

Dossier professionnel

DEVELOPPEUR WEB ET WEB MOBILE

Projet : Mettre en application le questionnaire ISALEM et son interface administrateur

Bruno Delaine | Alkas Formation | Session 2020-2021

Table des matières

Introduction :	2
Résumé :	3
Compétences du référentiel couvertes par le projet :	4
Maquetter une application	4
Réaliser une interface utilisateur web statique et adaptable	4
Développer une interface utilisateur web dynamique	4
Développer les composants d'accès aux données	4
Développer la partie back-end d'une application web ou web mobile	5
Cahier des charges :	5
Spécifications techniques du projet :	7
Maquette du site :	7
Front-end :	8
Back-end :	9
Réalisations et extraits de code	10
Le questionnaire :	10
Affichage du questionnaire :	11
Traitement du questionnaire :	11
Affichage du résultat :	13
Les statistiques :	19
Traitement de la demande :	19
Affichage du graphique :	21
La création d'un admin :	22
Présentation du jeu d'essai	24
Côté utilisateur :	24
Côté administrateur :	27
Veille effectuée sur les vulnérabilités de sécurité	28
Recherche effectuée sur site anglophone et sa traduction	32
Conclusion :	35
Annexes :	36

Introduction :

Dans le cadre de mon stage en entreprise en vue de l'obtention du titre de développeur web et web mobile, j'ai intégré la société WILLGO dirigé par Mr William Godiveau.

WILLGO est une jeune entreprise qui travaille essentiellement sur la création et la maintenance de sites internet à l'aide des CMS WordPress et PrestaShop ainsi que l'accompagnement des clients dans leur stratégie digitale (SEO, gestion des réseaux sociaux, ...)

Mr Godiveau élabore également différents projets personnels directement liés à son activité principale mais également ambitionne la création de son organisme de formation Absys-Formation.

Mes journées ont donc été découpé en deux parties : La participation à l'activité professionnelle de l'entreprise pendant laquelle j'ai pu travailler sur différents sites aux activités variées : BTP, e-commerce, immobilier, éducatifs... Principalement avec WordPress mais aussi PrestaShop, en utilisant différents constructeurs. L'autre partie de la journée étant dédiée à la réalisation de mon projet.

J'ai travaillé seul sous la responsabilité de Mr Godiveau, en totale autonomie avec des points réguliers avec ce dernier afin de discuter de la logique de programmation et de ses attentes sur l'évolution du projet.



Résumé :

En vue de la création de son organisme de formation, Absys-Formation, et étant déjà actif sur différentes plateformes de formation en ligne, Mr Godiveau m'a demandé de mettre en application le questionnaire ISALEM. Il a pour but à travers une série de 12 questions de renvoyer le profil idéal d'apprentissage de l'élève et ainsi permettre au formateur d'adapter son discours en fonction de l'apprenant.

Il m'a donc été demandé de programmer un remplissage simple et rapide du questionnaire pour l'utilisateur, ne nécessitant aucune création de compte à ce niveau, et permettant d'en obtenir le résultat. Quatre résultats sont possibles, représentant chacun l'un des styles d'apprentissage. A noter également la possibilité d'envoyer et/ou recevoir le résultat par mail, permettant par exemple d'informer le formateur.

Du côté administrateur, il est possible d'accéder après connexion à des fonctionnalités de gestion pour la sélection, création, modification ou la suppression de certains éléments (statistiques, administrateurs, mails types, ...). Il existe une hiérarchie des rôles, la gestion des administrateurs n'étant possible que par un SUPER_ADMIN.

Pour la réalisation du projet j'ai donc pu à la fois utiliser PHP pour communiquer avec la base de données, celle-ci composée de 10 tables, et pour sécuriser les différentes parties de l'application et leur accessibilité. Mais également JavaScript pour faciliter le remplissage du questionnaire, sa bibliothèque Chart.js utilisée pour l'élaboration de statistiques et également l'utilisation de Bootstrap pour sa rapidité d'utilisation et la création de carrousels et formulaires.

Compétences du référentiel couvertes par le projet :

MAQUETTER UNE APPLICATION

Le questionnaire ISALEM étant déjà existant, il me fallait d'un côté en respecter ses principes (les questions et réponses, les résultats donnant le profil de l'apprenant et le graphique indiquant d'un point rouge la position de l'apprenant sur celui-ci suite au traitement des réponses données) mais également veiller à intégrer les demandes de Mr Godiveau, à savoir une interface administrateur permettant la gestion des différents composants de l'application mais aussi la lecture de statistiques.

J'ai pour ce faire d'abord esquissé les différentes tables afin de poser mes idées avant une mise au propre avec draw.io afin de soumettre la maquette à Mr Godiveau et commencer à travailler avec son approbation.

REALISER UNE INTERFACE UTILISATEUR WEB STATIQUE ET ADAPTABLE

La structure des pages est faite en HTML. Ses balises intègrent le système de classes de Bootstrap permettant un visuel accessible rapidement et adaptable. En utilisant l'attribut data-label dans les balises <td> de mes différents tableaux qui composent les pages de l'interface administrateur, j'ai pu permettre un visuel adaptable aux différentes tailles d'écrans.

DEVELOPPER UNE INTERFACE UTILISATEUR WEB DYNAMIQUE

Chaque question propose 4 réponses, il s'agit de leur attribuer un classement de 1 à 4.

La programmation que j'ai réalisé du questionnaire permet pour chaque question, d'attribuer les valeurs aux réponses par simple clic dans l'ordre croissant, l'utilisation du clavier numérique n'est donc pas nécessaire. Il est également possible, toujours par simple clic sur un bouton, d'effacer les réponses déjà données par ordre décroissant cette fois. De plus si l'une des réponses présente déjà une valeur et que l'utilisateur tente à l'aide de son clavier numérique de donner la même valeur à une autre réponse de la même question, la première valeur sera alors effacée, garantissant un classement systématique de 1 à 4.

DEVELOPPER LES COMPOSANTS D'ACCES AUX DONNEES

J'ai créé une base de données composées de 10 tables dont certaines relationnelles comme par exemple la table « question » en relation avec la table « answer » ainsi il met possible à l'affichage de récupérer les réponses de chaque question.

J'ai développé une interface administrateur permettant la sélection, modification, création ou l'effacement, en fonction des besoins, les données de 9 tables.

Côté logique de programmation, j'ai utilisé le DataBaseHandler pour convertir les données en Objets.

DEVELOPPER LA PARTIE BACK-END D'UNE APPLICATION WEB OU WEB MOBILE

Côté back-end, une page « db.php » où on retrouve le DataBaseHandler et ses différentes méthodes CRUD communique avec une page « display_function.php » gérant l'affichage des objets renvoyés. A noter une fonction « connect() » dans le constructor de la classe DataBaseHandler ayant pour but de permettre une connexion automatique à chaque instantiation de ce dernier. Une fonction « getIsalemDataBaseHandler » permet de rapidement l'instancier dans les différents fichiers.

```
<?php
$instructions = getIsalemDatabaseHandler()->getAllInstructions();
foreach ($instructions as $instruction){
    echo displayInstruction($instruction);
}
?>
```

Exemple de l'affichage des instructions de remplissage du questionnaire

Dans la globalité du projet les pages communiquent entre elles pour le traitement des données ou leur affichage. Le traitement des données après validation de certaines conditions peut soit se faire soit entrainer une redirection afin d'informer par un message l'utilisateur d'une erreur de champ (erreur de saisie, champ vide, ...).

Cahier des charges :

1. Présentation du commanditaire

Le présent cahier des charges a pour but la création d'un site web permettant la présentation et la réalisation du questionnaire ISALEM, ce dernier créé en 1997 par l'université de Liège n'est à ce jour ni facile ni rapide dans son utilisation. Il convient donc de remédier à cela.

Société : Absys-Formation

Activité : Organisme de formation

Nombre de personnes : Le propriétaire de l'organisme Mr William Godiveau

Description des services : A ce jour l'organisme est en cours de création, son gérant attendant l'agrément nécessaire, cependant il proposera des formations diverses en rapport avec son autre entreprise WILLGO, telles que WordPress, Prestashop, SEO, réseaux sociaux.

2. Nature de la demande

La demande émane de Mr Godiveau, dirigeant de l'agence web WILLGO. Il est également formateur et intervient pour différents organismes, principalement à distance et projette donc très prochainement de représenter son organisme de formation, Absys-Formation. Le besoin est ainsi né de son expérience. Il est important pour lui de mieux connaître ses apprenants, la réalisation du questionnaire ISALEM lui offre cette possibilité.

Le remplissage du questionnaire par l'élève doit être le plus simple possible et ne pas nécessiter la création d'un compte.

Il convient de créer un espace administrateur permettant une gestion la plus globale possible des éléments du site par Mr Godiveau.

Le rendu du projet doit respecter les codes couleurs du site Absys-Formation.

La demande comprend donc la mise en application de manière dynamique du questionnaire ISALEM, de respecter ses fondamentaux (algorithme de traitement, profils renvoyés). Elle comprend également la mise en place d'outils permettant de consulter les statistiques de réalisation du questionnaire.

3. Prestations attendues

Technique :

- Création d'un site présentant le questionnaire ISALEM, ses questions et réponses et renvoyant l'un des 4 profils d'apprentissage possibles.
- Ajout d'un formulaire de contact
- Statistiques de réalisation du questionnaire consultable depuis l'interface administrateur

Graphique :

- Respect des codes couleurs du site Absys-Formation (#053f81, #f48120)
- Il doit être simple de remplir le questionnaire
- Le résultat doit être également accompagné d'un graphique fidèle à celui créé en 1997.
- Il doit être responsive et donc adaptable sur pc, tablettes et mobiles.

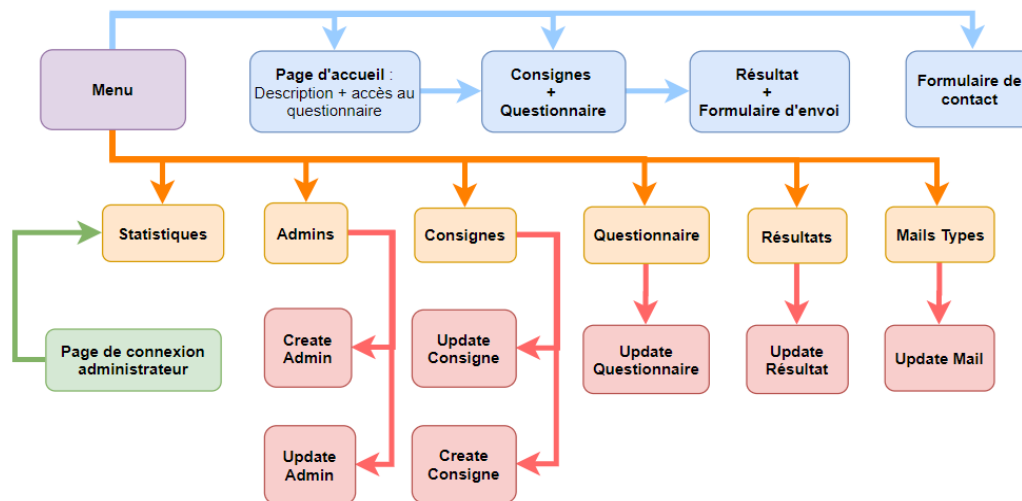
4. Livrables impératifs

En fin de prestation, Mr Godiveau doit recevoir les éléments suivants :

- Un site web permettant de réaliser le questionnaire ISALEM, d'en obtenir le résultat et la possibilité d'envoyer le résultat par email.
- Un espace administrateur permettant la gestion des éléments du questionnaire et la consultation de statistiques de réalisation de celui-ci.

5. Arborescence du site

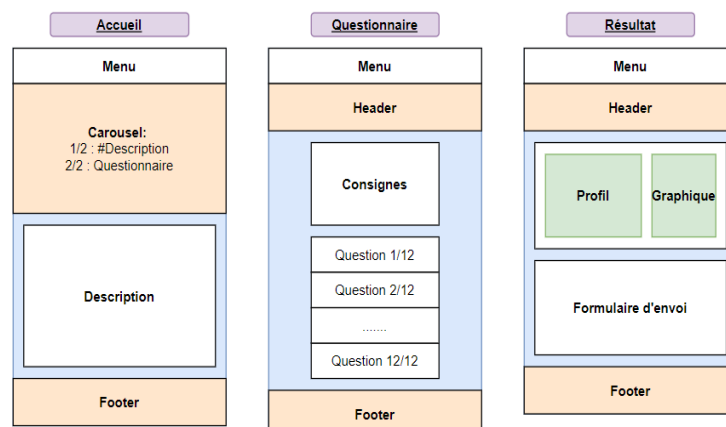
L'arborescence du projet correspondra au schéma suivant :

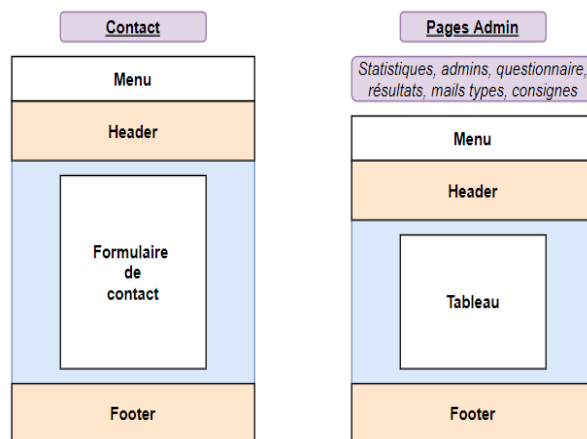


Spécifications techniques du projet :

MAQUETTE DU SITE :

Après discussion avec Mr Godiveau, à partir de ses attentes de gestion des composants du site et après recherches pour comprendre l'architecture et comment fonctionne le questionnaire ISALEM, j'ai, à partir de draw.io, maquetter le projet.





FRONT-END :

Pour cette partie j'ai utilisé les langages suivants :

HTML :

Pour structurer et mettre en forme les pages, mais également pour obtenir une première validation de conformité du questionnaire avec les attributs max et min par exemple ou l'attribut pattern pour les formulaires.

JavaScript (et ses bibliothèques ChartJS et JQuery) :

Ce langage m'a permis d'intervenir à plusieurs niveaux sur la programmation du projet :

La page du questionnaire :

- Pour la facilité de remplissage du questionnaire, chaque réponse d'une question devant être notée de 1 à 4, une fonction m'a permis à chaque clic de noter 1, puis 2, puis 3 et 4 les réponses. Evitant ainsi l'utilisation du clavier numérique.
- Toujours dans un but de faciliter le questionnaire, une autre fonction à partir d'un bouton présent pour chaque question permet d'effacer à chaque clic une réponse dans l'ordre décroissant des valeurs présentes.
- Pour chaque question les valeurs attribuées doivent être unique. Pour éviter une valeur doublée dans les réponses d'une question (par exemple la présence de deux fois la valeur 3) une fonction, lorsqu'une valeur déjà attribuée est ajoutée de nouveau, efface la première valeur. Ainsi si la réponse « a » a comme valeur 3 et que l'utilisateur décide de mettre à l'aide de son clavier numérique à la réponse « b » une nouvelle fois la valeur 3, la valeur de la réponse « a » sera effacée.

La page du résultat :

- Un des impératifs de la prestation était d'obtenir un graphique affichant l'axe des abscisses et des ordonnées, la valeur 0 étant le centre de celui-ci. Il fallait qu'un point s'affiche, correspondant à la jonction des points x et y, ces derniers étant le résultat du traitement du questionnaire précédemment rempli. J'ai donc utilisé la balise canvas en y associant les différentes fonction Javascript pour tracer le graphique, la graduation des axes et l'affichage du point.

La page d'accueil :

- La bibliothèque JQuery est brièvement utilisée pour simplement modifier le comportement par défaut du carrousel bootstrap. En effet utilisé dans sa version d'origine, il se met en pause dès que le curseur est dessus.

- La bibliothèque ChartJS, quant à elle, m'a permis de créer les statistiques du côté administrateur.

Bootstrap :

Ce Framework m'a permis de créer rapidement une interface graphique adaptable grâce à son système de classes. Particulièrement utile pour la mise en page des différents formulaires et pour la création de carrousels en page d'accueil et côté interface administrateur pour la page de statistiques.

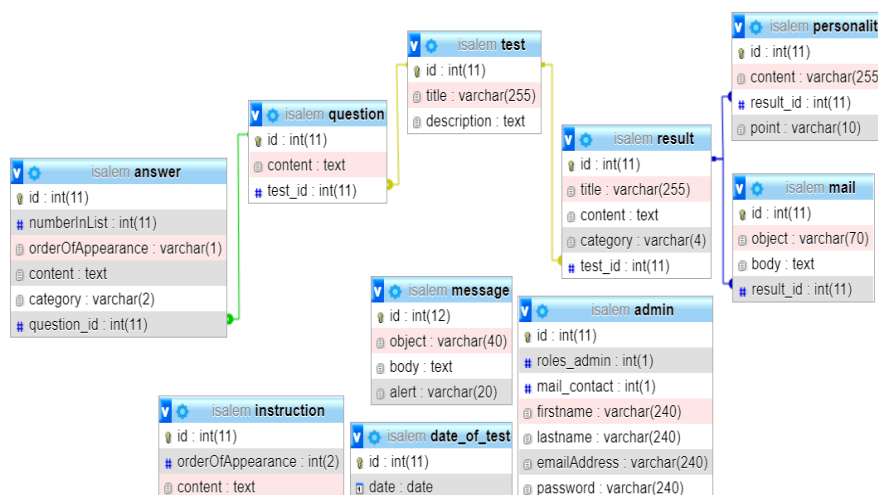
BACK-END :

phpMyAdmin:

L'interface graphique qui me permet de gérer le gestionnaire de la base de données du projet.

MariaDB :

Le gestionnaire de la base de données. Cette dernière est composée de 10 tables, dont certaines relationnelles.



PHP :

La logique d'exécution du projet se base sur l'outil DataBaseHandler qui va agir dans la logique d'un ORM, pour Object Relational Mapping, permettant de convertir les données de la base de données en des Objets manipulables dans PHP. Le code se retrouve ainsi plus lisible avec la création de méthodes permettant alors le dialogue avec SQL, et va renvoyer des objets à partir de mes modèles créés, c'est-à-dire des fichiers créés et retournant chacun d'eux une classe représentant une table de la base de données.

Un autre fichier nommé « display_function.php » me permet d'y programmer toutes les fonctions d'affichage des objets renvoyées par les méthodes du fichier précédent.

L'utilisation de PHP m'a permis également de sécuriser l'accès à l'interface admin en programmant entre autres la vérification dans \$_SESSION, grâce à la fonction isset(), d'un ID admin valide pour chaque fichier. Mais aussi à sécuriser le traitement du questionnaire pour vérifier que toutes les données souhaitées existent (présence de toutes les réponses attendues, données numériques entre 1 et 4, total des variables de \$_POST renfermant les réponses devant être égal à 120, ...). Ainsi, bien que du côté HTML par ses attributs et du côté Javascript par des fonctions créées, des vérifications de conformité du questionnaire sont déjà existantes, elles ne s'arrêtent pas là puisque facilement contournables pour un utilisateur renseigné.

Ce langage me permet également, pour la page contact, la mise en place d'un captcha et une vérification de conformité des champs par la fonction preg_match().

Des vérifications du côté de la gestion des admins (uniquement accessible par un SUPER_ADMIN) sont également faites côté PHP comme par exemple :

- Hachage du mot de passe dans la base de données avec la fonction password_hash() et sa vérification à la connexion avec cette fois password_verify().
- Ne pas pouvoir effacer un admin dont l'adresse email est enregistrée comme email de contact, un message d'information est alors renvoyé.
- La création ou la modification d'un admin ne peut pas se faire si l'adresse email renseignée est déjà utilisée.

Réalisations et extraits de code

LE QUESTIONNAIRE :

La première véritable tâche qui a demandé une réflexion, une succession d'essais de différentes logiques fut le questionnaire ISALEM, à savoir son affichage, son traitement, et son résultat.

Affichage du questionnaire :

Sur ce premier niveau il s'agissait d'afficher un formulaire ayant une première question demandant si l'apprenant à plus ou moins de 18 ans, et l'affichage du questionnaire et donc de chaque question et de ses 4 réponses associées.

Cela a nécessité 2 tables « question » et « answer » en relation « n1 » puisqu'une question peut avoir plusieurs réponses et une réponse une seule question.

A partir de cela, il me fallait créer, dans le fichier « db.php », des méthodes CRUD, plus précisément des READ ou SELECT, appartenant à ma classe « DataBaseHandler ». Ainsi que des fonctions d'affichage, dans un fichier spécifique à ces fonctions (« display_function.php »), pour les méthodes renvoyant des objets.

Pour cela j'ai créé 2 méthodes, l'une permet la récupération d'un objet « question » par son ID (voir Annexe 1) et l'autre la récupération d'un tableau d'objets de réponses par l'ID d'une question (voir Annexe 2).

Ainsi l'affichage peut se faire dans la logique suivante :

- Une boucle « for » qui pour chaque tour va utiliser l'index comme paramètre de la méthode `getQuestion(int $id)`, l'objet récupéré sera donné comme paramètre à sa fonction d'affichage.
- Et dans ce même tour, récupération du tableau d'objets de réponses avec le même index comme paramètre. Ainsi sont obtenues les bonnes réponses à la question récupérée précédemment. Un dernier « foreach » sur ce tableau me permet d'appeler la fonction d'affichage de la réponse avec comme paramètre l'objet.

```
80 | for ($i = 1; $i <= 12; $i ++){  
81 |     ?>  
82 |  
83 |     <div class="question-answers">  
84 |  
85 |     <?php  
86 |     $question = getIsalemDatabaseHandler()->getQuestion($i);  
87 |     echo displayQuestion($question);  
88 |     $answers = getIsalemDatabaseHandler()->getAnswersForOneQuestion($i);  
89 |     foreach ($answers as $answer){  
90 |         echo displayAnswer($answer);  
91 |     }  
92 |     ?>
```

Traitement du questionnaire :

- *Problématique rencontrée :*

Le questionnaire maintenant rempli et l'apprenant ayant confirmé l'envoi, il fallait d'abord récupérer les réponses et en retirer un résultat, mais également faciliter l'interprétation de ce résultat. Le traitement prévoyait une difficulté supplémentaire, mais l'algorithme offrait l'avantage d'être déjà connu :

- Plus de 18 ans : $\$x = \$I - \$Ab - 8$ et $\$y = \$Ac - \$R - 5$
- Moins de 18 ans : $\$x = \$I - \$Ab - 2$ et $\$y = \$Ac - \$R - 2$

Il y a donc 48 réponses nécessitant une valeur numérique de 1 à 4 et lors du traitement 4 variables dont la somme chacune de 12 réponses permettent le calcul ci-dessus. Cependant, afin d'éviter de deviner par avance un possible profil de résultat, toutes les réponses a, b c ou d ne remplissent pas la même variable mais suivent un schéma précis. Par exemple la valeur de la réponse « a » de la question « 1 » s'ajoute à la variable « \$Ab » tandis que celle de la question « 2 » à la variable « \$R ». Il fallait donc pouvoir cibler les réponses et les rediriger dans la bonne variable. Ceci ayant pour but d'obtenir une valeur « x » et « y » déterminant la position de l'apprenant sur un graphique et donc son profil.

- *Solution apportée :*

Afin de récupérer les valeurs des 48 réponses et pouvoir rediriger chacune d'elle dans la bonne variable, il me fallait à la fois une identité propre à chaque réponse mais aussi un repère permettant de les diriger vers les bonnes variables nécessaires à l'algorithme. J'ai donc ajouté une colonne « category » à la table « answer », celles-ci au nombre de 4 (Ab, Ac, Ai, Ar). J'ai ainsi prédéfini dans quelle variable traiter la valeur de chaque réponse. Pour l'identité propre, j'ai ajouté une colonne « numberInList » à ma table, attribuant ainsi un nombre de 1 à 48.

```

207     return sprintf("
208         <div class='d-flex input-and-answer'>
209             <div class='m-1 div-input'>
210                 <input type='number' name='%s%d' class='text-center input_answer' required autocomplete='off' min='1' max='4'>
211             </div>
212             <p class='ml-2 mr-1 text-wrap'>
213                 %s) %s
214             </p>
215         </div>",
216         $answer->category, // $answer->category combiné à $answer->numberInList permet un attribut name propre à chaque input
217         $answer->numberInList,
218         html_entity_decode($answer->orderOfAppearance, ENT_HTML5),
219         html_entity_decode($answer->content, ENT_HTML5)
220     );

```

Chaque réponse étant identifiable l'une de l'autre, j'ai donc pu diriger avec exactitude leur valeur dans l'une des 4 variables avant de procéder au calcul de x et y.

```

22 | foreach($_POST as $key => $value){
23 |
24 |     if ($value === 'under18' || $value === 'over18'){
25 |         // quand la valeur under18 ou over18 est rencontrée on passe à la suivante sans rien faire de plus
26 |         //.. traitement des valeurs numériques, elles doivent être un chiffre supérieur à 0 et inférieur à 5.
27 |     } else if(is_numeric($value) && $value > 0 && $value < 5){
28 |
29 |         if ($key[1] === "b"){
30 |             $Ab[] = $value;
31 |
32 |         } else if ($key[1] === "i"){
33 |             $I[] = $value;
34 |
35 |         } else if ($key[1] === "c"){
36 |             $Ac[] = $value;
37 |
38 |         } else if ($key[1] === "r"){
39 |             $R[] = $value;
40 |         }

```

Le fait de les inclure d'abord dans un tableau me permet une vérification supplémentaire, à savoir que chaque tableau à la fin du foreach() possède bien 12 valeurs et je vérifie également que la somme de tous les tableaux soit bien égale à 120 car il ne peut en être autrement.

```

47 | //Vérification de chaque tableau, ils doivent contenir obligatoirement 12 valeurs
48 | if(count($I) === 12 && count($Ab) === 12 && count($R) === 12 && count($Ac) === 12){
49 |     //Vérification également que la somme de toutes les réponses est bien égale à 120
50 |     if(array_sum($I)+array_sum($Ab)+array_sum($Ac)+array_sum($R) == 120){

```

Après vérification à la question « plus ou moins de 18 ans » il est possible de procéder au calcul de x et y.

```

52 | if ($_POST["age"] === "over18"){ //si plus de 18 ans ce calcul est fait à partir de la somme de chaque tableau
53 |     $x = array_sum($I) - array_sum($Ab) - 8;
54 |     $y = array_sum($Ac) - array_sum($R) - 5;
55 |
56 | } else if ($_POST["age"] === "under18"){ //..si moins de 18 ans c'est ce calcul qui est fait
57 |     $x = array_sum($I) - array_sum($Ab) - 2;
58 |     $y = array_sum($Ac) - array_sum($R) - 2;
59 | }

```

Affichage du résultat :

- *Problématique rencontrée :*

Maintenant que les valeurs de x et y sont connues, il me fallait les interpréter pour afficher le résultat approprié ; c'est à dire le bon profil de l'apprenant ainsi que la représentation visuelle d'un point sur le graphique correspondant à la rencontre de x et y.

- *Solution apportée :*

(La méthode d'affichage se découpe en 2 parties, la première en PHP et la seconde en Javascript.)

PHP

En définitive le bon profil ne se résumant pas à une valeur précise de x et y mais au fait que celles-ci soient positive ou négative, j'ai ajouté une colonne « category » à la table « result » ; 4 résultats étant possibles, chacun possède une catégorie en fonction de sa position sur le graphique à savoir « -xy », « xy », « x-y » et « -x-y ».

Ainsi j'ai pu, en déterminant la nature positive ou négative de x et y, créer une chaîne de caractères correspondant à l'une de ces catégories et la renvoyer dans \$_SESSION afin de l'interpréter dans la page de résultat par une méthode getResultofTest(\$category) et de l'afficher à partir d'une fonction displayResult(\$result).

```
68 //Vérification de la nature positive ou négative de $x...
69 if ($x >= 0){
70     $result = "x";
71 } else if ($x < 0) {
72     $result = "-x";
73 }
74 //...ainsi que celle de $y...
75 if ($y >= 0){
76     $result = $result."y";
77 } else if ($y < 0){
78     $result = $result."-y";
79 }
80
81 //...à présent la variable $result renfermant l'un des quatre
82 //indicateurs de profil existant du test ISALEM
83 //à savoir -xy, xy, x-y ou -x-y
84
85 //Les 3 variables $result, $x, $y sont préparées pour être
86 //interprétées dans la page de résultat.
87 $_SESSION['result'] = $result;
88 $_SESSION['x'] = $x;
89 $_SESSION['y'] = $y;
```

tr_questionnaire.php

```

516 //Va renvoyer un objet Result en fonction de la categorie donnée
517 public function getResultOfTest(string $category){
518     $sqlQuery = (
519         "SELECT *
520         FROM result
521         WHERE :category = category");
522
523     $stmt = $this->_handler->prepare($sqlQuery);
524     $stmt->execute(
525         [
526             ':category' => $category,
527         ]
528     );
529     $res = $stmt->fetch(PDO::FETCH_OBJ);
530     return new Result(
531         $res->id,
532         $res->title,
533         $res->content,
534         $res->category);
535 }

```

db.php

Le traitement du questionnaire côté PHP étant au point, le travail n'était pas pour autant terminé car il fallait maintenant faire apparaître le graphique avec le point placé en fonction de x et y.

Comme visible sur la fin de l'extrait de code du fichier « tr_questionnaire.php », les valeurs de « \$x » et « \$y » sont également renvoyées par l'intermédiaire de « \$_SESSION ».

JavaScript

J'ai opté pour l'utilisation de la balise <canvas> pour faire apparaître le graphique voulu et créé plusieurs fonctions dans son élaboration.

Sachant que le graphique comporte de nombreuses lignes puisqu'incluant les axes des abscisses et des ordonnées mais aussi une graduation, une première fonction permet cela.

```

13 //fonction permettant de tracer des lignes dans le canvas
14 function drawLine(x1, y1, x2, y2, linewidth, color){
15     context.beginPath()
16     context.moveTo(x1,y1)
17     context.lineTo(x2,y2)
18     context.lineWidth = linewidth
19     context.strokeStyle = color
20     context.stroke()
21 }

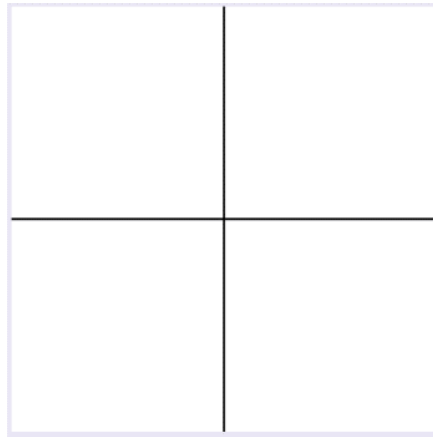
```

J'ai donc pu tracer les 2 axes.


```

23 //ici est tracé l'axe des abscisses et des ordonnées du graphique
24 drawLine(190,0,190,380, 2,"black")
25 drawLine(0,190,380,190, 2,"black")

```



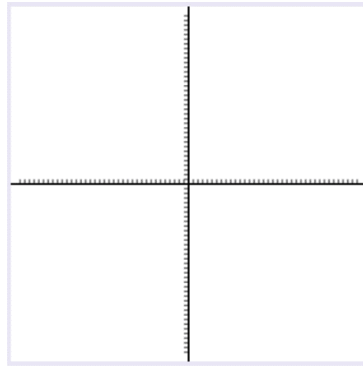
Extrait code et graphique étape 1

Puis effectuer la graduation des axes à l'échelle de 5px pour 1mm.

```

29 //graduation de l'axe horizontal
30 let a = 10
31 for (let i=0; i<73; i++){
32     drawLine(a,190,a,195, 1,"black")
33     a += 5
34 }
35 //puis vertical
36 let b = 370
37 for (let i=0; i<73; i++){
38     drawLine(190,b,185,b, 1,"black")
39     b -= 5
40 }

```



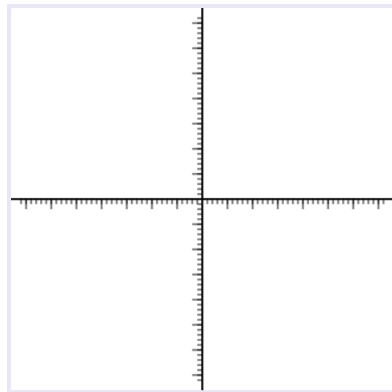
Extrait code et graphique étape 2

La graduation nécessitait un repère plus grand tous les 25px (ou à taille réelle 5mm), j'ai mis en place pour cela 2 fonctions qui les tracent (voir Annexe 3) en partant du point central du canvas, visuellement 0 et dans sa programmation du canvas (190,190).

```

77  graduationX(36,190,190,200,5)
78  graduationX(36,190,190,200,-5)
79
80  graduationY(36,190,190,180,5)
81  graduationY(36,190,190,180,-5)

```



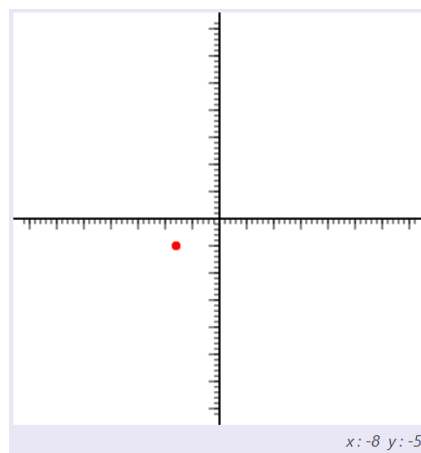
Extrait code et graphique étape 3

J'ai ensuite récupéré les valeurs de x et y dont l'information est située sous le graphique afin de les utiliser dans la création du repère visuel. Pour cela j'ai également dû convertir leur valeur afin d'être à l'échelle du graphique.

```

86 //récupération des valeurs de x et y importées par $_SESSION dans result.php
87 //ces valeurs vont être utilisées afin de créer le point de référence de l'utilisateur
88 //sur le graphique
89 let spanX = document.getElementById("x")
90 let spanY = document.getElementById("y")
91 //afin d'être à l'échelle du canvas (5px = 1mm), les valeurs de x et y sont multipliées par 5.
92 //le point x partant du bord gauche, il est additionné
93 //à la moitié du canvas
94 let xValue = 190 + (parseInt(spanX.textContent)*5)
95 //le point y partant du haut du canvas mais l'axe des ordonnées
96 //partant de bas vers le haut, ici la logique s'inverse, y est soustrait à 190.
97 let yValue = 190 - (parseInt(spanY.textContent)*5)
98 //..Maintenant yValue et xValue ont été converties afin d'être placées d'après les règles imposées
99 //par canvas tout en donnant à l'utilisateur une vision précise de son point sur le graphique
100 //d'après ses résultats.
101
102 context.beginPath()
103 context.lineWidth = "2"
104 context.fillStyle = "red"
105 context.arc(xValue,yValue,4,0,2*Math.PI)
106 context.fill()

```



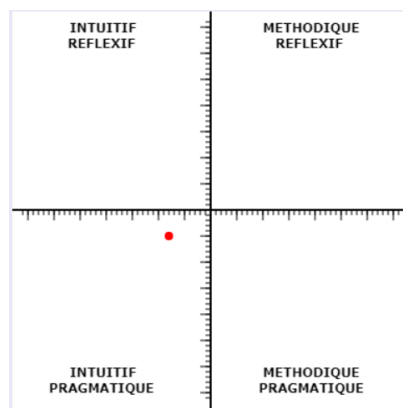
Extrait code et graphique étape 4

La dernière étape consistait à faire apparaître les noms des 4 grands profils sur le graphique. Là également, j'ai créé une fonction me permettant de rapidement le faire.

```

111 //fonction pour l'insertion de texte sur la canvas
112 //3 paramètres : le texte, la distance du texte par rapport au bord gauche du canvas
113 //et la distance par rapport au bord haut.
114 function writeTextInCanvas(text,left,top){
115     context.font = "bold 12px Verdana, Arial, serif"
116     context.fillStyle = "black"
117     context.fillText(text,left,top)
118 }
119
120
121 writeTextInCanvas("INTUITIF",55,20)
122 writeTextInCanvas("REFLEXIF",53,35)
123
124 writeTextInCanvas("METHODIQUE",240,20)
125 writeTextInCanvas("REFLEXIF",252,35)
126
127 writeTextInCanvas("METHODIQUE",240,350)
128 writeTextInCanvas("PRAGMATIQUE",236,365)
129
130 writeTextInCanvas("INTUITIF",55,350)
131 writeTextInCanvas("PRAGMATIQUE",35,365)

```



Extrait code et graphique étape 5

LES STATISTIQUES :

Du côté administrateur, l'affichage de statistiques était une demande spécifique du commanditaire du projet et comme pour l'affichage du résultat précédemment expliqué cela a nécessité l'utilisation de PHP et de Javascript.

Traitement de la demande :

J'ai mis en place tout d'abord un formulaire simple à deux champs, l'un pour renseigner le mois, il s'agit d'un bouton select dont les options correspondent chacune à un mois de l'année écrit en toutes lettres et l'autre l'année dont la valeur par défaut est l'année en cours par l'utilisation de :

```
<?php echo date("Y");?>
```

L'administrateur peut donc s'il remplit les 2 champs obtenir les statistiques jour par jour du mois de l'année demandé ou s'il ne remplit pas le mois, ceux mois par mois de l'année souhaitée.

Je récupère ensuite par l'intermédiaire de \$_POST les données afin de les interpréter et déterminer la demande de l'administrateur ; souhaite t'il les statistiques sur un mois ou de toute une année ? (voir Annexe 4)

Après cette condition vérifiée, le traitement de la demande, maintenant comprise, peut avoir lieu. Pour cela, comme visible aux lignes 97 et 103 de l'extrait de code de l'Annexe 4 j'ai mis en place 2 fonctions.

L'une pour renvoyer toutes les données mois par mois d'une année (voir Annexe 5) et la seconde pour renvoyer celles jour par jour dans un mois (voir Annexe 6).

Ces deux fonctions me permettent de récupérer un tableau d'objets dont les propriétés permettent plusieurs choses :

D'abord la création du carrousel Bootstrap, qui va nécessiter une <div> dont l'ID prend la valeur « slide-\$number », et également le remplissage du « slide » avec les bonnes données. Une fonction d'affichage, unique aux deux possibilités, inclus dans un « foreach() » permet cela :

120										<?php
121										foreach(\$datas as \$data){
122										echo displayDataDate(\$data);
123										}
124										?>

Extrait « statistiques.php »

```

305 //Permet l'affichage des propriétés d'un objet, à savoir sa position dans le carrousel
306 //sa clé (jour ou mois), le complément de cette clé (formats MM/YYYY OU YYYY) et le nombre de
307 //questionnaires réalisés à la date ("clé + complément")
308 function displayDataDate($data){
309
310     return sprintf("
311         <div class='carousel-item h-100' id='slide-%d'>
312             <p class='text-center'>
313                 <span class='font-italic'><span class='labels'>%s</span>/%s : </span>
314                 <span class='data_numbers'>%s</span>
315                 test(s) réalisé(s)
316             </p>
317         </div>
318         ",
319         $data->slide,
320         $data->label,
321         $data->dateComplement,
322         $data->data_number
323     );
324 }

```

Extrait « display_function.php »

Affichage du graphique :

Le carrousel maintenant créé, restait la création du graphique. Pour cela j'ai utilisé la bibliothèque Chart.js et sa documentation.

Pour qu'un graphique puisse s'afficher, certaines données sont évidemment primordiales et potentiellement différentes à chaque demande. Il me fallait donc principalement récupérer : les « labels » ou clés permettant de titrer chaque colonne du graphique, les « data » ou données qui seront les valeurs numériques associées aux clés, la dernière nommée « label » détermine le titre affiché en haut du graphique.

Il m'a fallu donc récupérer ces données puisque déjà existantes depuis le traitement PHP précédent. Ainsi le titre du graphique se trouvant affiché en première page du carrousel, j'ai pu par son id le récupérer. Tandis que les clés, et valeurs associées, également déjà présentes dans le carrousel, par leur classe

```

3   let labels = document.getElementsByClassName("labels");
4   let data_numbers = document.getElementsByClassName("data_numbers");
5   let labelGraph = document.getElementById("labelGraph");

```

Les deux dernières renvoyant des collections, la mise en place d'une boucle « for » pour chacune m'a permis de récupérer la valeur des éléments et de les intégrer dans un tableau

```

8      //une boucle pour récupérer le contenu des éléments de la classe labels et ajouter ces valeurs
9      //dans un tableau
10     for (let i=0;i<labels.length;i++){
11         label = labels[i].innerHTML;
12         labelsArray.push(label);
13     }
14
15     //même procédé pour le contenu des éléments de data_numbers
16     for (let i=0;i<data_numbers.length;i++){
17         data_number = data_numbers[i].innerHTML;
18         data_numbersArray.push(data_number);
19     }

```

Il ne me restait simplement à utiliser les données ainsi récupérées pour programmer le tableau de statistiques qui serait renvoyé

```

24     let ctx = document.getElementById('canvas').getContext('2d');
25
26     let chart = new Chart(ctx, {
27
28         type: 'bar',
29
30         data: {
31
32             labels: labelsArray,
33             datasets: [{
34                 label: labelGraph.textContent,
35                 backgroundColor: 'rgb(244, 129, 32)',
36                 borderColor: 'rgb(244, 129, 32)',
37                 data: data_numbersArray
38             }]
39         },
40
41         options: {}
42     });

```

LA CREATION D'UN ADMIN :

Coté back-end, le traitement de la création d'un admin a nécessité plusieurs étapes avant validation de la demande.

- 1- Vérification comme pour toutes les pages admin de l'existence d'un ID admin dans \$_SESSION avec la fonction isset().
- 2- Récupération de l'objet admin afin de vérifier si celui-ci est bien un SUPER_ADMIN, car seul rôle autorisé à la gestion des administrateurs.
- 3- Vérification de la validité de tous les champs du formulaire par la fonction isset().

- 4- A ce niveau des vérifications, j'en conclu que le formulaire n'a pas été modifié volontairement, j'enregistre donc les différentes valeurs de champs reçues par \$_POST dans des variables dans \$_SESSION. L'intérêt est que, si l'une des vérifications qui vont suivre empêche le processus de création d'un admin et redirige vers le formulaire, celui-ci ne sera pas remis à zéro. Il contiendra les informations notées par le SUPER_ADMIN avant l'envoi, il ne lui restera plus qu'à corriger les champs ne remplissant pas les conformités demandées.
- 5- Vérification si aucun des champs n'est vide avec la fonction empty().
- 6- Vérification si le mot de passe et sa répétition sont identiques.
- 7- Tentative de connexion par l'adresse email notée, le but est de vérifier si celle-ci n'est pas déjà enregistrée en base de données pour un autre administrateur.
- 8- Vérification si la case « Email de contact » du formulaire a été cochée. Si c'est le cas l'adresse email de ce nouvel admin doit être la seule enregistrée comme telle en base de données. Il convient par une méthode d'enlever l'adresse email de contact actuelle pour permettre à celle-ci de la remplacer.

```
//si la case contact mail est cochée alors avant modification, appel de la méthode suivante pour
//supprimer le mail de contact actuel, et $contactMail sera égale à 1
//ainsi cet admin modifié devient le seul mail de contact.
if (!empty($_POST['contact_mail'])){
    getIsalemDatabaseHandler()->updateAdminContact();
    $contactMail = 1;
    //sinon il sera égale à 0 et ne sera donc pas le contact mail administrateur.
} else {
    $contactMail = 0;
}
```

Étape 8

```
166 | //lorsque le mail de contact doit être changé, la méthode est appelée
167 | //pour remettre à 0 la valeur de la colonne d'admin contactMail étant à 1
168 | //c'est à dire que l'admin qui était le mail de contact ne l'est plus.
169 | public function updateAdminContact(){
170 |     $sqlQuery = (
171 |         "UPDATE admin SET
172 |         mail_contact = 0
173 |         WHERE mail_contact = 1");
174 |
175 |     $stmt = $this->_handler->prepare($sqlQuery);
176 |     $stmt->execute();
177 | }
```

Méthode nécessaire à l'étape 8

- 9- Cryptage du mot de passe par l'utilisation de password_hash().
- 10- « Nettoyage » des données reçues par la fonction cleanData().


```

3 //Retourne une donnée "nettoyée" c'est à dire :
4 //- sans espace au début et à la fin de la chaîne
5 //- en supprimant les antislashes
6 //- et convertissant les caractères spéciaux
7 function cleanData($data){
8     $data = trim($data);
9     $data = stripslashes($data);
10    $data = htmlspecialchars($data);
11
12    return $data;
13 }

```

Fonction nécessaire à l'étape 10

11- Appel de la méthode createAdmin()

12- Redirection vers le tableau d'admins avec message affichant la réussite de la procédure.

Présentation du jeu d'essai

COTE UTILISATEUR :

Le traitement du questionnaire a été l'un des points les plus importants à programmer et nécessitant donc une vigilance à trois niveaux : la récupération des données dans \$_POST, les différents calculs effectués à partir de ces données et leur interprétation permettant l'affichage du résultat.

Pour commencer, j'ai inscrit dans la fonction d'affichage des réponses du questionnaire une valeur par défaut égale à 4.

Les réponses reçues et analysées par un var_dump() sur \$_POST étaient donc toutes identiques à l'arrivée et donc d'après le calcul de traitement, nous avons, dans le cas par exemple d'un utilisateur de plus de 18 ans, \$x égale à -8 et \$y à -5.

Aucun souci n'apparaissait à ce niveau, le profil était le bon et le point semblait correctement placé. La récupération des données se faisait donc sans problème.

Cependant il ne s'agissait que d'une vérification très sommaire puisque l'intérêt du questionnaire était qu'il soit conforme et donc qu'il ne propose pas des valeurs systématiquement identiques mais bien pour chaque question, un classement de valeurs des réponses de 1 à 4.

J'ai donc moi-même répondu au questionnaire tout en effectuant le calcul manuellement afin de comparer aux résultats post-traitement. Pour m'aider dans ce calcul et vérifier chaque étape, j'ai permis un affichage lisible de toutes les étapes effectuées entre l'envoi du questionnaire et avant l'affichage du résultat.

J'ai pu à partir de cette page analyser les données et procéder à mes vérifications (ceci va nécessiter la consultation de l'annexe 7 en page 41) :

- Dans l'encadré bleu, les données reçues dans \$_POST : On peut voir la réponse « over18 » ainsi que les 48 réponses du questionnaire. Je les ai classé à l'aide d'une boucle for dans 4 tableaux permettant de regrouper toutes les réponses d'une même « famille » (Ab, Ai, Ac, Ar) sur lesquels j'ai utilisé la fonction print_r() afin d'en améliorer la lisibilité et la comparaison avec les 4 tableaux qui suivront.
- Les 4 tableaux suivant sont en effet les tableaux réellement utilisés dans les calculs à venir. Il m'a suffi de comparer comme les codes couleurs le montrent (rouge avec rouge, vert avec vert, ...) que chacun recevait les mêmes données et que le total pour chaque était le bon.
- L'encadré vert indiquant « calcul de x et y si plus de 18 ans » va permettre de voir que le calcul appliqué et le bon car, comme indiqué tout au début de la page, la réponse de l'âge est bien « over18 »
- L'encadré jaune suivant va maintenant indiquer les valeurs réelles utilisées dans l'affichage du résultat, ce sont ces valeurs x, y et result qui sont enregistrées dans \$_SESSION. Ainsi il est vérifiable que les calculs de x et y sont corrects de par le calcul expliqué dans l'encadré vert mais aussi que result renverra la bonne chaîne de caractère, à savoir ici « -xy », x étant égal -9 donc négatif et y à 2 donc positif.
- Avant de rediriger vers la page de résultat, dans l'encadré orange, j'utilise la méthode « getResultOfTest(-xy) » et sa fonction d'affichage afin de constater le profil renvoyé.

Ces vérifications terminées l'affichage peut à présent être fait, vérifiant ainsi que les données x et y tous deux affichés sous le graphique et le profil qui l'accompagne coïncident avec la page précédente.

Intuitif Réflexif

Vous excellez à considérer une situation sous des angles très variés. Votre réaction initiale est plutôt d'observer que d'agir.

Vous appréciez les situations qui nécessitent un **foisonnement d'idées** comme, par exemple, lors d'un "brainstorming".

Vous avez des **intérêts culturels** très larges et vous aimez rassembler des informations avec éclectisme.

Vos points forts

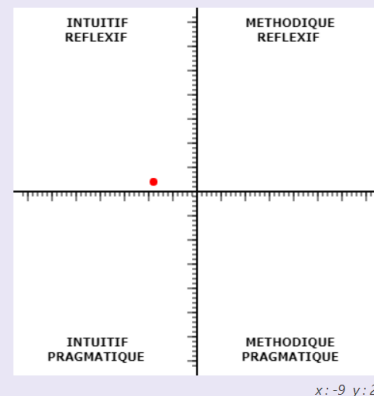
Vous êtes particulièrement doué pour :

- imaginer
- comprendre les gens
- identifier les problèmes.

Vos points faibles

Vous auriez tendance à :

- hésiter dans vos choix
- retarder vos décisions



Maintenant la récupération, le calcul et l'affichage vérifiés, au fur et à mesure de la programmation du projet, il me fallait régulièrement vérifier que le traitement et l'affichage du résultat se fassent toujours correctement. Il est vite devenu répétitif et contraignant de soit enregistrer une valeur par défaut qui par conséquent offrait la possibilité d'un questionnaire non conforme, soit de remplir aléatoirement à la main les réponses ...

J'ai donc mis en place une fonction Javascript auto-invoquée qui m'a permis pour chaque consultation de la page du questionnaire, d'obtenir un questionnaire prérempli de réponses aléatoires et répondant aux attentes de conformité.

```
37 ;(function testRandom(){
38     //récupération de la collection d'objet de class input-radio (les deux réponses possibles à la question
39     //avez-vous plus de 18 ans ?)
40     let radio = document.getElementsByClassName("input-radio")
41
42     //sélection aléatoire entre 0 et 1
43     radio_checked = Math.floor(Math.random() * 2)
44
45     //la sélection précédente devient l'index de la collection radio et prend l'attribut checked déterminant la réponse
46     //à la question posée
47     radio[radio_checked].setAttribute("checked","")
48 }
```

```

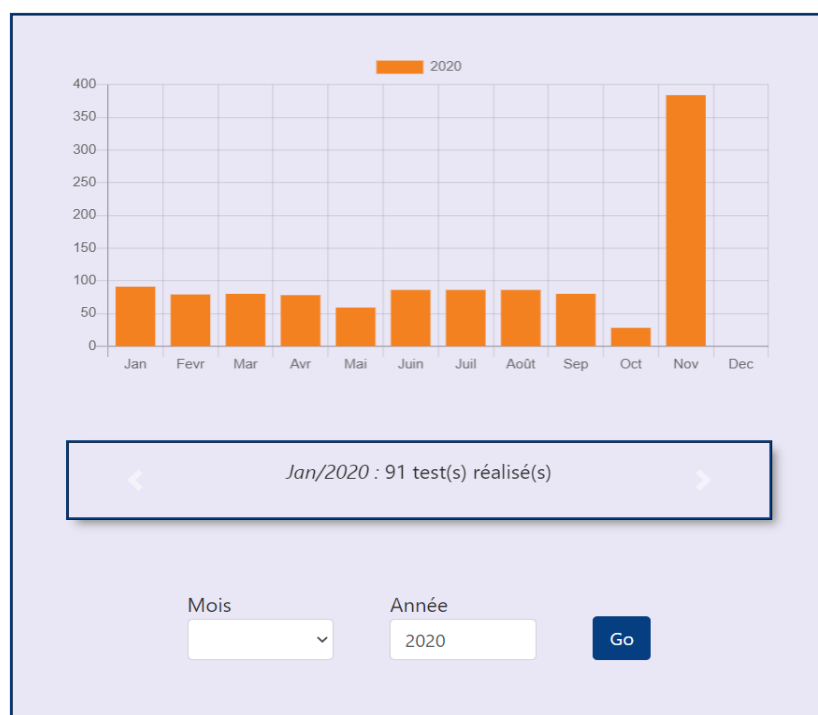
49 //ici est déterminé dans un tableau les réponses possibles aux inputs
50 let results = [1,2,3,4]
51
52 //initialisation d'une boucle sur la collection input afin de donner une valeur à chacun
53 for (let i=0; i< input.length; i++){
54 //on détermine indexTab, un nombre aléatoire entre 0 et, comme borne maximale exclue, la taille du tableau results ([0, result.lenght])
55 let indexTab = Math.floor(Math.random() * results.length)
56 //et on donne sa valeur à input[i]..
57 input[i].value = results[indexTab]
58 //..on supprime la valeur trouvée du tableau de résultats possibles afin qu'elle ne se répète pas
59 results.splice(indexTab,1)
60 }
61 })()

```

Des fonctions PHP m'ont été très utiles tout au long du projet, comme par exemple ici pour vérifier le traitement des réponses et donc avant même cela, leur bonne réception, l'utilisation de `print_r($_POST)` fut primordiale. Son utilisation ainsi que `var_dump()` furent dans la globalité du projet très utilisées me permettant de procéder à des tests et à mieux comprendre certaines de mes erreurs et donc de corriger mon code.

COTE ADMINISTRATEUR :

Du côté de l'interface administrateur, la particularité des tests à effectuer se trouvait dans la page de statistiques.



Exemple de statistiques sur l'année 2020

Il fallait du côté serveur à l'aide d'un select récupérer le nombre de dates entre deux bornes et ensuite du côté client les interpréter et les manipuler pour les intégrer dans un graphique.

Du côté programmation Javascript, l'utilisation du console.log() à différents niveaux m'a accompagnait tout au long du travail réalisé. Ce qui m'a permis de comprendre les données récupérées et de rectifier mon code dès que cela était nécessaire et d'avancer.

Sachant que l'affichage des statistiques peut représenter le nombre de questionnaires réalisés soit, chaque jour dans un mois d'une année, soit, chaque mois dans une année. Il était donc très important pour réaliser des tests sur cette partie du projet, d'avoir des données à interpréter, et ainsi pouvoir connaître et obtenir des statistiques sur par exemple l'année 2020 ou encore le mois de février 2019.

Afin de pouvoir donc travailler dans ce sens, j'ai programmé une fonction PHP, qui, à la manière des fixtures dans Symfony, m'a permis de créer rapidement dans la table « date_of_test » des données fictives et manipulables (voir Annexe 8).

Veille effectuée sur les vulnérabilités de sécurité

L'aspect sécurité du projet nécessitait une attention toute particulière à la partie administrateur. En effet Mr Godiveau souhaitant pouvoir depuis une interface modifier, créer ou supprimer les différents composants du site, cela offrait potentiellement des failles de sécurité dans l'application telles que présence malveillante depuis l'interface administeur, Cross Side Scripting (XSS) ou encore les injections SQL.

SECURISER L'ACCES A L'INTERFACE ADMINISTRATEUR :

La première chose à penser fut bien évidemment de permettre la création de profils administrateur, donc d'une table « admin » pour les stocker et aussi d'un côté, une page de création administrateur sécurisée avec identifiant et mot de passe crypté en base de données à l'aide de la fonction password_hash(\$password, PASSWORD_DEFAULT) et de l'autre côté une connexion sécurisée par la fonction password_verify() prenant 2 paramètres, le mot de passe entré dans le formulaire de connexion et celui qui appartient à l'admin dont l'adresse mail correspond à celle renseignée dans ce même formulaire, si les deux correspondent alors la connexion est traitée.

Cette méthode m'a donc permis d'enregistrer l'ID admin dans \$_SESSION et ainsi systématiquement pour chaque page, avant l'affichage ou le traitement de données, utilisée la logique suivante : « Si la clé valeur \$_SESSION[« id_admin »] est valide et si elle permet la récupération d'un admin valide alors le traitement/l'affichage peut se faire.

De plus une hiérarchie des administrateurs a été mise en place, donnant place à deux rôles possibles, SUPER_ADMIN et ADMIN. Le but étant de ne permettre la gestion des administrateurs et donc l'accès à cette partie de l'interface uniquement à un SUPER_ADMIN. Pour cela une colonne « rôle » existe dans la table « admin » ayant comme valeur « 1 » pour un SUPER_ADMIN et « 0 » pour un ADMIN. Sur la page « admins.php » ainsi que ses pages de traitements, il m'a donc suffi après vérification d'un admin que celui-ci possède bien « 1 » comme valeur de colonne « rôle ».

Sachant que suivant le rôle, que ce soit un simple utilisateur, un SUPER_ADMIN ou un ADMIN, cette vérification me permet également d'afficher un menu qui diffère en fonction des droits de l'utilisateur.



Accueil Questionnaire Nous contacter

Menu visiteur



Accueil Questionnaire Admin ▾ Se déconnecter

Menu rôle ADMIN



Accueil Questionnaire Admin ▾ Se déconnecter

Menu rôle SUPER_ADMIN

CROSS SIDE SCRIPTING (XSS) :

Afin de prévenir ce type de faille et donc d'éviter l'injection de contenu à partir des formulaires présents dans les différentes pages du site, la fonction `htmlspecialchars()` est systématiquement utilisée sur les variables contenant les entrées des champs des formulaires, permettant de convertir les caractères spéciaux. Cette fonction est utilisée au sein d'une fonction `cleanData()`.

```
3 //Retourne une donnée "nettoyée" c'est à dire :
4 //- sans espace au début et à la fin de la chaîne
5 | //- en supprimant les antislashes
6 | //- et convertissant les caractères spéciaux
7 function cleanData($data){
8     $data = trim($data);
9     $data = stripslashes($data);
10    $data = htmlspecialchars($data);
11
12    return $data;
13 }
```

Cette vérification est d'autant plus importante que l'interface administrateur peut effectuer différentes modifications sur la base de données.

Sur ce point il fallait permettre également à l'administrateur de pouvoir modifier si besoin l'architecture ou la mise en forme des données.

C'est le cas par exemple de l'instruction n° 2 qui apparaît sous la forme d'une liste à puces, comme ceci :

2. Pour chacune des 12 situations, il faut :

- mettre un chiffre dans les 4 cases (de 1 à 4)
- utiliser chaque fois un chiffre différent
- utiliser les quatre chiffres

questionnaire.php

Il fallait donc, d'un côté pour le visiteur permettre la mise en page HTML de s'exécuter correctement, comme sur l'exemple ci-dessus, et du côté administrateur permettre lors d'une éventuelle modification ou création la possibilité de retrouver de manière visuelle le signalement des balises et pouvoir si besoin les modifier.

Ordre d'apparition

2

Contenu

Pour chacune des 12 situations, il faut :

- mettre un chiffre dans les 4 cases (de 1 à 4)
- utiliser chaque fois un chiffre différent
- utiliser les quatre chiffres

Enregistrer

updateInstruction.php

La fonction `html_entity_decode($variable, ENT_HTML5)` est donc utilisée à chaque lecture nécessaire des données faisant le chemin inverse de la fonction `htmlspecialchars()` permettant la lecture des balises HTML et donc la mise en page et prenant comme 1^{er} argument la donnée voulue, le 2^{ème} étant le type de document devant être utilisé.

Après différents tests, j'ai pu me rendre compte que de cette façon, la fonction permet d'éviter la lecture de certaines balises `<script></script>`, celles dont les chaînes se trouvaient entourées de double-guillemets mais pas celles entourées de guillemets simples. Après réflexion sur plusieurs possibilités comme l'utilisation de `preg_replace` à différents niveaux mais qui ne donneraient pas pleinement satisfaction, j'ai opté pour `preg_match()`. Partant du principe qu'aucune balise `<script></script>` ne peut se trouver dans la donnée envoyée, si après vérification cela s'avère pourtant être le cas, le processus de traitement de donnée est tout simplement bloqué et une redirection se ferait sur la page d'accueil.

Ainsi le chemin des données peut se faire de manière sécurisée. Il crypte les caractères spéciaux en base de données et à l'inverse permet de lire les balises nécessaires à la mise en page de la donnée sans pour autant permettre le contenu d'une balise `<script></script>` d'être exécuté.

INJECTIONS SQL :

Pour prévenir ce type de faille et protéger les requêtes, celles-ci sont préparées en utilisant la fonction `prepare()` rendant impossible, si bien utilisée, les injections SQL.

PROTECTION DE DONNEES SENSIBLES :

Afin d'éviter de compromettre les données de connexion à la base de données (nom de la base, identifiant et mot de passe) en les stockant sur « git » par exemple, elles sont séparées du reste de la programmation et incluse dans un fichier « .gitignore ».

Les sites consultés :

- <https://www.php.net/manual/fr/index.php> (pour toutes les fonctions et failles évoquées)
- <https://owasp.org/www-community/attacks/>
- <https://stackoverflow.com/questions/7130867/remove-script-tag-from-html-content>
- <https://stackoverflow.com/questions/8218230/php-domdocument-loadhtml-not-encoding-utf-8-correctly>
- <https://www.aurone.com/5-mesures-de-securite-php-implementer>
- https://fr.wikipedia.org/wiki/Cross-site_scripting

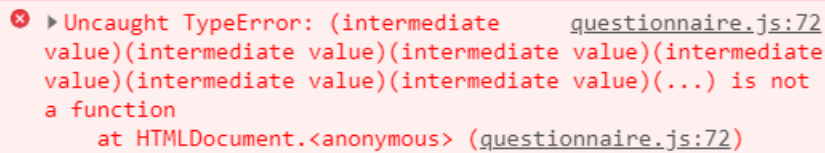
Recherche effectuée sur site anglophone et sa traduction

SITUATION AYANT REQUIS LA RECHERCHE :

Durant la conception du projet, sur la partie front du questionnaire, je cherchais des solutions pour permettre d'obtenir un questionnaire répondant le plus possible aux consignes de remplissage des réponses et plus précisément le fait que pour chaque question, les 4 réponses possibles soient classées de 1 à 4, chaque valeur devant être unique pour que le questionnaire soit conforme, et donc sans qu'il existe la possibilité de doubler l'une de ces notations au sein d'une même question.

Il s'agissait d'un contrôle qui n'était pas une priorité pour Mr Godiveau, c'est donc une interrogation sur laquelle je me suis penché lorsque le projet était déjà fonctionnel et donc programmé en grande partie.

Plusieurs idées se sont succédées, mais c'est lorsque j'ai voulu tester une fonction auto-invoquée qui vérifierait que pour chaque question le tableau des valeurs des réponses renvoyé était bien égal à [1, 2, 3, 4] qu'une erreur que je ne connaissais pas encore est survenue dans la console...



```
✖ ▶ Uncaught TypeError: (intermediate value)(intermediate value)(intermediate value)(intermediate value)(intermediate value)(...) is not a function
    at HTMLDocument.<anonymous> (questionnaire.js:72)
```

A ce stade, une fonction auto-invoquée, conçue pour mes phases de test, était déjà présente dans le script et fonctionnait très bien donc je ne comprenais pas bien pourquoi celle-ci posée problème, du moins sur le moment.

C'est à l'adresse suivante que j'ai pu trouver des réponses :

<https://stackoverflow.com/questions/42036349/uncaught-typeerror-intermediate-value-is-not-a-function>

TEXTE EN ANGLAIS :

The error is a result of the missing semicolon on the third line:

```
window.Glog = function(msg) {
  console.log(msg);
}; // <--- Add this semicolon
(function(win) {
  // ...})(window);
```

The ECMAScript specification has specific rules for automatic semicolon insertion, however in this case a semicolon isn't automatically inserted because the parenthesised expression that begins on the next line can be interpreted as an argument list for a function call.

This means that without that semicolon, the anonymous window.Glog function was being invoked with a function as the msg parameter, followed by (window) which was subsequently attempting to invoke whatever was returned.

This is how the code was being interpreted:

```
window.Glog = function(msg) {
  console.log(msg);
}(function(win) {
  // ...
})(window);
```

To make semicolon rules simple

Every line that begins with a (, [, `, or any operator (/, +, - are the only valid ones), must begin with a semicolon.

```
func()
;[0].concat(myarr).forEach(func)
;(myarr).forEach(func)
;`hello`.forEach(func)
;/hello/.exec(str)
;+0
;-0
```

This prevents a

```
func()[0].concat(myarr).forEach(func)(myarr).forEach(func)`hello`.forEach(func)/hell
o/.forEach(func)+0-0
```

TRADUCTION :

L'erreur est le résultat de l'oubli d'un point-virgule à la troisième ligne :

```
window.Glog = function(msg) {
  console.log(msg);
}; // <--- Ajout d'un point-virgule
(function(win) {
  // ...})(window);
```

La spécification ECMAScript a des règles spécifiques pour l'insertion automatique de point-virgule, cependant dans ce cas un point-virgule n'est pas automatiquement inséré parce que l'expression entre parenthèses qui commence à la ligne suivante peut être interprétée comme une liste d'arguments pour un appel de fonction.

Cela signifie que sans ce point-virgule la fonction anonyme « window.Glog » était invoquée avec une fonction « msg » comme paramètre, suivie par « window » qui par la suite tentait d'invoquer tout ce qui était retourné.

Voici comment le code était interprété :

```
window.Glog = function(msg) {
  console.log(msg);
}(function(win) {
  // ...
})(window);
```

Pour Simplifier les règles du point-virgule

Chaque ligne qui commence par un (, [, ', ou tout opérateur (/, +, - sont les seuls valides) doit commencer avec un point-virgule

```
func()
;[0].concat(myarr).forEach(func)
;(myarr).forEach(func)
```

```
;'hello`.forEach(func)
;/hello/.exec(str)
;+0
;-0
```

Cela empêche un :

```
func()[0].concat(myarr).forEach(func)(myarr).forEach(func)`hello`.forEach(func)/hell
o/.forEach(func)+0-0
```

Commentaire sur cette recherche :

En effet il m'a fallu simplement rajouter un point-virgule au départ de la fonction auto-invoquée qui affichait l'erreur dans la console pour que celle-ci s'exécute normalement. Ce n'est en définitive pas la solution conservée dans le code final mais ce fut tout de même une recherche intéressante dans la conception du projet.

Conclusion :

J'ai pu dans les délais prévus fournir le projet dans une version répondant aux attentes du cahier des charges et de ses impératifs.

Le remplissage du questionnaire est simple et rapide, son traitement fiable. L'espace administrateur intègre une gestion des principaux éléments qui compose le site, ainsi qu'un affichage de statistiques comme demandé par Mr Godiveau. Les couleurs sont directement liées au site Absys-Formation comme convenu et le rendu est adaptable aux différents écrans du marché.

J'adresse des remerciements tout particuliers à Mr Godiveau pour la confiance, ses conseils mais aussi l'autonomie accordée. Ce projet partant de son idée, présentant un réel intérêt dans ses projets d'avenir, m'a permis de travailler les deux modules nécessaires à l'obtention du diplôme, mais surtout de faire preuve d'analyse, de réflexion, d'échanges, de gestion de temps, de recherches et documentations.

D'autres remerciements sont bien évidemment adressés à l'organisme Alkas-Formation pour leur gestion de la formation malgré un contexte sanitaire particulier. Merci donc à Mlle Élodie Dubus pour la gestion globale de la formation et à Mr Thomas Muzy d'avoir partagé son savoir, son expérience et son plaisir de la programmation.

Annexes :

Annexe 1 (page 11)

```
237 //Pour renvoyer une variable contenant un objet de question
238 public function getQuestion(int $id){
239     $sqlQuery = (
240         "SELECT *
241         FROM question
242         WHERE question.id = :id");
243
244     $stmt = $this->_handler->prepare($sqlQuery);
245     $stmt->execute(
246         [
247             ':id' => $id
248         ]
249     );
250     $res = $stmt->fetch(PDO::FETCH_OBJ);
251     $question = new Question($res->id, $res->content);
252
253     return $question;
254 }
```

Annexe 2 (page 11)

```
299 //Pour renvoyer un tableau d'objets de toutes les réponses d'une question
300 public function getAnswersForOneQuestion($id){
301
302     $sqlQuery = (
303         "SELECT * FROM question
304         INNER JOIN answer ON question_id = question.id
305         WHERE question.id = :id");
306
307
308     $stmt = $this->_handler->prepare($sqlQuery);
309     $stmt->execute(
310         [
311             ':id' => $id
312         ]
313     );
314     $answers = [];
315     $res = $stmt->fetchAll(PDO::FETCH_OBJ);
316     foreach ($res as $answer){
317         array_push(
318             $answers,
319             new Answer(
320                 $answer->id,
321                 $answer->numberInList,
322                 $answer->orderOfAppearance,
323                 $answer->content,
324                 $answer->category));
325     }
326     return $answers;
327 }
```

Annexe 3 (page 17)

```
42 //afin d'obtenir tous les 5mm un repère de graduation visible
43 // 2 fonctions permettent en partant du point central
44 //de définir ces repères
45 //les paramètres :
46 //1 - Correspond au nombre de graduation en partant du point central
47 //2 - Le point de départ de la ligne par rapport au bord gauche du canvas
48 //et le point d'arrivée par rapport au bord gauche.
49 //3 - Le point de départ de la ligne par rapport au bord haut du canvas
50 //4 - Le point d'arrivée de la ligne par rapport au bord haut du canvas
51 //5 - La différence en pixels apportée à chaque tour au 2ème paramètre de la fonction
52 function graduationX(tour, departX, y1, y2, diff){
53     for (let i=0; i<tour; i++){
54         //Pour chaque index divisé par 5 il reste 0, une ligne est tracée
55         if (i%5 === 0){
56             drawLine(departX, y1, departX, y2, 1, "black")
57         }
58         departX += diff
59     }
60 }
61 //les paramètres :
62 //1 - Correspond au nombre de graduation en partant du point central
63 //2 - Le point de départ de la ligne par rapport au bord gauche du canvas
64 //3 - Le point de départ de la ligne par rapport au bord haut du canvas
65 //et le point d'arrivée par rapport au bord haut
66 //4 - Le point d'arrivée de la ligne par rapport au bord gauche du canvas
67 //5 - La différence en pixels apportée à chaque tour au 3ème paramètre de la fonction
68 function graduationY(tour, x1, departY, x2, diff){
69     for (let i=0; i<tour; i++){
70         if (i%5 === 0){
71             drawLine(x1, departY, x2, departY, 1, "black")
72         }
73         departY += diff;
74     }
75 }
```

```

80     if (isset($_POST['month']) && isset($_POST['year'])) {
81
82         if (!empty($_POST['year'])) {
83
84             if (!empty($_POST['month'])) {
85                 //si un mois est renseigné alors création d'une date au format YYYY-MM-01
86                 $date = $_POST['year']."-".$_POST['month']."-01";
87                 //afin d'obtenir la timestamp de la date souhaitée
88                 $timestamp = strtotime($date);
89                 //d'en récupérer le titre (avec l'abréviation du mois)
90                 //et afficher dans le carrousel qui sera également
91                 //repris comme titre du graphique avec Chart.js
92                 $labelGraph = date("M-Y", $timestamp);
93                 //récupération du nombre de jours maximum du mois de l'année demandé
94                 $maxDays = date("t", mktime(0, 0, 0, $_POST['month'], 1, $_POST['year']));
95                 //puis récupération du total des entrées jour par jour dans le tableau $datas
96                 //prenant comme paramètres la date au format YYYY-MM-01 et le maximum de jours dans MM
97                 $datas = graphDataMonth($date, $maxDays);
98
99             } else {
100                 //sinon récupération de l'année pour une date au format YYYY-01-01
101                 $date = $_POST['year']."-01-01";
102                 //et récupération du total des entrées mois par mois dans un tableau
103                 $datas = graphDataYear($date);
104                 //ici le titre du carrousel et du graphique sera l'année renseignée
105                 $labelGraph = $_POST['year'];
106             }

```

Annexe 5 (page 20)

```
15 //Retourne un tableau associatif du total des entrées existantes mois par mois
16 function graphDataYear(string $date){
17     //tableau permettant l'abréviation en français des mois de l'année
18     $months = ["Jan", "Fevr", "Mar", "Avr", "Mai", "Juin", "Juil", "Août", "Sep", "Oct", "Nov", "Dec"];
19     //récupération de la timestamp
20     $timestamp = strtotime($date);
21     //récupération de l'année
22     $year = date("Y", $timestamp);
23     //le tableau va renfermer les 12 données et sera retourné
24     $datas = [];
25     //unité nécessaire au bon fonctionnement du carrousel
26     $number = 1;
27     //boucle initiée dont chaque tour correspond à un mois dans l'année
28     for ($i=1 ; $i<=12; $i++){
29         //si $i est inférieur à 10 alors rajout d'un zéro pour obtenir un format de date conforme à l'arrivée
30         if ($i <= 9){
31             $i = '0' . $i;
32         }
33         $number++;
34         //récupération du nombre maximum de jours dans le mois du tour de boucle en cours
35         $maxDay = date("t", mktime(0,0,0,$i,1,$year));
36         //deux variables servant de borne de départ et d'arrivée dans la récupération du nombre de données
37         $firstDay = date('Y-'. $i .'-01', strtotime($date));
38         $lastDay = date('Y-'. $i .'-'. $maxDay , strtotime($date));
39         // $count va renfermer le nombre d'entrées existantes du 1er jour au dernier jour du mois
40         $count = getIsalemDatabaseHandler()->getNumberOfDates($firstDay, $lastDay);
41         //récupération de l'abréviation en français du mois du tour
42         $month = $months[$i-1];
43         //ajout au tableau de l'objet dont les valeurs des propriétés seront utilisées pour la création du graphique
44         //et du carrousel
45         array_push($datas, (object)['slide'=>$number, 'label'=>$month, 'dateComplement'=>$year, 'data_number'=>$count]);
46     }
47     return $datas;
48 }
```


Annexe 6 (page 20)

```
50 //Retourne un tableau associatif du total des entrées existantes jour par jour
51 //2 paramètres : la date au format YYYY-MM-JJ et le nombre de jours dans MM
52 function graphDataMonth(string $date, int $maxDays){
53     //récupération de la timestamp
54     $timestamp = strtotime($date);
55     //permettant la récupération de l'année
56     $year = date("Y", $timestamp);
57     //et du mois
58     $month = date("m", $timestamp);
59     //la date réécrite au format MM/YYYY
60     $dateComplement = $month.'/'.$year;
61     //initialisation du tableau qui sera retournée avec les données à interprétées
62     $datas = [];
63     //la variable number utilisée pour numéroté les pages du carrousel Bootstrap
64     //permettant leur affichage
65     $number = 1;
66     //boucle initiée partant du 1er jour du mois demandé au dernier jour
67     for ($i=1 ; $i<=$maxDays; $i++){
68         //pour un format de jour valide, rajout d'un 0 si l'index est inférieur à 10
69         if ($i <= 9){
70             $i = '0' . $i;
71         }
72         $number++;
73         //remplacement du jour initial de $date par l'index du tour
74         $dateData = date('Y-m-'. $i, strtotime($date));
75         //récupération du nombre d'entrées de la table date_of_test identiques à $dateData
76         $count = getIsalemDatabaseHandler()->getDateData($dateData);
77         //la donnée est ajoutée sous forme d'objet dans le tableau à retourner
78         //les propriétés de chaque objets sont nécessaires au fonctionnement du carrousel et de la création du graphique
79         array_push($datas, (object)['slide'=>$number, 'label'=>$i, 'dateComplement'=>$dateComplement, "data_number"=>$count]);
80     }
81     return $datas;
82 }
```

age : over18

Données reçues et classées :

Array ([0] => Ab1 => 4 [1] => Ab6 => 1 [2] => Ab11 => 4 [3] => Ab14 => 2 [4] => Ab19 => 2 [5] => Ab21 => 4 [6] => Ab26 => 1 [7] => Ab32 => 1 [8] => Ab34 => 1 [9] => Ab37 => 3 [10] => Ab42 => 4 [11] => Ab48 => 3)

Array ([0] => Ai3 => 2 [1] => Ai8 => 3 [2] => Ai9 => 1 [3] => Ai16 => 3 [4] => Ai18 => 3 [5] => Ai23 => 1 [6] => Ai28 => 3 [7] => Ai29 => 2 [8] => Ai35 => 3 [9] => Ai38 => 4 [10] => Ai43 => 3 [11] => Ai46 => 1)

Array ([0] => Ac2 => 1 [1] => Ac7 => 4 [2] => Ac12 => 3 [3] => Ac13 => 4 [4] => Ac20 => 1 [5] => Ac22 => 2 [6] => Ac27 => 4 [7] => Ac30 => 4 [8] => Ac33 => 4 [9] => Ac39 => 2 [10] => Ac44 => 1 [11] => Ac45 => 4)

Array ([0] => Ar4 => 3 [1] => Ar5 => 2 [2] => Ar10 => 2 [3] => Ar15 => 1 [4] => Ar17 => 4 [5] => Ar24 => 3 [6] => Ar25 => 2 [7] => Ar31 => 3 [8] => Ar36 => 2 [9] => Ar40 => 1 [10] => Ar41 => 2 [11] => Ar47 => 2)

Les 4 tableaux après redirection des données et avant le calcul de x et y :

Array ([0] => 4 [1] => 1 [2] => 4 [3] => 2 [4] => 2 [5] => 4 [6] => 1 [7] => 1 [8] => 3 [10] => 4 [11] => 3) total Ab = 30

Array ([0] => 2 [1] => 3 [2] => 1 [3] => 3 [4] => 3 [5] => 1 [6] => 3 [7] => 2 [8] => 3 [9] => 4 [10] => 3 [11] => 1) total Ai = 29

Array ([0] => 1 [1] => 4 [2] => 3 [3] => 4 [4] => 1 [5] => 2 [6] => 4 [7] => 4 [8] => 4 [9] => 2 [10] => 1 [11] => 4) total Ac = 34

Array ([0] => 3 [1] => 2 [2] => 2 [3] => 1 [4] => 4 [5] => 3 [6] => 2 [7] => 3 [8] => 2 [9] => 1 [10] => 2 [11] => 2) total Ar = 27

calcul de x et y si plus de 18 ans :

x = 29 - 30 - 8

y = 34 - 27 - 5

Les valeurs de x et y renvoyées pour résultat et graphique :

x = 9 et y = 2

Chaine permettant la récupération du résultat :

result = -xy

Profil obtenu avec la chaîne de caractère :

Intuitif Reflexif

```

5 //génère des dates aléatoires dans la table date_of_test
6 //les 3 paramètres sont : la borne minimale et la borne maximale
7 //(ces deux premiers paramètres aux formats AAAA-MM-JJ)
8 //et le nombre d'entrées souhaitées dans la table
9 function test_createDates($date1, $date2, $x){
10     //les données actuelles de date_of_test sont d'abord effacées
11     getIsalemDatabaseHandler()->deleteDatas('date_of_test');
12     //les bornes min et max sont transformées en timestamp
13     $start = strtotime($date1);
14     $end = strtotime($date2);
15     //une boucle for prenant comme fin le 3ème paramètre de la fonction
16     //est initialisée
17     for ($i = 0; $i < $x; $i++){
18         //pour chaque tour une timestamp aléatoire est générée entre les
19         //deux bornes
20         $timestamp = mt_rand($start, $end);
21         //puis celle-ci est traduite dans un format AAAA-MM-JJ
22         $date = date("Y-m-d", $timestamp);
23         //pour ensuite servir de paramètre à une fonction d'insertion de date
24         //dans la base de données
25         getIsalemDatabaseHandler()->createDate($date);
26     }
27 }

```