



BTS SIO – Option SLAM

Documentation d'épreuve

Rédacteur	Version	Date	Nb pages
Johann DIETRICH	1.3	14/04/2023	20

**Création d'une page de pseudo quiz a
plusieurs choix pour tester le niveau
d'anglais.**

SOMMAIRE

1	CAHIER DES CHARGES	3
1.1	Introduction.....	3
1.2	Expression fonctionnelle du besoin	4
1.3	Contraintes.....	5
1.4	Gestion des droits d'accès.....	6
2	DESCRIPTION DES ENVIRONNEMENTS	7
3	METHODOLOGIE.....	8
3.1	Méthodologie et versioning.....	8
3.2	Gestion des tests de la solution	8
3.3	Gestion de projet	9
4	MISE EN OEUVRE	10
4.1	Création de la page du quiz.....	10
4.2	Page d'insertion de mots anglais et leur équivalent français	14
4.3	Page d'ajout de mots français	15
5	GESTION DE LA MAINTENANCE (CORRECTIVE / EVOLUTIVE).....	19
5.1	Mise à jour de la documentation du SI.....	19
5.2	Procédure de correction d'un dysfonctionnement	19
5.3	Gestion des tests de mise à jour.....	19
6	BILAN DU PROJET.....	20
6.1	Validation des exigences point par point.....	20
6.2	Axes d'amélioration	20
6.3	Compétences acquises	20

1 Cahier des charges

1.1 Introduction

Type de mission
Mission effectuée en indépendant, durant d temps libre
Demande du client
Besoin d'un petit quiz permettant d'améliorer le vocabulaire en anglais.
Budget disponible
Aucun budget disponible, mais un limitation de temps due a l'API étant gratuite uniquement un nombre limité de temps
Outils disponibles
VSCode

1.2 Expression fonctionnelle du besoin

Liste des fonctionnalités attendues :

Front office

Page de quiz

- Affiche le quiz et le score

Page d'ajout de mots

- Ajout des mots français et leur traduction en anglais

Page d'ajout de mots français :

- Ajout de mots anglais et français en connaissant uniquement l'équivalent français

1.3 Contraintes

Générales

Pas de réelles contraintes de temps, ni de budget. Seule contrainte de temps sera la durée limitée de l'essai gratuit de l'API

Juridiques

Aucune donnée sensible n'est stockée donc aucun problème juridique

Techniques

PHP, JS, et Bootstrap conseillé

Ergonomique

Simplement doit être suffisamment lisible

1.4 Gestion des droits d'accès

Administrateur
Peut faire le quiz et rajouter des mots

Utilisateurs
Peut faire le quiz et rajouter des mots

Visiteurs
Peut faire le quiz et rajouter des mots

2 Description des environnements

Environnement de développement

Le tout à été réalisé en local avec un code PHP live server, simplement afin de vérifier le fonctionnement du programme.

3 Méthodologie

3.1 Méthodologie et versioning

GitHub sera utilisé pour le Versionning tout simplement car il s'agit de l'outil le plus efficace pour cela.

3.2 Gestion des tests de la solution

expliquer comment vous gérez les tests

unitaires

Tout simplement, un test de vérification en essayant le dit quiz permettant de voir si celui-ci a des erreurs

3.3 Gestion de projet

3.3.1 Budget


Aucun réel budget car le tout est en phase de test, mais en cas d'utilisation professionnelle, un budget devra être alloué pour l'utilisation continue de l'API.

4 Mise en oeuvre

Pour la mise en place de ce petit quiz, il nous faudra plusieurs choses. Déjà une base de données permettant de stocker les mots que l'on veut, une page dans laquelle se fera le quiz avec les réponses multiples, et une page permettant d'ajouter de nouveaux mots de manière à utiliser une API. Donc commençons déjà par la première partie.

4.1 Création de la page du quiz

Pour commencer donc, il nous faut créer la base de données et la table qui servira à stocker les mots. Dans notre cas, la base de données « quiz » contient la table « mots » que voici :

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
<input type="checkbox"/> 1	id 	int		UNSIGNED	Non	Aucun(e)		AUTO_INCREMENT
<input type="checkbox"/> 2	mot_en_anglais	varchar(255)	utf8mb4_0900_ai_ci		Non	Aucun(e)		
<input type="checkbox"/> 3	mot_en_francais	varchar(255)	utf8mb4_0900_ai_ci		Non	Aucun(e)		

Permettant donc de stocker les mots anglais et leur version en français.

A partir de là, il nous faut savoir la mise en forme que l'on voudra pour notre page de quiz. J'ai opté pour quelque chose de classique à base d'un formulaire avec quatre réponses possibles grâce à des input.

Le code du formulaire ressemble à ceci :

```
<div class="container">
  <h3 class="text-center mb-4">Traduisez le mot suivant en français:</h3>
  <div class="card">
    <div class="card-body">
      <p class="card-text"><?php echo $mot_en_anglais; ?></p>
      <form method="post">
        <?php foreach ($reponses as $reponse) { ?>
          <div class="form-check">
            <input class="form-check-input" type="radio" name="choice" id="<?php echo $reponse; ?>" value="<?php echo $reponse; ?>">
            <label class="form-check-label" for="<?php echo $reponse; ?>">
              <?php echo $reponse; ?>
            </label>
          </div>
        <?php } ?>
        <button type="submit" name="submit" class="btn btn-primary mt-3">Répondre</button>
      </form>
    </div>
  </div>
  <form method="post" action="reset_score.php">
    <button type="submit" name="reset" class="btn btn-secondary">Réinitialiser</button>
```

```
</form>
</div>
```

Ce qui donne ceci

Traduisez le mot suivant en français:

table

- ☐ oiseau
- ☒ table
- ☐ eau
- ☐ basketball

Répondre

Il est important a noter que dans celui-ci plusieurs variables sont utilisées, celles-ci étant récupérées de la base de données avec ce code ci :

```
if (isset($_SESSION['mot_en_anglais'])) {
    $mot_en_anglais = $_SESSION['mot_en_anglais'];
    $score = $_SESSION['score'];
} else {
    // Sélectionner un mot au hasard dans la base de données si le mot en anglais actuel
    n'est pas encore défini dans la session
    $query = "SELECT * FROM mots ORDER BY RAND() LIMIT 1";
    $result = $db->query($query);
    $row = $result->fetch(PDO::FETCH_ASSOC);
    $mot_en_anglais = $row['mot_en_anglais'];
    // Stocker le mot en anglais actuel dans une variable de session
    $_SESSION['mot_en_anglais'] = $mot_en_anglais;
    // Initialiser le score à 0 si ce n'est pas déjà fait
    if (!isset($_SESSION['score'])) {
        $_SESSION['score'] = 0;
    }
    $score = $_SESSION['score'];
}

// Récupérer les traductions en français
$reponses = array();

$query = "SELECT * FROM mots WHERE mot_en_anglais = '{$mot_en_anglais}'";
$result = $db->query($query);
$row = $result->fetch(PDO::FETCH_ASSOC);
$reponses[] = $row['mot_en_francais'];

$query = "SELECT * FROM mots WHERE mot_en_anglais != '{$mot_en_anglais}' ORDER BY RAND()
LIMIT 3";
$result = $db->query($query);
while ($row2 = $result->fetch(PDO::FETCH_ASSOC)) {
    $reponses[] = $row2['mot_en_francais'];
}

// Mélanger les réponses
```

```
shuffle($reponses);  
?>
```

Un très grand morceau de programme simplement pour récupérer un mot anglais de la base de données de manière aléatoire et de récupérer la bonne réponse liée a celui-ci. Puis de récupérer avec la bonne réponse 3 réponses aléatoires stockées afin de rendre l'entièreté des réponses aléatoires. Il permet également de commencer une variable session qui enregistrera notre score pour plus tard.

Il est important a noter qu'avant faire des requêtes SQL afin de récupérer des informations de la BDD, il faut mettre cette partie de code avant afin de s'y connecter :

```
<?php  
session_start();  
  
// Connexion à la base de données  
$host = 'localhost';  
$dbname = 'quiz';  
$username = 'root';  
$password = 'aaaa';  
  
$db = new PDO("mysql:host=$host;dbname=$dbname;charset=utf8", $username, $password);
```

Bien sûr, selon la BDD il faut changer le nom, le mot de passe etc...

Maintenant que l'on a la forme, il nous faut le pratique. Pour faire fonctionner celui-ci on se servira d'une autre page et d'un peu de jQuery.

Pour commencer, on va mettre en place du code jQuery qui nous permet d'envoyer le résultat du formulaire sur une autre page qui va s'occuper de vérifier la réponse. Celui-ci ressemble à ça :

```
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>  
<script>  
$(document).ready(function() {  
    $('form').submit(function(event) {  
        event.preventDefault(); // Empêcher le formulaire de se soumettre normalement  
        $.get('verifier_reponse.php?' + $(this).serialize(), function(response) {  
            // Afficher la réponse du serveur dans une alerte  
            alert(response);  
            // Actualiser la page après 1 seconde  
            setTimeout(function() {  
                location.reload();  
            }, 1000);  
        });  
    });  
});  
</script>
```

Celui-ci nous permet donc de vérifier la réponse ET d'actualiser la page une seconde après afin d'avoir une nouvelle question aléatoire.

La page qui est appelée vérifie donc si la réponse est bien liée au mot en anglais.

```
if (isset($_GET['choice'])) {
    $choice = $_GET['choice'];
    if (isset($_SESSION['mot_en_anglais'])) {
        $mot_en_anglais = $_SESSION['mot_en_anglais'];
        $query = "SELECT * FROM mots WHERE mot_en_anglais = '{$mot_en_anglais}' AND
mot_en_francais = '{$choice}'";
        $result = $db->query($query);
        $row = $result->fetch(PDO::FETCH_ASSOC);

        if ($row) {
            // Mettre à jour le score de l'utilisateur
            $_SESSION['score']++;
            $response = "Correct! Votre score est de {$_SESSION['score']}.";
        } else {
            $response = "Incorrect. La réponse correcte était '{$row['mot_en_francais']}'".
Votre score est de {$_SESSION['score']}.";
        }

        // Détruire la variable de session pour le mot en anglais actuel
        unset($_SESSION['mot_en_anglais']);

        echo $response;
    }
}
```

Celle-ci met également à jour le score en y ajoutant un point et en réinitialisant le mot en anglais, permettant donc d'obtenir un autre mot aléatoire pour la prochaine question du quiz.

Après tout cela, il ne nous reste plus qu'une seule chose à rajouter pour notre quiz, un bouton pour réinitialiser le score, en effet, sinon le score ne fera qu'augmenter sans pouvoir se retester.

Pour cela, rien de plus simple. Il nous suffit d'insérer un petit bouton avec un formulaire sous notre quiz avec cela :

```
<form method="post" action="reset_score.php">
    <button type="submit" name="reset" class="btn btn-
secondary">Réinitialiser</button>
</form>
```



Et une page « reset_score.php » qui ne fait que remettre le score à 0 :

```
<?php
session_start();
$_SESSION['score'] == 0;
header('Location: index.php');
exit;
?>
```

Tout ceci ensemble, nous donnant ceci :

Traduisez le mot suivant en français:

computer

- ☐ basketball
- ☐ chien
- ☐ table
- ☐ ordinateur

Répondre

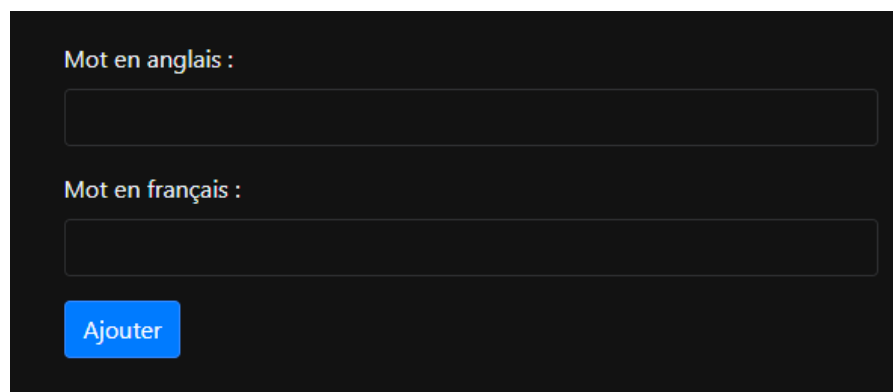
Réinitialiser

4.2 Page d'insertion de mots anglais et leur équivalent français

Page assez simple mais que l'on va réutiliser plus tard. Celle-ci est composée d'un petit formulaire et d'un petit programme permettant d'insérer les données mises dans le formulaire dans la BDD.

```
<div class="container">
  <div class="row justify-content-center">
    <div class="col-lg-6">
      <form method="POST" action="ajouter_mot.php">
        <div class="form-group">
          <label for="mot_en_anglais">Mot en anglais :</label>
          <input type="text" class="form-control" id="mot_en_anglais"
name="mot_en_anglais" required>
        </div>
        <div class="form-group">
          <label for="mot_en_francais">Mot en français :</label>
          <input type="text" class="form-control" id="mot_en_francais"
name="mot_en_francais" required>
        </div>
        <button type="submit" class="btn btn-primary">Ajouter</button>
      </form>
    </div>
  </div>
</div>
```

Le formulaire ressemble à ceci :

A screenshot of a web form on a dark background. It has two text input fields. The first is labeled 'Mot en anglais :' and the second is labeled 'Mot en français :'. Below the second field is a blue button with the text 'Ajouter'.

Celui-ci fait donc appel à une page « ajouter_mot.php » qui celle-ci prend les informations du formulaire et les insère dans la BDD. Le code ressemble donc à cela :

```
// Vérifier si le formulaire a été soumis
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    // Récupérer les valeurs des champs de texte
    $mot_en_anglais = $_POST['mot_en_anglais'];
    $mot_en_francais = $_POST['mot_en_francais'];

    // Vérifier si le mot en anglais existe déjà dans la base de données
    $query = "SELECT * FROM mots WHERE mot_en_anglais = :mot_en_anglais";
    $stmt = $db->prepare($query);
    $stmt->bindParam(':mot_en_anglais', $mot_en_anglais);
    $stmt->execute();
    $result = $stmt->fetch(PDO::FETCH_ASSOC);

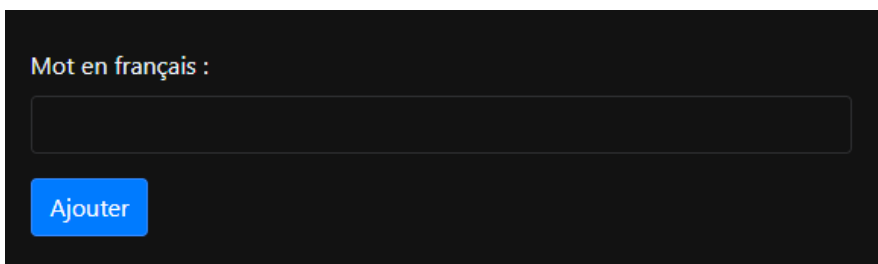
    if ($result) {
        // Le mot en anglais existe déjà, afficher un message d'erreur
        echo "Le mot en anglais existe déjà dans la base de données.";
    } else {
        // Le mot en anglais n'existe pas encore, l'ajouter à la base de données
        $query = "INSERT INTO mots (mot_en_anglais, mot_en_francais) VALUES (:mot_en_anglais, :mot_en_francais)";
        $stmt = $db->prepare($query);
        $stmt->bindParam(':mot_en_anglais', $mot_en_anglais);
        $stmt->bindParam(':mot_en_francais', $mot_en_francais);
        $stmt->execute();

        // Afficher un message de confirmation
        echo "Le mot a été ajouté à la base de données.";
    }
}
```

Celui-ci, comme on peut le voir, a également une sécurité afin d'éviter d'avoir plusieurs fois le même mot anglais dans la BDD. Maintenant se trouvera la partie plus intéressante qui a été basée sur ce même fonctionnement. La traduction automatique avec un formulaire.

4.3 Page d'ajout de mots français

La base de notre page, le formulaire n'est pas très différent donc je ne vais pas vous le montrer, la partie intéressante est la manière dont notre API fonctionne et la manière dont on l'utilise.



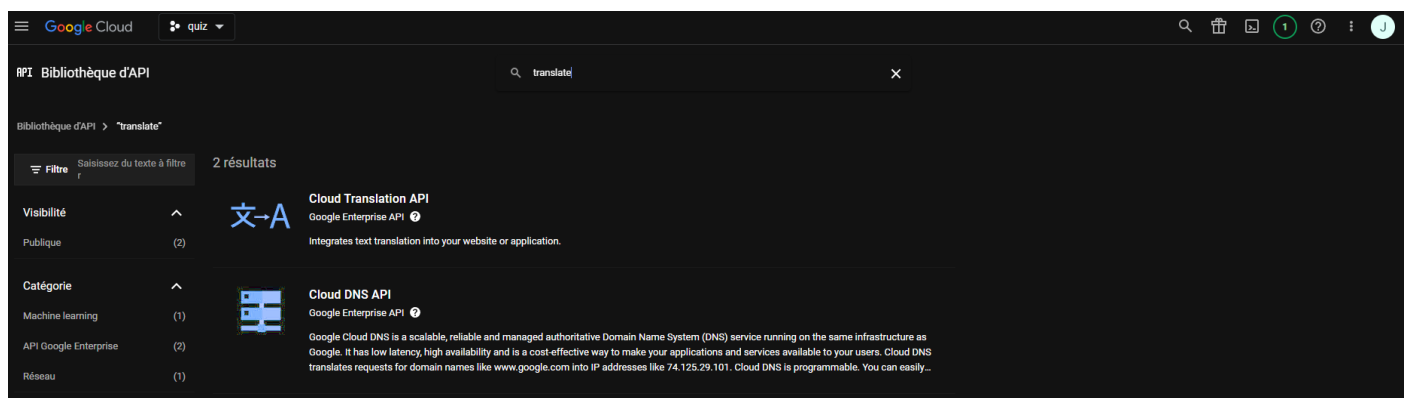
Mot en français :

Notre petit formulaire fait simplement appel à une autre page qui celle-ci utilise notre API.

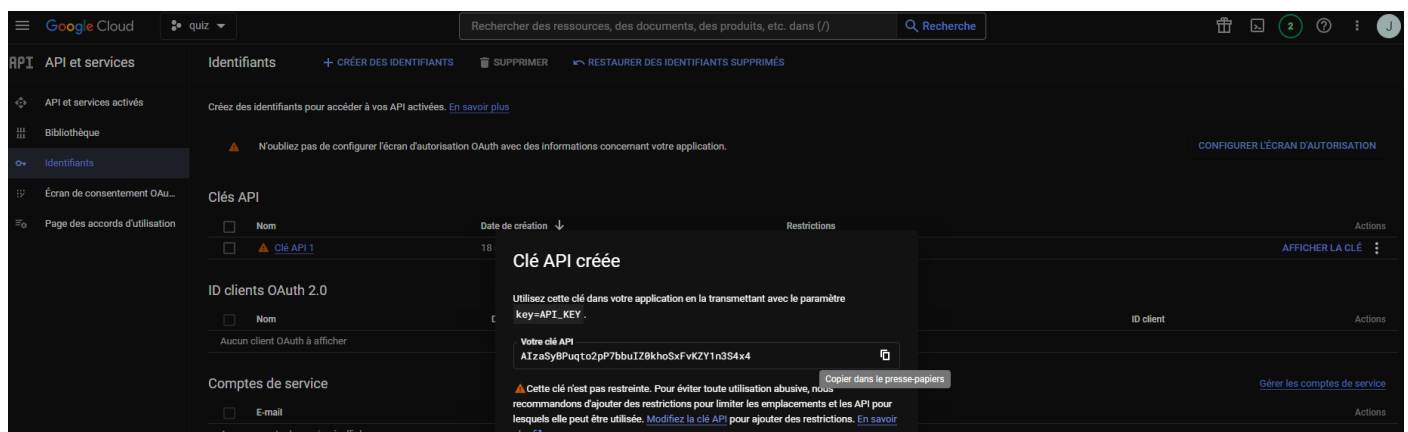
Mais justement, avant de pouvoir se servir de l'API, il nous faut déjà créer un compte capable de l'utiliser (comme c'est le cas pour beaucoup). Dans notre cas, vu que l'on utilise google translate API, il suffit de se connecter a son compte google, de donner les informations nécessaires.

The screenshot shows the 'Étape 1 sur 2 Informations de compte' (Step 1 of 2 Account Information) form in Google Cloud. At the top, it displays the user's profile: 'Johann Dietrich' with email 'jodietpro@gmail.com' and a 'CHANGER DE COMPTE' link. Below this is a 'Pays' (Country) dropdown menu set to 'France'. The next section asks 'Laquelle de ces propositions décrit le mieux votre organisation ou vos besoins ?' (Which of these proposals best describes your organization or needs?) with a 'Please select' dropdown menu set to 'Autre'. A 'Conditions d'utilisation' (Terms of Use) section follows, with a checkbox for 'J'ai lu et j'accepte les Conditions d'utilisation de Google Cloud Platform, les Conditions d'utilisation complémentaires liées à l'essai gratuit, ainsi que les conditions d'utilisation des services et API concernés.' (I have read and accept the Google Cloud Platform Terms of Use, the supplementary terms of use linked to the free trial, as well as the terms of use of the services and APIs concerned.). A note states 'Vous devez cocher cette case pour continuer' (You must check this box to continue). At the bottom is a blue 'CONTINUER' button.

Puis de chercher dans le menu google cloud la bibliothèque d'API. Et dans celle-ci la « cloud translation API » qu'il faut donc activer.



Après l'avoir activé, il faut encore obtenir une clé API qui nous permet donc d'utiliser l'API. Il est important a noter que en situation réelle, il y a la capacité de limiter quels site ont accès a l'utilisation de l'API avec notre compte, mais vu que l'on est en situation de pure test, cela ne sera pas nécessaire.



Maintenant que l'on a toutes les parties prêtes, on peut commencer à créer notre programme.

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {  
    // Récupérer le mot en français saisi dans le formulaire  
    $mot_francais = $_POST['mot_francais'];
```

On commence donc par récupérer les informations du formulaire (le mot en français)

```
// URL de l'API de traduction de Google  
$url = 'https://translation.googleapis.com/language/translate/v2?key=VOTRE_CLE_API';  
  
// Données à envoyer à l'API de traduction  
$data = array(  
    'q' => $mot_francais, // Le mot en français à traduire  
    'source' => 'fr', // La langue source est le français  
    'target' => 'en', // La langue cible est l'anglais  
    'format' => 'text' // Le format du texte est du texte brut  
);
```

On continue en donnant l'URL de l'API avec la clé API que l'on a généré plus tôt et les informations nécessaires pour le bon fonctionnement de l'API (le mot à traduire, les langues et le format).

```
// Options de la requête HTTP à envoyer à l'API de traduction  
$options = array(  
    'http' => array(  
        'header' => "Content-type: application/x-www-form-urlencoded\r\n", // En-tête de la  
        requête  
        'method' => 'POST', // La méthode HTTP à utiliser est POST  
        'content' => http_build_query($data), // Les données à envoyer  
    ),  
);  
  
// