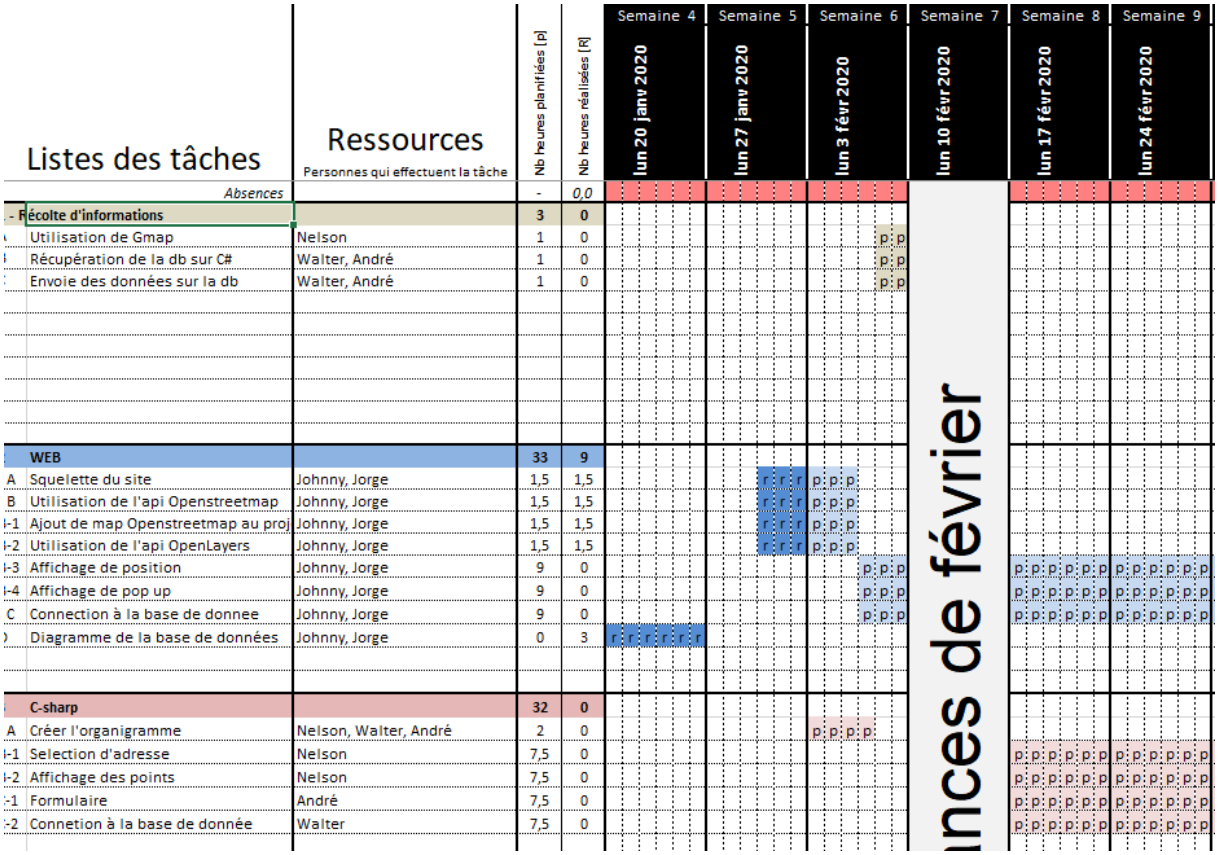
1. S'informer

La première étape du projet a été de s'informer sur l'énoncé de notre projet, sur les technologies à utiliser et les programmes similaires.

Nous avons donc défini le projet, compris le travail à faire. Ensuite, nous nous sommes documentés et informés sur les délais liés au projet.

2. Planifier

Dans cette deuxième étape nous avons planifié notre projet à l'aide de tâches attribuées aux membres de l'équipe. Nous en avons fait un tableau Excel pour suivre l'avancement du projet et des tâches à finir pour chaque personne dans le temps dont voici un extrait :



Listes des tâches	Ressources (Personnes qui effectuent la tâche)	Nb heures planifiées [p]	Nb heures réalisées [R]
Absences		-	*0,0*
- Récolte d'informations		3	0
Utilisation de Gmap	Nelson	1	0
Récupération de la db sur C#	Walter, André	1	0
Envoie des données sur la db	Walter, André	1	0
WEB		33	9
A Squelette du site	Johnny, Jorge	1,5	1,5
B Utilisation de l'api Openstreetmap	Johnny, Jorge	1,5	1,5
I-1 Ajout de map Openstreetmap au proj	Johnny, Jorge	1,5	1,5
I-2 Utilisation de l'api OpenLayers	Johnny, Jorge	1,5	1,5
I-3 Affichage de position	Johnny, Jorge	9	0
I-4 Affichage de pop up	Johnny, Jorge	9	0
C Connection à la base de donnee	Johnny, Jorge	9	0
Diagramme de la base de données	Johnny, Jorge	0	3
C-sharp		32	0
A Créer l'organigramme	Nelson, Walter, André	2	0
I-1 Selection d'adresse	Nelson	7,5	0
I-2 Affichage des points	Nelson	7,5	0
-1 Formulaire	André	7,5	0
-2 Connetion à la base de donnée	Walter	7,5	0

Semaine 4 (lun 20 janv 2020) | Semaine 5 (lun 27 janv 2020) | Semaine 6 (lun 3 févr 2020) | Semaine 7 (lun 10 févr 2020) | Semaine 8 (lun 17 févr 2020) | Semaine 9 (lun 24 févr 2020)

nces de février

Figure 1 : Diagramme de Gantt

Nous nous sommes ensuite mis d'accord sur les logiciels et les outils à utiliser et sommes passé à la prochaine étape du projet.

3. Décider

Dans cette étape, nous avons fait valider le cahier des charges avec notre mandant (l'enseignante) et, il fut validé après quelques modifications.

4. Réaliser

Nous nous sommes alors mis au développement, scindés en deux groupes : messieurs Marinho et Machado pour la partie web et messieurs Gouveia, Jauch, Jeanrenaud pour la partie C#. Toute l'équipe a également participé à la rédaction des différentes documentations.

5. Contrôler

Les plans de tests prévus n'ont finalement pas pu être mis en place suite aux mesures prises par le CFPT.

6. Evaluer

Nous avons, dans le but d'avoir un suivi, tenté d'analyser un maximum notre travail dans le journal de bord. Cela nous permet de garder un regard critique afin d'améliorer notre travail.

Implémentation

Base de données

Que ce soit l'application WEB ou l'application de bureau (C#), une base de données est indispensable pour que nous puissions stocker les données nécessaires au bon fonctionnement du système.

Nous nous sommes donc mis d'accord sur un schéma de base de données qui convenait aux utilisations des différentes applications. La base devait être capable de stocker tous les événements ajoutés. Cela implique la position à laquelle l'évènement est enregistré ainsi que la catégorie d'événements (Loisir, Travail, etc.). Voici le modèle que nous avons décidé d'appliquer :

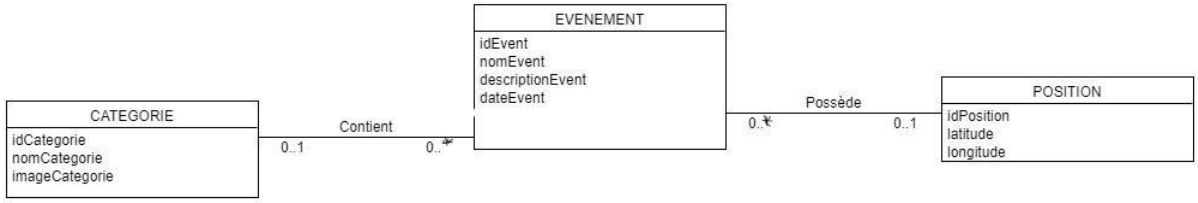


Figure 2: Diagramme de la base de données

C'est une base de données composée de trois tables. Les tables événement et position se remplissent simultanément étant donné que les positions stockées correspondent aux événements ajoutés sur l'application de bureau. La table concernant les catégories est également maintenue via l'application C#.

Structure du projet

Partie WEB

La structure du projet web est la suivante : trois dossiers, un pour le style, les fichier css, un autre pour le modèle de la base ainsi que pour le fichier de création de la base et enfin un dernier pour les images. Les fichiers php sont à la racine du projet.

Partie C# : Diagramme de classes

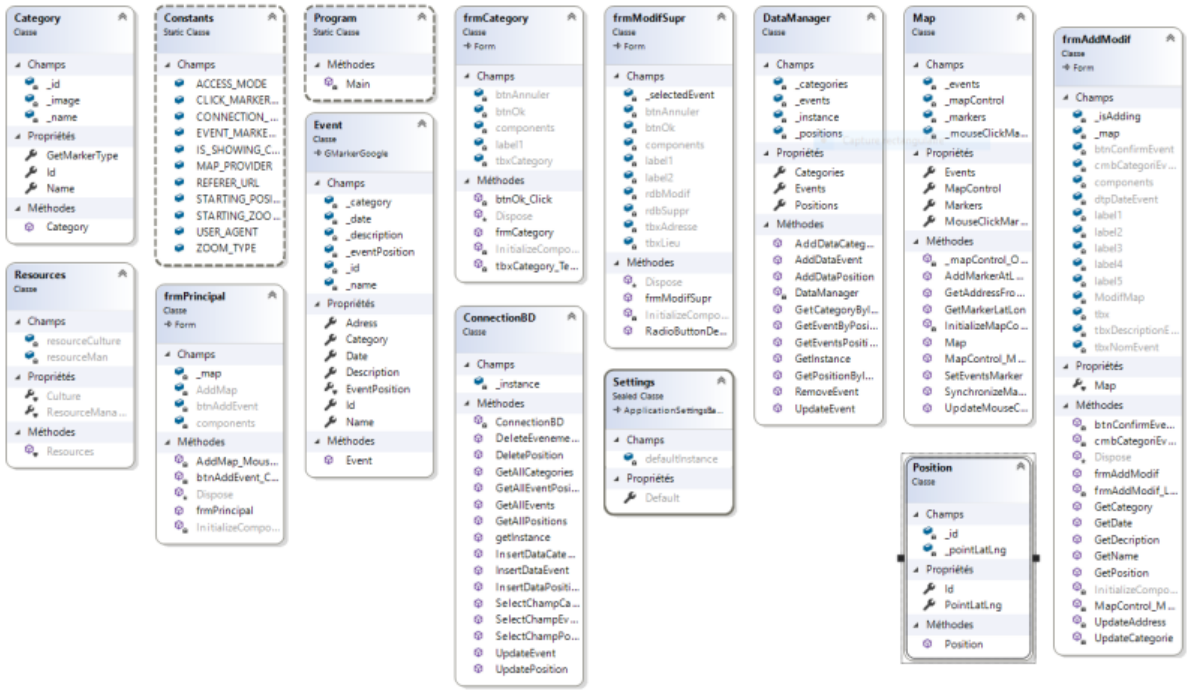


Figure 3 : Diagramme de classes

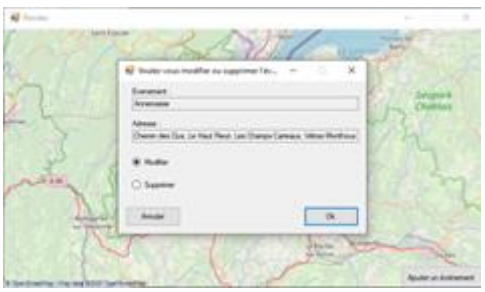
Présentation de l'interface C#

Fiche frmPrincipale

La fiche principale est simple elle contient une GMap qui permet d'afficher les points qui sont dans la base de donnée et d'un composant Button qui permet d'ouvrir une fiche pour l'ajout d'un événement.

