

PROJET PROFESSIONNEL 2022



**Le sel de
la vie**



TABLE DES MATIERES

Compétences du référentiel couvertes par le projet :.....	3
Résumé du projet :.....	4
Spécifications fonctionnelles.....	5
Description des fonctionnalités.....	6
1.Authentification/création de compte.....	6
2.Catalogue Articles/Actions et filtre.....	6
3.Fiche Article.....	6
4.Profil utilisateur.....	6
5.Contact Association et liens vers sites administratifs.....	6
6.Back office.....	7
a.Partie modérateurs - fonctionnalité création de formulaires.....	7
b.Partie modérateurs - fonctionnalité création d'article.....	8
c.Partie Administrateur	8
Spécifications techniques.....	9
Réalisations.....	11
1.Charte graphique.....	11
2.Maquette.....	11
3.Conception de la base de données.....	12
4.Extraits de code significatifs - Générateur de formulaires :.....	13
a.élaboration de la base de données :.....	13
b.interface en interaction avec l'utilisateur.....	14
c.traitement de la donnée.....	19
5.Veille sur les vulnérabilités de sécurité.....	22
6.Recherche effectuées à partir d'un site anglophone.....	24

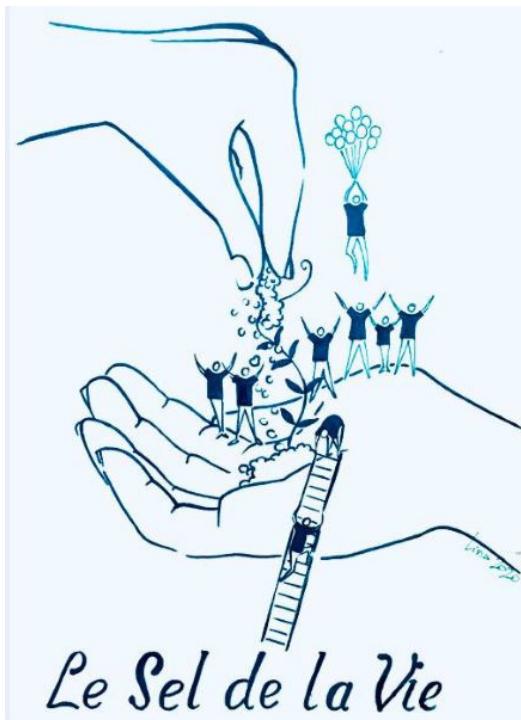
PRÉAMBULE

Le sel de la vie est un collectif de bénévoles issus de différents secteurs d'activités, se mobilisent pour soutenir et accompagner des familles, enfants, et étudiants marseillais autour de cinq axes majeurs : éducation, formation, culture, sports et loisirs.

Dans le cadre de la formation au sein de LaPlateforme_, nous avons rencontré une association marseillaise nommée Le Sel de La Vie.



LES FONCTIONNALITÉS



Les fonctionnalités demandés:

Le site Le Sel de la Vie a pour objectif de construire un site interactif (responsif):

- La genèse de l'association (histoire, photos, films, articles et liens)
- Les activités proposées éducatives, de formation, sports et loisirs et accompagnement adultes et jeunes: inscription en ligne, dépôt des documents (formats différents) générés en pdf
- Portail d'entrée vers les sites en lien avec les démarches des usages (CAF, Sécurité Sociale, Pole Emploi, etc): permettant un dépôt sécurisé des documents et gestion par ceux ci

D'autres fonctionnalités complémentairent était demandais tel que la création d'un coffre fort et un outil de génération de formulaire .

Nous avons convenue que si nos compétence était en cohesion avec la RGPD nous le feront auxquels cas les dites fonctionnalités ne seront pas présentes

SPÉCIFICATIONS FONCTIONNELLES

Description du contexte :

Aucun site n'existait auparavant, seule une page Facebook et un compte linkedin permettait à l'association de communiquer avec son public. Nous avons donc convenu lors d'un premier entretiens des différents besoins. Puis à chaque étape du développement, nous avons essayé de fonctionner en méthode agile et d'inclure les acteurs associatifs à notre progression.

Ils ont ainsi validé progressivement et cela nous à permis de renforcer notre partenariat.
Cible adressée par le site internet :

Le site a pour cible le public marseillais et en particuliers les populations défavorisées. Il sera réalisé en français et ce dernier devra être accessible sur différents supports, à savoir mobile et ordinateur.

Arborescence du site :

L'arborescence du site se décline comme suit :

- Page d'accueil qui mets en évidence les articles/actions de l'association
- Page connexion
- Page inscription
- Page article contenant le descriptif et le cas échéant le formulaire d'inscription
- Page profil

Une partie back-office est également prévue afin de permettre la gestion du site, elle a deux niveaux d'accès admin et modérateurs.

Les modérateurs ont comme accès supplémentaire :

- Page création de formulaire
- Page création d'article

Les admins peuvent supprimer les articles, octroyer des droits supplémentaires aux utilisateurs, validation des articles...

SPÉCIFICATIONS FONCTIONNELLES

L'arborescence du site se décline comme suit :

Page d'accueil qui mets en évidence les articles/actions de l'association

Page connexion

Page inscription

Page article contenant le descriptif et le cas échéant le formulaire d'inscription

Page profil

Une partie back-office est également prévue afin de permettre la gestion du site, elle a deux niveaux d'accès admin et modérateurs.

Les modérateurs ont comme accès supplémentaire :

Page création de formulaire

Page création d'article

Les admins ont la main sur tout le site, ils peuvent supprimer les articles, octroyer des droits supplémentaires aux utilisateurs, validation des articles...

Description des fonctionnalités

1. Authentification/création de compte

Il a été convenu de pouvoir s'inscrire et se connecter de manière classique via un formulaire d'inscription/connexion pour les utilisateurs du site. Dans un premier temps, elle ne sert pas fondamentalement, puisque les formulaires du site ont pour vocation d'être imprimés et non enregistrés en base de donnée. Mais, le client a demandé cette fonctionnalité en vue d'un coffre-fort pour héberger les fichiers individuels.

