YOUTH CENTER
Livret Technique et Fonctionnel

Démo sur le site :

https://obiwan.univ-brest.fr/~e22110297/

rédigé par Elham Jaber
L3-informatique
Madame Marc - RESAWEB-2025
@UBO-BREST

Aperçu du Projet et des Technologies Utilisées

YOUTH CENTER *est une association de jeunesse qui offre aux jeunes un espace pour développer et exprimer leurs talents, tout en proposant des tarifs accessibles à tous.*

Dans le cadre de ce projet, nous utilisons les technologies suivantes :

- ***CodeIgniter 4*** *comme framework,*
- ***phpMyAdmin*** *pour la gestion de la base de données,*
- ***Visual Studio Code (VS Code)*** *comme environnement de développement (IDE).*
- ***SQL :Créer la base de données (tables, relations, clés primaires, clés étrangères…),Insérer des données (INSERT),Modifier des données (UPDATE),Supprimer des données (DELETE),Consulter / récupérer des données (SELECT),Assurer l'organisation et la structure des informations ,Communiquer entre CodeIgniter 4 et la base de données.***

Méthodologie utilisée : la méthodologie Agile Scrum

*Pour la gestion du projet, nous avons adopté la **méthodologie Agile Scrum**, un cadre de travail itératif et collaboratif permettant de développer un produit de manière progressive. Scrum repose sur des cycles courts appelés **sprints**, généralement de 1 à 2 semaines, au cours desquels l'équipe planifie, développe, teste et livre un incrément fonctionnel du produit.*

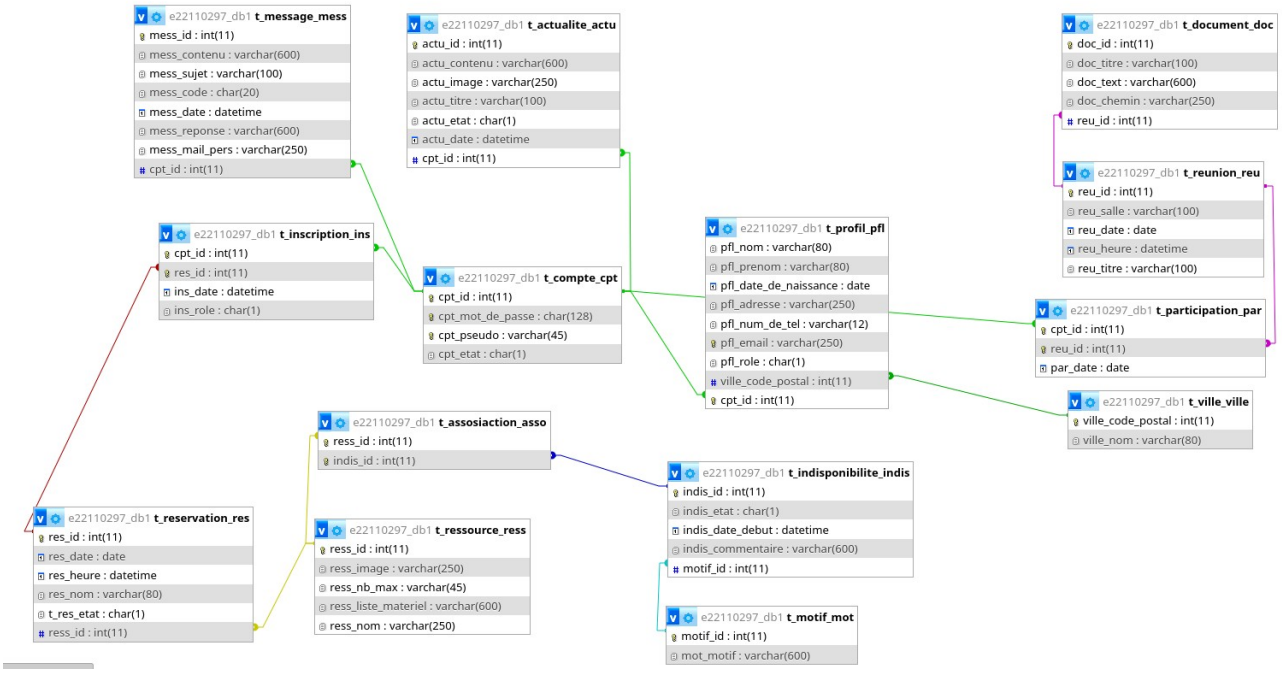
Cette méthodologie facilite :

- *l'**adaptation rapide** face aux changements,*
- *la **communication régulière** avec le client et au sein de l'équipe,*
- *la **livraison fréquente** de fonctionnalités utilisables,*
- *et l'**amélioration continue** du processus de travail.*

Sauvegarder la version

il faut penser à sauverger la version sur la mchine locale , vador , sur ler serveur obiwan et sur gitlab et créer aussi des branche pour chaque version

Avant tout, voici le diagramme relationnel sur lequel nous nous appuyons pour modéliser la base de données. Il permet de structurer les tables et leurs relations afin d'assurer le bon fonctionnement des requêtes et de l'ensemble du système