Le but de la fiche est de voir les évènements sur la Map et de pour les gérées que ce soit en cliquant dessus pour choisir si on les modifie out les supprimes ou encore en cliquant sur le Button pour en ajouter des nouveaux.

Figure 4 et 5 : Forme principale



Fiche frmAddModif

La fiche AddModif contient une carte GMap pour pouvoir placer les points, trois text box qui permettent d'avoir le nom de l'événement, la description et le nom de la rue, il y aussi une combo box pour choisir la catégorie de l'événement, un date time picker pour choisir la date et un bouton pour enregistrer les données dans la base.

Le but de la fiche est d'ajouter des évènements dans la base en spécifiant les champs.

Figure 6 : Ajout d'un point

Fiche frmCategory

La fiche Category est très simple elle contient un text box pour avoir le nom de la catégorie et deux buttons qui sont « annuler » pour ne pas enregistrer dans la base les données et « ok » pour enregistrer les données dans la base.

Le but de la fiche est d'ajouter des évènements dans la base.

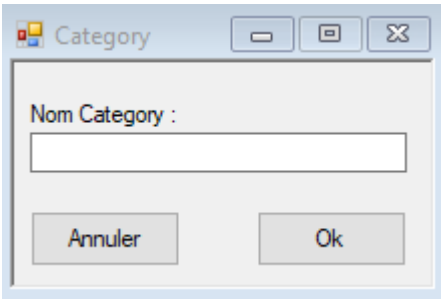


Figure 7 : Ajout de catégorie

Fiche frmModifSupr

La fiche ModifSupr contient deux text box qui indique le nom de l'évènement et l'adresse, il y a deux Radio button qui permet de choisir ce qu'on fait entre Modification ou Suppression et deux button qui sont « annuler » pour ne pas enregistrer dans la base les données et « ok » pour enregistrer les données dans la base.

Le but de la fiche est de choisir entre la modification et suppression de l'évènement.

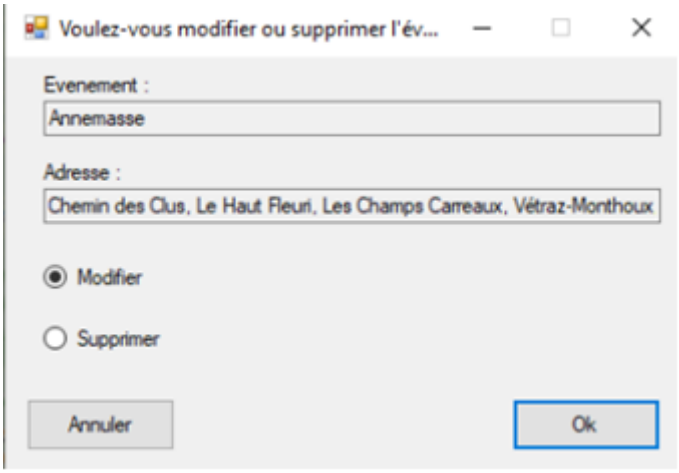


Figure 8 : Modification et suppression d'évènement

Analyse du code – C#

Dans cette section le fonctionnement interne de l'application C# sera lister et détailler. D'abord les actions principales que l'application effectue pour répondre au besoin de l'utilisateur. Puis l'utilité et le fonctionnement des classes de celle-ci.

Actions du programme

- Afficher les événements de la base de données sur la carte.
- Créer un événement en laissant l'utilisateur rentrer les détails de ce dernier et choisir l'adresse en cliquant sur la carte.
- Modifier des événements existants.
- Supprimer des événements existants.
- Ajouter des catégories d'événement
- Enregistrer les modifications d'événements et de catégorie à la base de données.

Description des classes

Classe DataManager

C'est la classe centrale du projet, elle lie les données de la base avec le reste de l'application. Elle implémente le design pattern singleton pour s'assurer qu'il n'y a qu'une seul instance de la classe. Elle contient la liste des catégories, des événements et des positions.

Méthode GetInstance() - Classe DataManager

Cette méthode publique et statique retourne l'instance de la classe et en crée une nouvelle si ell n'a pas encore été instanciée.

Méthode GetCategoryByIdIfExist(int paramId) - Classe DataManager

Cette méthode publique retourne une catégorie qui correspond à l'id envoyé en paramètre. Si elle n'existe pas la méthode retourne null.

Méthode GetPositionByIdIfExist(int paramId) - Classe DataManager

Cette méthode publique retourne une position qui correspond à l'id envoyé en paramètre. Si elle n'existe pas la méthode retourne null.

Méthode GetEventByPosition() - Classe DataManager

Cette méthode publique retourne la position de tous les événements enregistrés sous la forme d'une liste de point latitude/longitude

Méthode GetEventByPosition(PointLatLng paramPosition) - Classe DataManager

Retourne l'événement qui se trouve à la position envoyée en paramètre retourne null si aucun événement ne correspond.

Méthode RemoveEvent(string paramNom) - Classe DataManager

Cette méthode publique supprime un événement qui correspond au nom donné en paramètre

Méthode UpdateEvent(int paramId, string paramNewNom, string paramNewDescription, DateTime parmNewDate, Position paramNewPosition, Category paramNewCategorie) - Classe DataManager

Cette méthode publique prend en paramètre l'id de l'événement à modifier et les nouvelles valeurs. Il appelle ConnectionDB pour mettre à jour l'évenement en question.

AddDataPosition(double latitude, double longitude) - Classe DataManager

Cette méthode public ajoute une position à la base et retourne son id en type long.

AddDataEvent(string nameEvent, string descriptionEvent, DateTime date, int idPosition, int idCategorie)

Cette méthode public ajoute un événement à la base et retourne son id en type long.

AddDataCategory(string nameCategory)

Cette méthode public ajoute une catégorie à la base et retourne son id en type long.

Classe Map

Cette classe est initialisée dans toute les formes qui utilise la carte graphique. Elle utilise Gmap.Net pour modifier le contrôle GmapControl et afficher les points des événements sur les formes.
Elle requiert que ce contrôle lui soit passé en paramètre lors de son initialisation.

Elle contient également la totalité des marqueurs d'événement et le marqueur rouge du click.

InitializeMapControl() - Classe Map

Cette méthode privée est appelée à l'instanciation de la classe. Elle initialise la carte avec toute les options par défaut spécifié dans la classe Constants.cs. Elle ajoute également les événements Mouseclick et OnMarkerClick a ceux du contrôle.

UpdateMouseClickMarkerPosition(PointLatLng paramPosition) - Classe Map

Cette méthode publique modifie la position du marqueur de la souris à la position en paramètre. Elle est appelée lors d'un click sur la carte et lors de la création d'une forme pour placer le point au bon endroit.

AddMarkerAtLocation(PointLatLng paramPosition) - Classe Map

Cette méthode publique ajoute le marqueur à la position spécifié en paramètre.

SetEventsMarker(List<PointLatLng> paramEventsPositions) - Classe Map

Cette méthode publique utilise AddMarker pour ajouter un marqueur à chaque position spécifiée en paramètres

SynchronizeMapEventsWithServer() - Classe Map

Cette méthode publique va chercher les positions des événements dans le DataManager et appel SetEventsMarker pour placer les points correspondants au bonne endroit sur la map.

GetAddressFromLatLon(PointLatLng paramPosition) - Classe Map

Cette méthode statique et publique permet de convertir une coordonnés latitude/longitude en adresse si aucune adresse est trouvée elle lance une exception.

Classe Event

Cette classe qui hérite de GMarkerGoogle contient toute les variables propres à un événement. La propriétés Address utilise la méthode statique GetAddressFromLatLon de Map.

Classe Category

Cette classe sert à contenir les informations d'une catégorie.

Classe Position

Cette classe sert à contenir les informations d'une position.

Classe Constants

Cette classe contient contient toutes les constantes de connection à la base de données et pour les paramètres de base de la carte.

Classe ConnectionDB

getInstance()

Cette méthode retourne l'instance de la classe

SelectChampEvent(int idEvent, string typeValue)

Cette méthode permet de retourner un Evenement de la base de donnée en fonction de l'id.

SelectChampCategory(int idCategorie, string typeValue)

Cette méthode permet de retourner une Category de la base de donnée en fonction de l'id.

SelectChampPosition(int idPosition, string typeValue)

Cette méthode permet de retourner une Position de la base de donnée en fonction de l'id.

GetAllCategories()

Cette méthode permet d'avoir la liste de toute les Catégories.

GetAllPositions()

Cette méthode permet d'avoir la liste de toute les Positions.

GetAllEvents()

Cette méthode permet d'avoir la liste de tous les Evenements et de mettre à jour les listes Position et Category.

InsertDataEvent(string nameEvent, string descriptionEvent, DateTime date, int idPosition, int idCategorie)

Cette méthode permet d'ajouter un événement dans la base de donnée.

InsertDataPosition(double latitude, double longitude)

Cette méthode permet d'ajouter une position dans la base de donnée.

InsertDataCategorie(string nomCategorie)

Cette méthode permet d'ajouter une catégorie dans la base de donnée.

UpdateEvent(int idEvent, string nomEvent, string descriptionEvent, DateTime date, Position position, Category categorie)

Cette méthode permet de mettre à jour un événement dans la base de donnée.

UpdatePosition(int idPosition, int latitude, int logitude)

Cette méthode permet de mettre à jour une Position dans la base de donnée.

DeleteEvenement(string nomEvent)

Cette méthode permet de supprimer un Evenement de la base de donnée.

DeletePosition(int idPosition)

Cette méthode permet de supprimer une Position de la base de donnée.

Analyse du code - WEB

Les fonctions

Côté PHP, nous avons 3 fonctions. La première, **getEvents()**, nous permet de recevoir tous les évènements, ce qui permet, lors du lancement de la page, de récupérer leur latitude, longitude ainsi que leur id et de les placer sur la carte. Cette fonction utilise le design pattern singleton. On va donc créer une variable statique pour y stocker la requête. Ensuite, si cette variable n'a jamais été instanciée, nous allons le faire et l'exécuter dans la base et retourner le résultat à la fin. Si elle a déjà été instanciée, nous allons juste retourner son résultat puisque ça veut dire qu'elle a déjà été exécutée auparavant.

La deuxième, **getEventByID()**, va entrer en jeu lorsque l'utilisateur va sélectionner un évènement sur la carte. Cela va déclencher une action Javascript, que l'on décrira plus loin, qui enverra l'id de l'événement sélectionné à cette fonction, qui retournera ensuite un tableau avec les informations relatives à cet évènement. Cette dernière ne permet pas d'implémenter le design pattern singleton car elle prend en compte un paramètre qui peut changer dans la requête.

La troisième fonction, **displayEvent()**, permet d'afficher les informations précédemment reçues.

Coté Javascript, nous commençons par créer trois variables qui reprennent, avec la fonction json_encode, tous les évènements ainsi que la latitude et la longitude de celui sélectionné.

Pour la partie javascript, nous avons utilisé le framework openlayers https://openlayers.org/, en jointure avec openstreetmap https://www.openstreetmap.org/.
Nous avons utilisé donc dû créer des objets map et overlays, dans l'objet map nous avons également dû mettre en place une vue de départ, qui correspond au spawn sur la map.
Les objets utilisés pour le bon fonctionnement de la map sont donc :

```
ol.Map
ol.Overlay
ol.layer.Tile
ol.View
```

Nous avons également dû utiliser des objets pour les différents styles utiliser :

```
ol.geom.Circle
ol.style.Stroke
ol.style.Fill
```

Par la suite, nous avons dû prendre les informations venant de la base de données, afin de dessiner des cercles en fonctions des différents événements sur la base de données.

Pour ce faire, nous avons utilisé une fonction **AddLayer()**, qui prend en paramètre une position et un id.

Exemple :

```
  AddLayer(positionLatLonEvent, idEvent);
```

Pour l'ajout des cercles nous avons dû utiliser des nouveaux objets qui sont le suivants :

```
ol.layer.Vector
ol.Feature
ol.geom.Circle
```

Grâce à cela nous obtenons des cercles en fonction de la position des différents événements qui sont contenus dans la base de données.

Pour finir, nous avons utilisé la fonction **map.on** avec comme paramètre "pointermove" qui permet que lorsque le curseur est sur un layer celui-ci lance la fonction.

Grâce à cette fonction nous pouvons prendre l'id correspondant à chaque layer présent sur la map.

Les requêtes à la base

Voici les requêtes que nous avons utilisées dans les fonctions décrites ci-dessus :

```
'SELECT `idEvenement`, `nomEvenement`, `descriptionEvenement`,`dateEvent`, `latitude`,`longitude`
    FROM evenement
    JOIN position ON `evenement`.`position_idPosition` = `position`.`idPosition`';
```

Figure 9 : Requete getEvents()

Cette requête est utilisée dans la fonction **getEvents().**

```
'SELECT `idEvenement`,`nomEvenement`,`descriptionEvenement`,`dateEvent`,`latitude`,`longitude`,`nomCategorie`
FROM evenement
JOIN position ON `evenement`.`position_idPosition` = `position`.`idPosition`
JOIN categorie ON `evenement`.`categorie_idCategorie` = `categorie`.`idCategorie`
WHERE idEvenement = :id';
```

Figure 10 Requete getEventByID()

Celle-ci est utilisée dans la fonction **getEventByID().**

Rapport de tests :

Plan de test

Voir annexe.

Rapport de test

Nous n'avons malheureusement pas pu effectuer de phases de test sur la base de notre plan de test en raison des mesures prises par l'état.

Conclusion :

La planification du travail s'est bien déroulé en suivant les tâches et le journal de bord malgré les quelques retards ou avances. Le travail en équipe et sur git s'est déroulé sans accroc mais le travail à effectuer a cependant dû être raccourci. C'est pour cela, par exemple, que nous n'avons pas pu implémenter les images liées aux événements et aux catégories.

Au final, notre projet remplit parfaitement le cahier des charges et est déjà fonctionnel. Il reste néanmoins quelques améliorations qui peuvent être faites comme une gestion d'images dans les événements, un système de login et d'utilisateurs pour gérer ses propres événements ou encore une meilleure adaptation aux formats mobiles pour le client WEB.

Bilan personnel (Jorge Machado)

Ce projet m'a permis d'améliorer mes compétences de travail en groupe mais aussi mes compétences javascript et php. J'ai appris comment utiliser openstreetmap ainsi que openlayers, qui sont tous deux sous Javascript.

Bilan personnel (Walter Jauch)

La connexion à une base de données depuis une application C# est une chose que je n'avais jamais eu l'opportunité de faire. Ce projet m'a aidé à avoir une meilleure idée de comment il faut s'y prendre. Nous avons également dû utiliser une autre méthodologie que le SCRUM, ce qui nous a tous permis de travailler d'une manière à laquelle nous n'étiez pas habitués.

Bilan personnel (Johnny Marinho)

Ce projet m'a permis de pratique en profondeur l'utilisation du framework openlayers, j'ai également pu anticiper des problèmes que je peux avoir dans un projet futur. Il m'a également permis de travailler le php avant le TPI.

Bilan personnel (Nelson Jeanrenaud)

Grâce à ce projet, j'ai pu utiliser une base de donnée avec une application C# pour la première fois. J'ai aussi pu découvrir un package de cartographie C# qui est GreatMap. D'un point de vue moins technique, j'ai pu avoir une meilleure idée de comment se déroule un projet en groupe et la rédaction d'une documentation typé TPI.

Bilan personnel (André Gouveia)

Grâce à ce projet j'ai appris à comment faire une connexion à une base de donnée depuis une application C# pour la première fois. J'ai aussi pu découvrir les étapes que j'aurai à faire durant le TPI et avoir déjà de l'expérience pour le TPI. J'ai pu voire les difficultés que j'ai eu durant ce projet pour ne pas le reproduire durant le TPI.