2. Catalogue Articles/Actions et filtre

La page d'accueil doit permettre d'afficher l'ensemble des articles disponibles. Cet affichage devra comprendre la photo de l'article, son titre ainsi que le descriptif réduit à 150 caractères. Un filtre doit être implémenté afin de trier les articles en fonction de leur catégorie.

3. Fiche Article

L'utilisateur devra être en mesure d'accéder à une fiche article. Cette dernière devra comprendre:

le titre

la photo

le descriptif complet

le cas échéant un formulaire personnalisé pour l'article permettant l'enregistrement d'un PDF contenant les informations et pièces renseignées dans le dit formulaire

SPÉCIFICATIONS FONCTIONNELLES

4. Profil utilisateur

L'utilisateur aura accès à un espace dans lequel il lui sera possible de consulter et modifier ses informations. A savoir son nom, son prénom, son adresse mail, son mot de passe mais également son adresse postale.

5. Contact Association et liens vers sites administratifs

Le client nous a demandé que les utilisateurs puissent avoir facilement accès aux différents sites des acteurs sociaux primordiaux ainsi qu'aux instances éducatives majeurs de Marseille et la région. Il a souhaité aussi que l'utilisateur puisse simplement trouver comment les contacter par téléphone ou e-mail

6. Back office

Le back-office se divise en deux sous-partie :

a. Partie modérateurs – fonctionnalité création de formulaires

À la demande du client, nous avons dû élaborer un générateur de formulaire permettant aux modérateurs de créer des formulaires personnalisés sans connaissances en informatiques. Ils ont ainsi la possibilité de créer :

des champs de texte

des champs de type textarea pour les commentaires et notes

des champs de type select pour les choix parmi une liste

des champs de type checkbox pour les choix multiples

des champs de type radio pour les choix uniques type oui / non

des champs de type file pour l'upload des pièces demandées

La complexité de cette fonctionnalité ne nous a pas apparue à la demande du client mais lors de la phase complète de développement.

Le modérateur a à sa disposition une prévisualisation du formulaire qu'il créé au fur et à mesure que celui-ci apparaît sur l'écran. (ci-dessous un aperçu)

b. Partie modérateurs – fonctionnalité création d'article

Il peut évidemment créer des articles et c'est sa fonctionnalité primordiale. Les articles ont des titres, des photos d'illustration, un descriptif. Le modérateur doit choisir parmi les catégories disponibles et peut choisir un formulaire à ajouter à son article. Il a encore à sa disposition une prévisualisation de sa création. Une fois l'évènement créé, il peut l'éditer via le listing des articles qui lui donnera une option supplémentaire lorsqu'il est connecté en modérateur. Ci-dessous une illustration

Creation d'article

Nom de l'article :
Inscription études du soir éc

 Ajouter une photo



Description de l'article :
Nouvelle campagne d'inscription pour la rentrée 2022-2023 pour l'école Canet Larousse

Catégorie :
scolaire

Choisir le formulaire :
Soutien Scolaire

Enregistrer l'article

Prévisualisation

Inscription études du soir école Canet Larousse



Nouvelle campagne d'inscription pour la rentrée 2022-2023 pour l'école Canet Larousse

Nom Prénom (parent 1) :

 Ajouter une photo

Carte d'identité (parent 1)

Nom Prénom (parent 2) :

 Ajouter une photo

Carte d'identité (parent 2)

Nom Prénom (enfant) :

Carte d'identité (parent 1)

champ de texte

Choisissez le label de votre champ
(Ex : Veuillez entrer votre nom)
Nom Prénom (parent 2) :

Fichier

Choisissez le champ de référence (Veuillez ajouter votre carte d'identité)
Carte d'identité (parent 2) :

champ de texte

Choisissez le label de votre champ
(Ex : Veuillez entrer votre nom)
Nom Prénom (enfant) :

Liste déroulante

Choisissez le label de votre champ (ex : En quelle classe est votre enfant ?)
En quelle classe est votre enfant ?

Prévisualisation

Soutien Scolaire

Nom Prénom (parent 1) :

 Carte d'identité (parent 1)

Nom Prénom (parent 2) :

 Carte d'identité (parent 2)

Nom Prénom (enfant) :

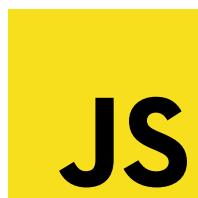
En quelle classe est votre enfant ?
Veuillez choisir parmi la liste

Informations complémentaires à nous transmettre

Quel matériel avez-vous ?

Stylos multicolores Cahier gros carreaux Règle Colle

LES OUTILS & LES LANGAGES



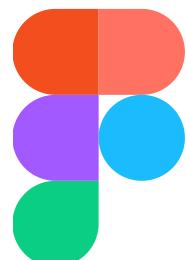
HTML



CSS



miro



draw.io

Choix techniques et environnement de travail

Technologies utilisées pour la partie back-end :

- PHP
- Base de données SQL

Technologies utilisées pour la partie front-end:

- HTML et CSS pour la structure et le visuel du site
- Javascript pour les interactions et modifications DOM dynamiques.

L'environnement de développement est le suivant:

- Éditeur de code: VSCode
- Outil de versioning: GIT, Github.
- Maquettage: Figma

Du point de vue de l'organisation, nous avons utilisé Miro.com afin de simuler un product backlog à l'image des tableaux remplis de post-it, pour mettre en évidence le travail en cours, à faire, à tester. Nous nous sommes essayé à la méthode agile dans l'optique d'en acquérir les bases pour les mettre en pratique plus facilement en entreprise.



ARCHITECTURE

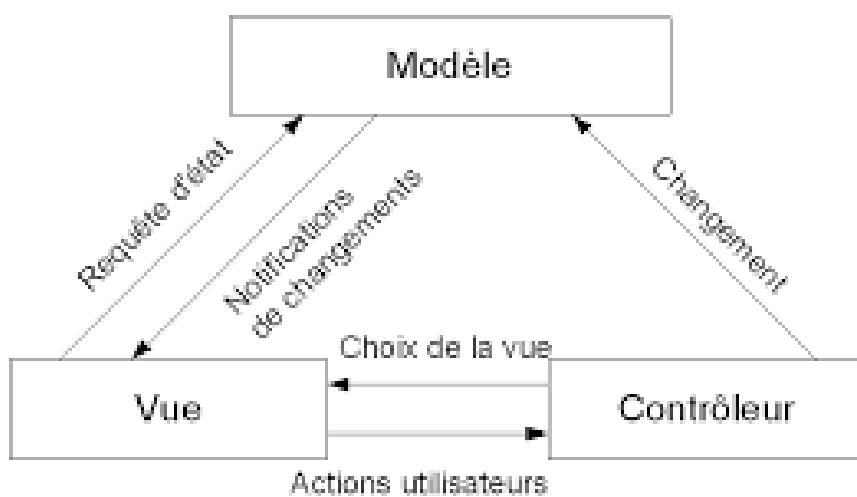
Architecture du projet

Nous avons voulu nous inspirer d'un design pattern de type MVC (Model-View-Controller) et ainsi avons expérimenté en créant notre cadre de travail. L'architecture MVC est l'une des architectures les plus utilisées pour les applications Web, elle se compose de 3 aspects:

- View : aspect visuel du site en interaction directe avec l'utilisateur et présentant les jeux de données mis à sa disposition
- Model : aspect qui permet d'accéder, écrire, supprimer, altérer les informations dans la base de données et de les renvoyer ordonnées pour ensuite être traitées par le controller.
- Controller: composant responsable des prises de décisions, gère la logique du code, il est l'intermédiaire entre le modèle et la vue.

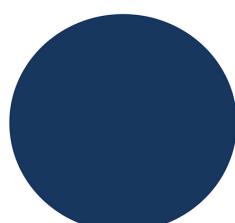
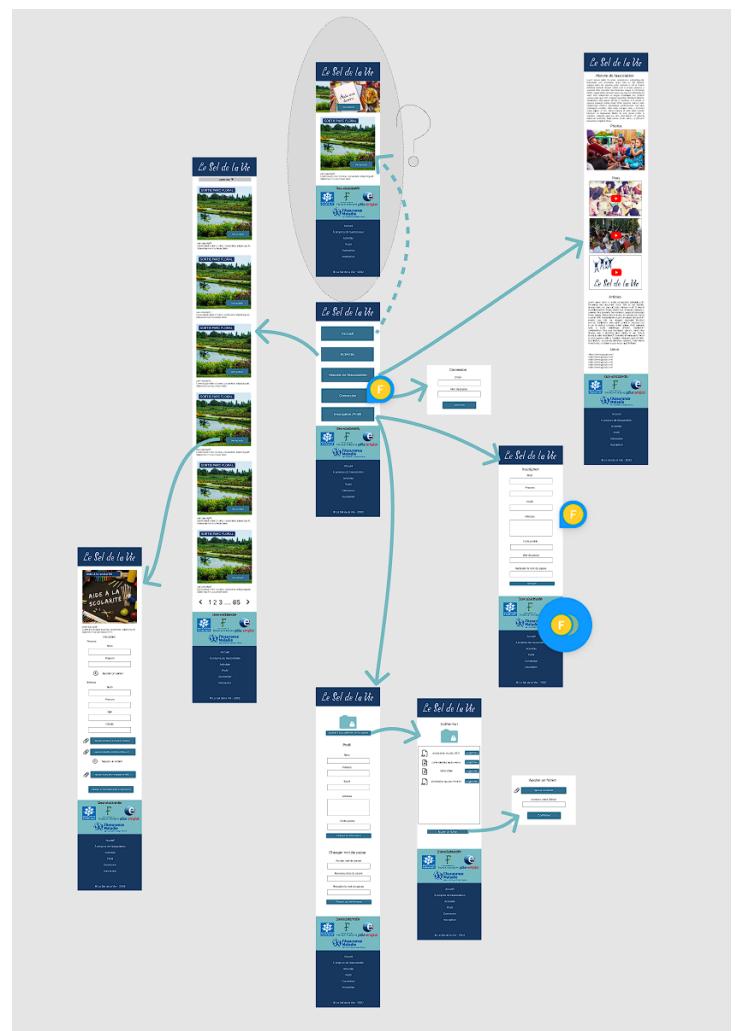
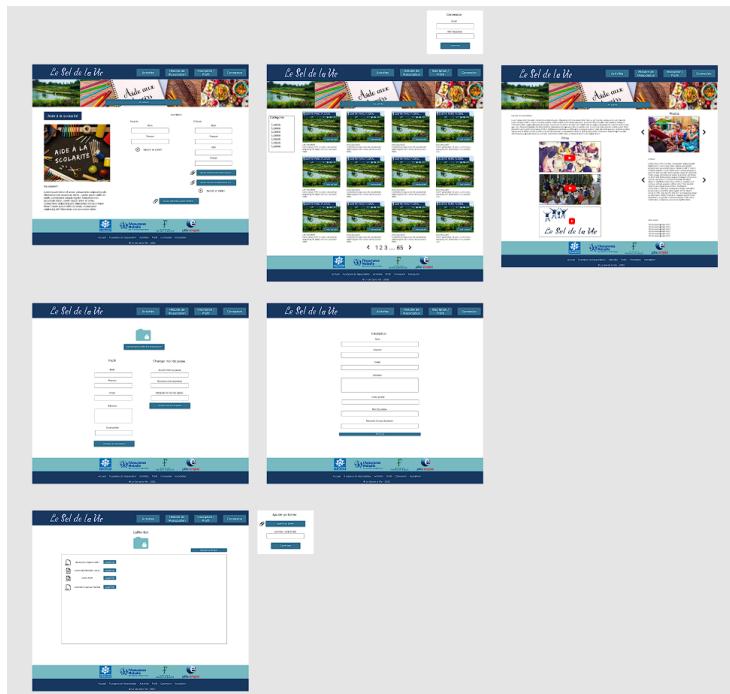
Un débat est apparu dans notre équipe sur l'utilité de coder nos controller en tant que classe avec des méthodes de réactions aux différents événements. Un de nos membres avait pour conception des choses qu'un controller fonctionne comme la moelle épinière chez l'humain, servant pour les réactions réflexes (`if(isset($_POST['submit']))` par exemple).

Au fur et à mesure des débats, nous avons compris que dans la logique de progrès et l'implémentation de routeur, avoir des controller codé en classe permet d'utiliser de manière ponctuelle les méthodes en fonction des routes invoquées.

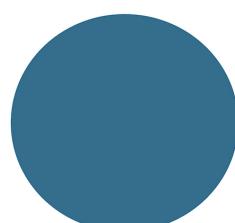


LE MAQUETTAGE

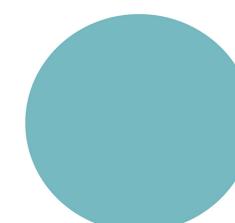
Comme dit à la page précédente nous avons utilisés figma pour le maquettage du site il ya deux version la version laptop/tablette et la version mobile
La charte graphique est composé de trois couleurs précises



#18375F



#346E8C



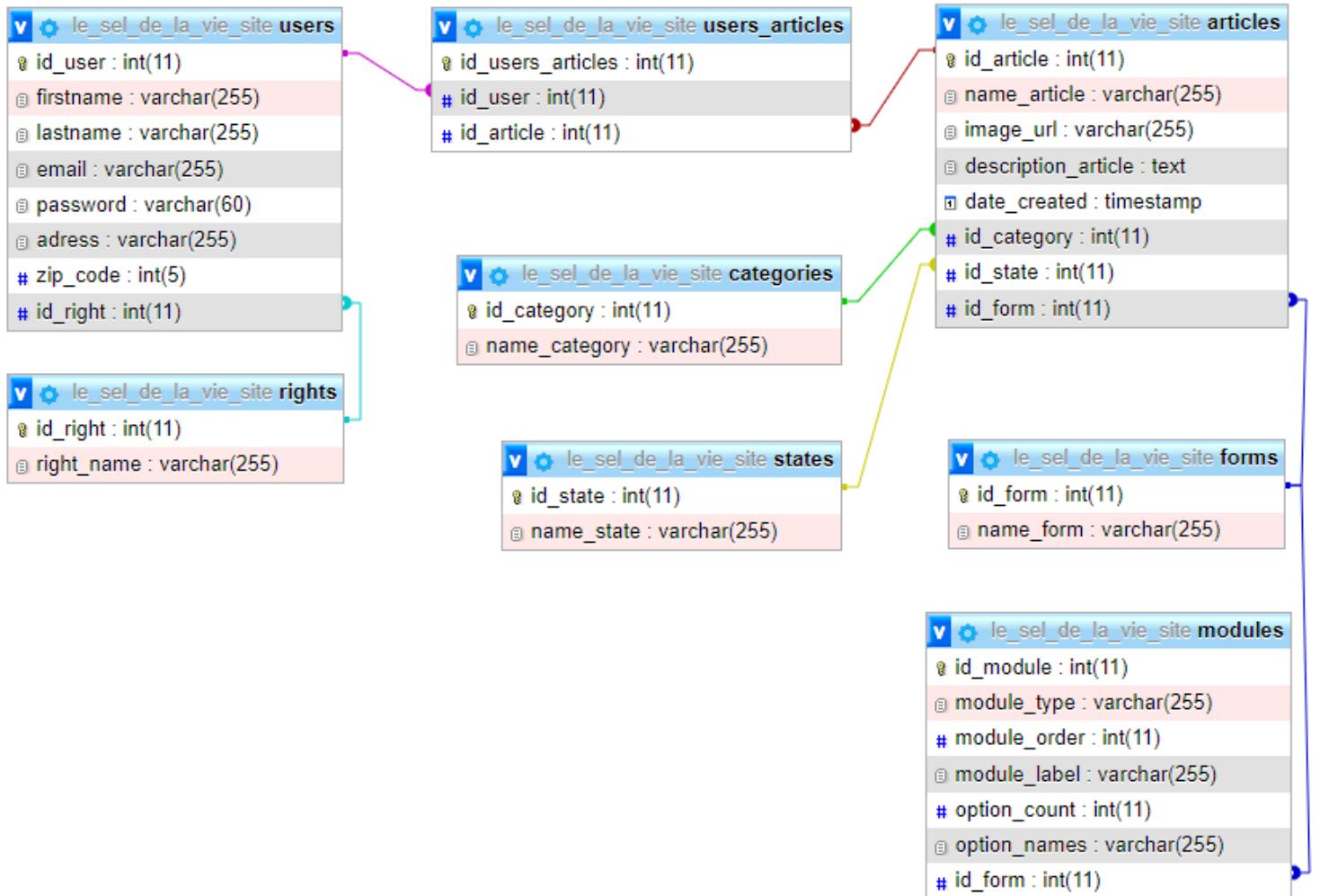
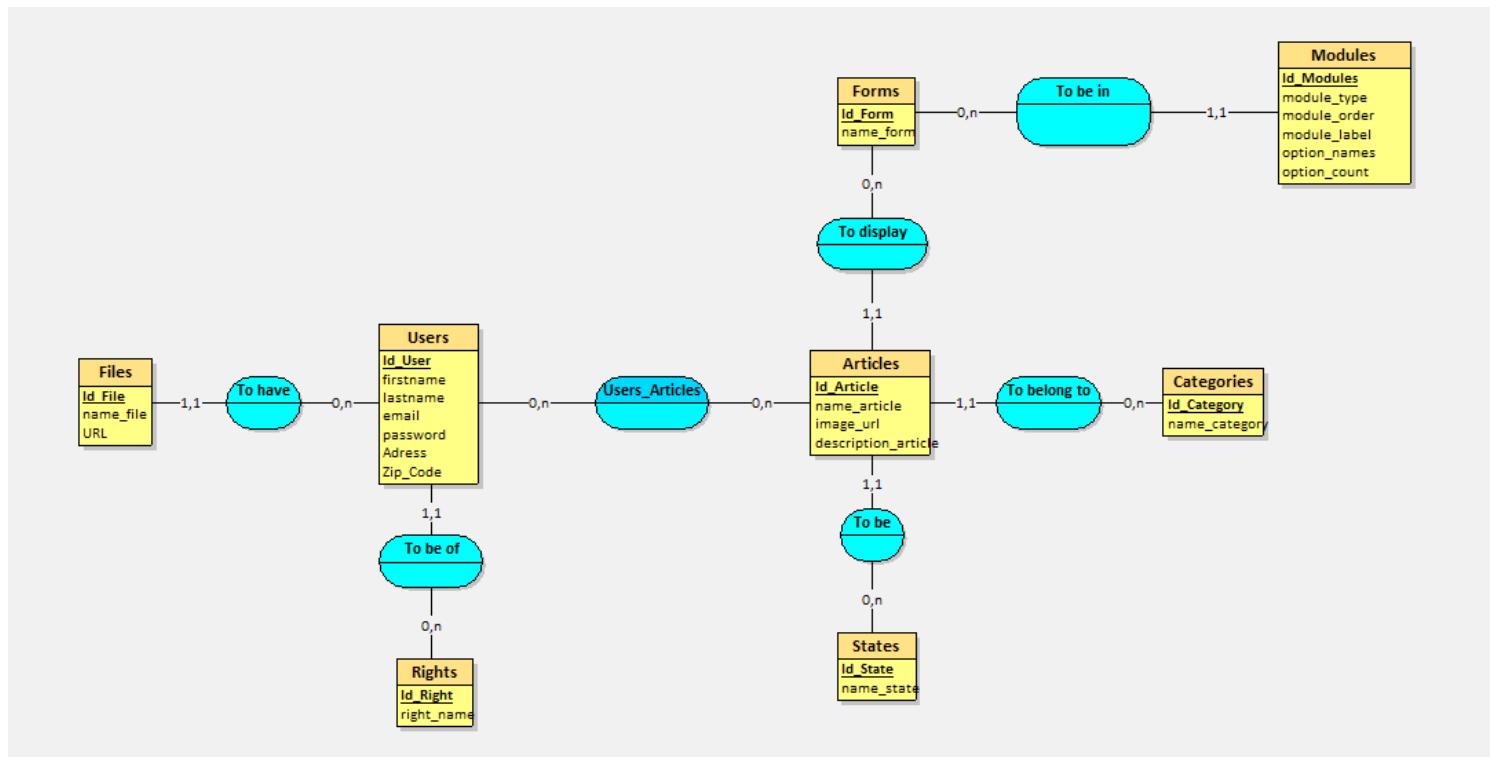
#77B9C1

CONCEPTION DE LA BASE DE DONNÉE

Après discussion avec les membres de l'association et nous avons élaboré les différents modèles :

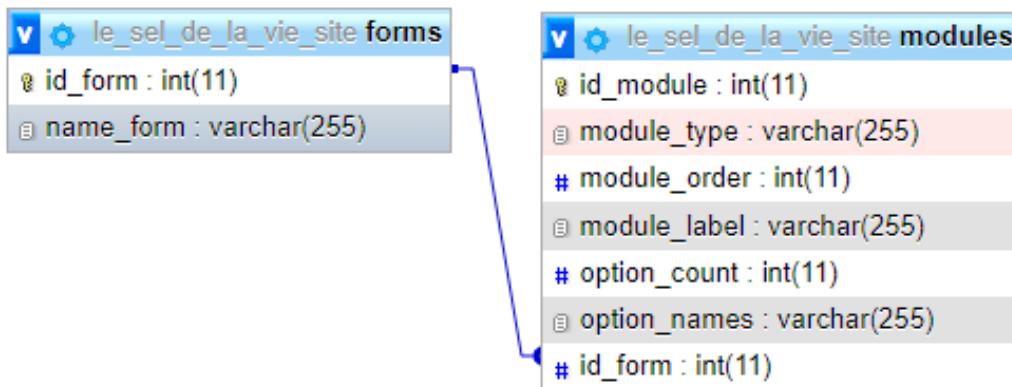
Les trois tables principales de la base de données sont :

- Users, qui contient les données relatives aux utilisateurs (nom, prénom, adresse, email, password(chiffré), code postal et une clé étrangère id_right en référence à une sous-table rights contenant les différents niveaux de droits d'utilisateur (basique, modérateur et administrateur)
- Articles, qui contient les données relatives aux articles (nom, adresse de l'image d'illustration, la description, la date de création ainsi que trois clés étrangères :
 - id_catégorie référence la table categories pour y ajouter par liaison un nom de catégorie qui servira au tri sur la page d'affichage des articles
 - id_state référence la table states pour y ajouter par liaison un état que l'administrateur transformera entre en cours de validation et validé
 - id_form qui référence, le cas échéant, le formulaire qui fait partie de l'article
- Forms, qui contient les données relatives aux formulaire (id, nom de formulaire), c'est la porte d'entrée vers les modules qui sont enregistrés au moment de la création de formulaire et qui portent tous l'id du formulaire auquel ils sont attachés



4. Extraits de code significatifs - Générateur de formulaires :

Pour implémenter cette fonctionnalité à la demande de notre client, nous avons abordé le développement sous trois angles :
une partie élaboration de la base de donnée relative à cette fonctionnalité
une partie interface en interaction avec l'utilisateur
une partie traitement de la donnée
a. élaboration de la base de données :



Pour concevoir notre fonctionnalité, il a été nécessaire de bien pondérer ces deux tables. La table Forms est la porte d'entrée du formulaire en vue de la récupération des données. Simple d'accès elle ne porte que le nom et l'id du formulaire.

La table module se devait d'être modulable, puisque nous devions mettre à disposition des modérateurs du site 6 type d'input de formulaire ainsi que les labels s'y afférant :

- les input de type text
- les input de type textarea
- les input de type file
- les input de type select
- les input de type checkbox
- les input de type radio

Il nous a vite apparu que les inputs se divisaient en deux catégories, ceux n'ayant besoin que du type de module et du label et ceux ayant un certain nombre d'options et leur nom .

Nous avons ainsi fait le choix de rendre « nullable » les colonnes « option_count » et « option_names » afin d'avoir une table disponible pour l'intégralité de nos inputs.

b.interface en interaction avec l'utilisateur

Par soucis de simplicité, nous avons choisi de ne pas permettre aux modérateurs d'altérer les formulaires une fois créés. Ci-dessous, une capture d'écran de la fonctionnalité lors de l'arrivée sur la page de création de formulaire.

Attention les formulaires ne sont plus modifiables une fois enregistrés !

Nom du formulaire

Prévisualisation

+

Enregistrer le formulaire

Le code relatif à cette page est très simple et le but complet de notre fonctionnalité est une interaction directe et dynamique avec le DOM. Ci-dessous le code html de base :

```
<main class="mainFormulaire">
    <h2 class="titreFormulaire">Attention les formulaires ne sont plus modifiables une fois enregistrés !</h2>

    <div id="FormulaireGen">
        <form action="" method="POST" enctype="multipart/form-data">
            <label for="name_form">Nom du formulaire</label>
            <input type="text" name="name_form" id="name_form">
            <hr>
            <input type="submit" value="Enregistrer le formulaire" name="reg_form" id="reg_form">
        </form>
    </div>

    <fieldset id="previewFormulaire">
        <legend>Prévisualisation</legend>
    </fieldset>
</main>
```

Sur la première capture d'écran, vous pouvez voir un « + », un bouton qui a pour but de rajouter un champ à notre formulaire. Ce bouton est rajouté grâce à un script Javascript, que vous pouvez voir ci-dessous :

```
//---- création du bouton d'ajout de champ
const addSelectType = document.createElement('button');
addSelectType.setAttribute('type', 'button');
addSelectType.setAttribute('class', 'addSelectType');

//append de l'image qui sera utilisée pour le bouton
let addSelectTypeImg = document.createElement('img');
addSelectTypeImg.setAttribute('src', './View/Media/plus.svg');
addSelectType.append(addSelectTypeImg);

//ajout au DOM
submitButton.before(addSelectType);

//ajout de l'event listener pour le bouton d'ajout de champ
addSelectType.addEventListener('click', () => {
    selectTypeAppend();
});
```

Comme vous pouvez voir, nous avons ainsi rajouter un écouteur d'événement sur le bouton fraîchement créé, qui réagit au clic. Une fois cliqué, l'affichage devient comme suit :

The screenshot shows a user interface for creating a form. At the top, there is a text input field labeled "Nom du formulaire". Below it is a "fieldset" element containing a "select" dropdown with the placeholder "choisissez une optic". To the left of the dropdown is a blue circular "plus" button with a white "+" sign. To the right is a blue circular "trash" icon. At the bottom of the fieldset is a large blue rounded rectangle button labeled "Enregistrer le formulaire".

Un fieldset apparaît contenant un select qui va permettre à l'utilisateur de choisir le type d'input qu'il veut créer. Il porte aussi un bouton pour ajouter à la suite un nouvel input

(et le cas échéant, entre deux éléments) . Nous avons aussi un bouton supprimer pour enlever un input selon le désir de l'utilisateur. Ci dessous le code relatif à la fonction `selectTypeAppend()` pour l'affichage du select, ainsi que du bouton d'effacement.

```
//---- creation d'une fonction pour ajouter les champs
function selectTypeAppend(targetedDiv = null, beforeOrAfter = null)
{
    //---- creation du select d'element à ajouter au formulaire
    //label
    let selectTypeLabel = document.createElement('label');
    selectTypeLabel.setAttribute('for', 'select_type');
    selectTypeLabel.innerText = 'Choisir un type de champ à ajouter';

    //select
    let selectType = document.createElement('select');
    selectType.setAttribute('name', 'select_type[]');
    selectType.setAttribute('id', 'select_type');

    //function d'append d'option au select
    function appendOption(valueString, selectedBool, disabledBool, innerHtmlString)
    {
        let option = document.createElement('option');
        option.setAttribute('value', valueString);
        option.selected = selectedBool;
        option.disabled = disabledBool;
        option.innerText = innerHtmlString;
        selectType.appendChild(option);
    }

    //utilisation de la fonction d'append
    appendOption('disabled', true, true, 'choisissez une option');
    appendOption('text', false, false, 'champ de texte');
    appendOption('textarea', false, false, 'champ de commentaire');
    appendOption('select', false, false, 'choix parmis une liste');
    appendOption('checkbox', false, false, 'choix multiples, case à cocher');
    appendOption('radio', false, false, 'choix unique (ex: oui/non)');
    appendOption('file', false, false, 'ajout de fichier');
}
```

Nous allons maintenant voir un exemple dans le cas où le modérateur choisi de rajouter un input de type texte. Nous avons rajouté un écouter d'événement sur un changement sur le select permettant le choix. Un switch réagit au changement en fonction de la value de l'élément selectType :

```
//---- ajout d'event listener sur le select qui permet de choisir le type d'élément
selectType.addEventListener('change', () => {

    //switch case pour tester le choix d'option
    switch (selectType.value) {
        {

            //choix champ de texte
            case "text":

                //---- creation de la légende du fieldset
                let fieldsetTextLegend = document.createElement('legend');
                fieldsetTextLegend.innerText = 'champ de texte';

                let firstTextBoxDiv = document.createElement('div');
                //---- création de l'input text pour saisir le champ libellé
                let textDiv = document.createElement('input');
                textDiv.setAttribute('type', 'text');
                textDiv.setAttribute('name', 'text['+ textCount +'][description]');
                textDiv.setAttribute('id', 'text');

                //---- création de l'input hidden pour sauvegarder l'ordre de la div
                let hiddenInput = document.createElement('input');
                hiddenInput.setAttribute('type', 'hidden');

                //label
                let textDivLabel = document.createElement('label');
                textDivLabel.setAttribute('for', 'text');
                textDivLabel.innerText = "Choisissez le label de votre champ"
                textDivLabel.innerHTML += "\n ( Ex : Veuillez entrer votre nom )";
                firstTextBoxDiv.appendChild(textDivLabel);
                firstTextBoxDiv.appendChild(textDiv);

                //ajout au dom
                deleteSelectType.before(fieldsetTextLegend);
                deleteSelectType.before(firstTextBoxDiv);
                firstTextBoxDiv.after(hiddenInput);
        }
    }
}
```

Ci-dessous le visuel lors du choix :

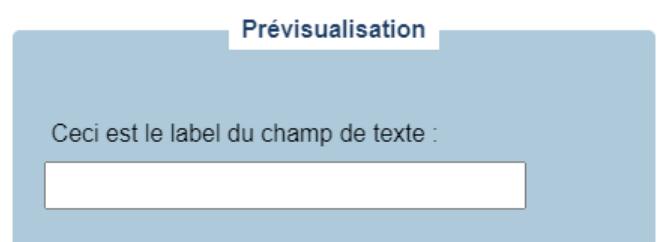
Nom du formulaire

champ de texte

Choisissez le label de votre champ
(Ex : Veuillez entrer votre nom)

st le label du champ de texte |



Javascript rajoute en direct sur l'espace de prévisualisation le label du champ de texte choisi, qui lui, apparaît instantanément lors du choix dans le select.

```
▼<fieldset> flex
  <legend>champ de texte</legend>
  ▼<div> flex
    ▼<label for="text">
      "Choisissez le label de votre champ"
      <br>
      " ( Ex : Veuillez entrer votre nom )"
    </label>
    <input type="text" name="text[0][description]" id="text" style="border: 1px solid black">
  </div>
  <input type="hidden" name="text[0][order]" value="0">
▶<button type="button" class="deleteSelectType">...</button>
▶<button type="button" class="addSelectTypeInsideFieldset">...</button>
</fieldset>
```

Ci dessus, nous pouvons voir les éléments dans le DOM portant les « name » qui seront utilisés lors de l'enregistrement du formulaire et l'envoie des données via la méthode POST.

c. traitement de la donnée

```
if (isset($_POST['reg_form']))
{
  // receive all input values from the form
  $name_form = htmlentities($_POST['name_form']);

  // check if form already exists
  $names = Formulaire::getAllFormsNames();
  $form_exists = false;
  foreach ($names as $name)
  {
    if ($name['name_form'] == $name_form)
    {
      $form_exists = true;
    }
  }
}
```

Lors de la soumission du formulaire de création de formulaire, un controller réagit à linstanciation de la variable \$_POST['reg_form']. Le contrôleur traite les informations et en particulier le nom du formulaire pour vérifier si un formulaire n'existe pas déjà avec le même nom, pour éviter la confusion lors de la création d'article et du choix de formulaire. Si un formulaire ayant le même nom, il change la valeur du booléen \$form_exists de false vers true. Il exploite la méthode statique getAllFormsNames() de la classe abstraite Formulaire ci-dessous :

```
public static function getAllFormsNames()
{
  $sql = 'SELECT name_form FROM forms';

  $forms = self::requestExecute($sql)->fetchAll(PDO::FETCH_ASSOC);

  return $forms;
}
```

Nous avons choisi des classes abstraites afin de ne pas avoir à les instancier et ainsi avons codé nos méthodes en static pour pouvoir les exploiter directement via l'appel de la classe comme ci dessous :

Formulaire::getAllFormsNames()

Si aucun nom de formulaire existe, le controller exécute le code suivant :

```
if form doesn't exist, create it
(!$form_exists)

Formulaire::createForm($name_form);

$idForm = Formulaire::getLastInsertedId();

function moduleCreation($postVariable, $moduleType, $idForm)
{
    if(isset($postVariable))
    {
        foreach($postVariable as $key => $value)
        {
            if(isset($value['description']))
            {
                if ($moduleType == 'text' || $moduleType == 'textarea' || $moduleType == 'file')
                {
                    Formulaire::createModule($moduleType, htmlentities($value['order']), htmlentities($value['description']), $idForm);
                }
                else if ($moduleType == 'select' || $moduleType == 'checkbox' || $moduleType == 'radio')
                {
                    $tempString = '';

                    foreach($value as $key2 => $element)
                    {
                        if (is_int($key2) && $key2 === array_key_last($value))
                        {
                            $tempString .= htmlentities($element);
                        }
                        else if (is_int($key2))
                        {
                            $tempString .= htmlentities($element).'||';
                        }
                    }

                    $idModule = Formulaire::getLastInsertedId();
                    Formulaire::createModule( $moduleType, htmlentities($value['order']), htmlentities($value['description']),
                                              $idForm, htmlentities($value['count']), $tempString);
                }
            }
        }
    }
}
```

La méthode `createForm` créé un formulaire comme son nom l'indique en lui donnant le nom renseigné. La méthode `moduleCreation` admet deux possibilités qui découlent des deux types fondamentaux d'`input` (comme vu précédemment lors de l'élaboration de la base de donnée), le groupe `text/textarea/file` et le groupe `select/checkbox/radio`. Elle récupère l'`id` du formulaire fraîchement créé puis en fonction du type d'`input` exécute une requête qui enregistre en base de donnée les modules dans l'ordre d'apparition sur la page, pour une restitution lors de son utilisation dans un article.

Les deux méthodes invoquées sont détaillées ci-dessous :

```
abstract class Formulaire extends Model
{
    public function __construct() {}

    public static function createForm($formName)
    {
        $params = array($formName);

        $sql = 'INSERT INTO forms (name_form)
                VALUES (?)';

        $register = self::requestExecute($sql, $params);

        return $register;
    }

    public static function createModule($moduleType, $moduleOrder, $moduleLabel,
                                      | | | | | | | | $idForms, $optionCount = NULL, $optionNames = NULL)
    {
        if ($optionCount == NULL and $optionNames == NULL)
        {
            $params = array($moduleType, $moduleOrder, $moduleLabel, $idForms);

            $sql = 'INSERT INTO modules (module_type, module_order,
                                         module_label, id_form)
                    VALUES (?, ?, ?, ?, ?)';
        }
        else
        {
            $params = array($moduleType, $moduleOrder, $moduleLabel,
                           $optionCount, $optionNames, $idForms);

            $sql = 'INSERT INTO modules (`module_type`, `module_order`, `module_label`,
                  `option_count`, `option_names`, `id_form`)
                    VALUES (?, ?, ?, ?, ?, ?)';
        }

        $register = self::requestExecute($sql, $params);

        return $register;
    }
}
```

Ainsi, nous revenons aux tables de la base de données et de l'utilité de pouvoir rendre « NULL » les champs « option_count » et « option_names ».