Scénario de la Version 1 du projet RESAWEB

En premier temps il est demandé de faire la partie publique pour le visiteur , dans la barre de navigation , il y a les boutons ; accueil , suivre ma demande , contact : pour poser une question , et bouton connexion qui ne marche que si le visiteur est inscrit à l'association , sur la page d'accueil, il y a le fil d'actualités récentes, en cliquant sur le bouton Contact , un formulaire de saisi à remplir entièrement avec des conditions à respecter et des message d'erreur

en cliquant sur suivre ma demande le visiteur doit impérativement entrer son code déjà reçu lors de validation de son message ; sir le code est bon il se permet de voir l'état de sa demande sinon un message d'erreur s'affichera ; pas que ; quand un administrateur se connecte , il a en une rebrique CONTACT dans laquelle

il peut visualiser tous les messages ; parmi ces message il y a ceux qui n'ont pas de réponse donc il pourra répondre grace au bouton répondre et au formulaire quo s'affichera ensuite

Attention : lors de validation de code pour visualiser la suivi de message , le code ne doit pas s'afficher !

Tout ces informations sont mentionnées en détails dans le BACKLOG produit DE MADAME MARC .

NOTE :

1. Contrôleur Contact.php :

- index() : affiche le formulaire.

- envoi() : vérifie si la requête est POST, valide les données, renvoie le formulaire en cas d'erreur, redirige vers suivi_message si succès.

- suivi_message() : affiche la page de confirmation.

2. Validation :

- Utilisation de $this→validate().

- Règles : required, min_length, max_length, valid_email.

- En cas d'échec : view('form', ['validation' => $this→validator]).

3. Formulaire

form.php :

- Utilisation de form_open('contact/envoi') et csrf_field().

- Les champs doivent avoir des name="" (sujet, email, message).

- Bouton Valider non désactivé.- Affichage des erreurs : $validation→listErrors().

4. Routes : $routes->get('contact', 'Contact::index');

$routes->post('contact/envoi', 'Contact::envoi');

$routes->get('suivi_message', 'Contact::suivi_message');

5. Points essentiels :

- form_open doit pointer vers contact/envoi.

- Le contrôleur ne doit pas contenir l'ancien code du TP compte.- La validation doit renvoyer $validation à la vue.

→ **getResultArray() → plusieurs lignes**

- Elle **récupère toutes les lignes** du résultat d'une requête SQL.
- Chaque ligne est renvoyée sous forme de **tableau associatif** (clé = nom de la colonne).

→ **getRow() → une seule ligne**

- Elle récupère **juste une seule ligne** du résultat (souvent la première).
- Tu l'utilises quand ta requête renvoie **un seul enregistrement** (par exemple un utilisateur, un article précis, etc.).

→ **CodeIgniter** est un **framework PHP MVC**.

MVC = **Modèle – Vue – Contrôleur,** Ce modèle sépare ton code en 3 couches, pour qu'il soit plus organisé et facile à maintenir

C'est lui que CodeIgniter exécute en premier quand tu visites une page.
C'est le "chef d'orchestre".

Le Modèle

➡ Il s'occupe **des requêtes SQL et de la base de données**.

La Vue

➡ C'est ce que voit l'utilisateur — du HTML, du CSS, parfois du JavaScript.

Scénario de la V2
SPRINT 2 SEULEMENT

à ce stade , l(a) product owner demande d'ajouter des fonctionnalités aux deux espaces : membre et administrateur , selon la sprint 2 les deux utilisateurs vont pouvoir acces à une rubrique :
pour consulter les réservations pour un créneau précis , triés par ressource réservée ,
et une deuxième pour consulter les ressources réservables pour l'admin seule ;

-Il faut faire attention à plusieurs points essentiels comme : la supression d'une ressource faut qu'elle soit précédée d'une supression de toutes reservations , inscriptions et indispoiblité liées à cette ressource ,

-le bilan d'un réservation doit etre afficher après la séance ,

un compte invité pourra se connecter pour voir ses réservations seulement , il ne disposera pas de la meme espace que celui d'un membre ou d'un admin ,

- l'admin pourra faire l'ajout ou la supression d'une ressource…

des users stories ont été rédigées par madame Marc pour tout détailler !

Activités 4

→ Préparez plusieurs vues pour votre base de données (au moins une vue par étudiant).

Dans db.Model :

```
  public function get_reservation($date)
{
    // Vérifier le nombre de réservations
    $nbsql = "SELECT nb_reservations(?) AS nb;";//fonction
    $nbRow = $this->db->query($nbsql, [$date])->getRow();

    if (!$nbRow || $nbRow->nb == 0) {
        return []; //  retourner un tableau vide
    }

    // Requête principale + utiliser une vue
    $sql = "
       SELECT * FROM v_reservations_details //vue
        WHERE DATE(res_date) = ?
        ORDER BY ress_nom;
    ";

    return $this->db->query($sql, [$date])->getResult();

}
```

-j'ai préparé une vue où j'ai mis toutes les informations et les détails nécessaires pour afficher une réservation à une date précise triée par le nom de la ressource , cette vue m'a servis à ecrire juste SELECT FROM v_reservations_détails au lieu d'écrire une dizaine de lignes dans le db.Model.

→ Préparez plusieurs fonctions pour votre base de données (au moins une fonction par étudiant·e qui sera à utiliser dans le code PHP de l'application Web).

-J'ai préparé une fonction : nb_reservations('jour' Date)
qui compte le nombre réservation à venir pour un utilisateur qui se connecte ; si cette fonction renvoie un nombre plus grand que 0 j'applique la requete qui récupère les réservations sinon j'affiche directement 'aucune réservation à venir'

Préparez plusieurs procédures pour votre base de données (au moins une procédure→par étudiant·qui sera à utiliser dans le code PHP de l'application Web).

Une procédure `ressource_exister`(IN nom_ress VARCHAR(255)) ;
dans db,model :

```
public function ressourceExister($nom)
{
    $query = $this->db->query("CALL ressource_exister(?)", [$nom]);
    $result = $query->getRow();



    return ($result->total > 0);
}
```

qui lors d'un ajout d'une ressource par un administrateur , regarde sin une autre ressource existe de meme nom , si oui elle affiche un avertissemnt mais ça ne joue pas sur l'ajout de la ressource , donc la ressource est ajoutée mais accompagnée d'un warning : 'une ressource de meme nom existe déjà'

→ Préparez plusieurs triggers

j'ai fait deux triggers

1-TRIGGER `trg_delete_ressource`
le premier supprime toutes réservation, inscription , indisponibilitées d'une ressource avant la suppression de la ressource donc il suffit de clicquez sur le bouton supprimer → ok→ la fonction de db model pourtant une seule requete de delete , appelée par le controller et le trigger derrière les coulisses s'en occupe de tout pour effectuer cette suppression

2-TRIGGER `trg_set_image_before_insert`
le deuxième qui lors d'un ajout de la ressource , l'admin ne se charge pas de la photo de cette ressource une photo actuellement est par défaut , donc en

appuyant sur ajouter , le trigger ajoutera le chemin_ templateprivé/img/default,png dans la colonne ress_image , celui si est récupérée par le Model , qui par la suite appelée par le controller qui s'occupe de la validation de formulaire d'ajout et ajoute dans data[res_img] le chemin et l'ajoute dans la colonne de l'image de la vue_ .

Tester les liens

après avoir se déconnecter_ ; il n'est pas possible d'accéder à toute espace privée sans la saisie de mot de passe et de logins correctes ,
si ce n'est pas le cas_ ; un problème de sécurité se présente_ !

https://obiwan.univ-brest.fr/~e22110297/index.php/compte/lister

https://obiwan.univ-brest.fr/~e22110297/index.php/compte/creer

https://obiwan.univ-brest.fr/~e22110297/index.php/compte/connecter

https://obiwan.univ-brest.fr/~e22110297/index.php/compte/deconnecter

https://obiwan.univ-brest.fr/~e22110297/index.php/compte/index

https://obiwan.univ-brest.fr/~e22110297/index.php/compte/index2

https://obiwan.univ-brest.fr/~e22110297/index.php/compte/index3

https://obiwan.univ-brest.fr/~e22110297/index.php/compte/lister2

https://obiwan.univ-brest.fr/~e22110297/index.php/compte/afficher_formulaire

https://obiwan.univ-brest.fr/~e22110297/index.php/compte/afficherlistemessage

https://obiwan.univ-brest.fr/~e22110297/index.php/compte/affichercreneau

https://obiwan.univ-brest.fr/~e22110297/index.php/compte/affichercreneau1

https://obiwan.univ-brest.fr/~e22110297/index.php/compte/reserver_action

https://obiwan.univ-brest.fr/~e22110297/index.php/compte/reserver_action1

https://obiwan.univ-brest.fr/~e22110297/index.php/compte/afficher_ressource

https://obiwan.univ-brest.fr/~e22110297/index.php/compte/formulaire_ress

https://obiwan.univ-brest.fr/~e22110297/index.php/contact/index

https://obiwan.univ-brest.fr/~e22110297/index.php/contact/index1

https://obiwan.univ-brest.fr/~e22110297/index.php/contact/afficher_message2

https://obiwan.univ-brest.fr/~e22110297/index.php/delete/delete_ress

Démonstration

réservation passée

30/11/2025 état A
27/11/2025 état A
10/10/2025 état D

réservation à venir triés par ressource :15/12/2025

Data comptes :

Adhérent	code	Id
elham.cpt	elhamjaber	251
marie.cpt	marie123	252
joseph.cpt:	joseph123	253
pape.cpt	pape1233	254

momo.cpt	momo.cpt	260
invité	code	détails
invite25	invite25	reservation ok
Invite26	invite26	pas de reservation

compte	code	id	détails
principal	princ25*!ASER	261	Res ok
jaber.cpt	jaber.cpt	240	Res ok
rebecca.cpt	rebecca.cpt	262	Res ok
fadi.cpt	fadi123	263	
bjorc	amandine	264	

MERCI

fin de projet