En annexe sont disponibles des vues du formulaire soutien scolaire lors de sa création et de son utilisation lors de la création d'un article

5. Veille sur les vulnérabilités de sécurité

Nous avons étudié les différentes failles de sécurité par le site owasp.org qui est une référence en matière de cybersécurité.

Nous allons voir la fonction qui sert à créer un utilisateur dans la base de données avec 6 paramètres : le prénom, le nom, l'adresse e-mail, le mot de passe, l'adresse et le code postal. Ces six variables sont récupérées via le formulaire d'inscription que j'ai dû coder afin de permettre aux utilisateurs de s'enregistrer sur le site. Extrait du composant inscription.php et début du formulaire :

```
<form action="inscription.php" method="POST">
    <div class="px-2 p-3 mt-1 mb-2">
        <h5>Données de facturation</h5>
    </div>

    <div class="py-1">
        <label for="code_postal" class="h6 py-1 text-muted px-2 fw-light "><i>Insérer votre prénom</i></label><br>

        <input type="text" placeholder="Prénom" name="prenom" required>
    </div>

    <div class="py-1">
        <label for="code_postal" class="h6 py-1 text-muted px-2 fw-light "><i>Insérer votre nom</i></label><br>
        <input type="text" class="p-1 rounded-1 w-100" placeholder="Nom" name="nom">
    </div>

    <div class="py-1">
        <label for="code_postal" class="h6 py-1 text-muted px-2 fw-light "><i>Insérer votre adresse</i></label><br>
        <input type="text" class="p-1 rounded-1 w-100" placeholder="Adresse" name="address">
    </div>
```

Ci-dessous, nous pouvons voir un bout du code relatif au contrôleur qui traite et tri les données envoyées via le formulaire. On voit ici l'utilisation de la fonction PHP htmlentities qui permet d'enlever les caractères spéciaux afin d'éviter la faille connue sous le nom de faille XSS. Le principe de cette faille est d'injecter un code malveillant en langage de javascript dans un site web vulnérable. Par exemple en déposant un message dans un forum qui redirige l'internaute vers un faux site (phishing) ou qui vole ses informations (cookies).

```
// receive all input values from the form
$prenom = htmlentities($_POST['prenom']);
$nom = htmlentities($_POST['nom']);
$password_1 = htmlentities($_POST['password_1']);
$password_2 = htmlentities($_POST['password_2']);
$email = htmlentities($_POST['email']);
$address = htmlentities($_POST['address']);
$zipCode = htmlentities($_POST['code_postal']);

// form validation
// by adding (array_push()) corresponding error unto $errors array
if (empty($prenom)) { array_push($_SESSION['errors'], "Firstname is required"); }
if (empty($nom)) { array_push($_SESSION['errors'], "Lastname is required"); }
if (empty($password_1)) { array_push($_SESSION['errors'], "Password is required"); }
if (empty($email)) { array_push($_SESSION['errors'], "Email is required"); }
if (!preg_match('/^a-zA-Z0-9._-+[a-zA-Z0-9._-]+[a-z]{2,3}$/', $email)) { array_push($_SESSION['errors'], "Email format is wrong"); }
if ($password_1 != $password_2) { array_push($_SESSION['errors'], "The two passwords do not match"); }
if (!preg_match('/^a-zA-Z0-9{8,}$/', $password_1)) { array_push($_SESSION['errors'], "Password format is wrong"); }
if (empty($address)) { array_push($_SESSION['errors'], "Address is required"); }
if (empty($zipCode)) { array_push($_SESSION['errors'], "Code postal is required"); }
if (!preg_match('/^a-zA-Z0-9{5}$/', $zipCode)) { array_push($_SESSION['errors'], "ZipCode format is wrong"); }
```

La combinaison du traitement ci-dessus ainsi que l'utilisation de la préparation de requête puis de l'injection des variables lors de son exécution donne une solide protection contre la faille d'injection SQL. (voir ci-dessous)

```
$sql = "INSERT INTO utilisateurs (prenom, nom, email,
                                    password, address,
                                    code_postal, id_droit)
        VALUES (:prenom, :nom, :email,
                :password, :address,
                :code_postal, :id_droit)";
$params = ([':prenom' => $prenom, ':nom' => $nom, ':email' => $email,
            ':password' => password_hash($password, PASSWORD_DEFAULT), ':address' => $address,
            ':code_postal' => $code_postal, ':id_droit' => 1]);
```

Une injection SQL est une forme de cyberattaque lors de laquelle un pirate utilise un morceau de code SQL (« Structured Query Language », langage de requête structurée) pour manipuler une base de données et accéder à des informations potentiellement importantes.

Il existe aussi une faille qui s'appelle file upload. Le principe de l'attaque est très simple. Le pirate essaie d'uploader un fichier qui contient du code malveillant ou un code PHP de sa création. Si la faille est là alors le fichier finira pas atterrir sur le serveur. Il suffit ensuite au pirate d'appeler son fichier pour que celui-ci s'exécute. Pour contrer cette faille, il faut limiter le type de fichier qui sont téléchargés ainsi que leur taille comme suit :

```
//transfert de l'image vers l'endroit
$file = $_FILES['img']['tmp_name'];
$ext = pathinfo($file, PATHINFO_EXTENSION);

//check extension
if (($ext !== 'gif' || $ext !== 'png' || $ext !== 'jpg') && (filesize($file_tmp) > 1000)) { echo 'error'; }
move_uploaded_file($_FILES['img']['tmp_name'], $targetFile);
```

6. Recherche effectuées à partir d'un site anglophone

Pour créer la fonctionnalité de génération de PDF pour l'inscription aux actions de l'association, nous avons dû rechercher sur internet des solutions. Le langage le plus utilisé en matière d'informatique étant, selon moi, l'anglais, la recherche s'est faite en anglais. Voici, une capture d'écran d'un des résultats obtenu sur le site stackoverflow.com :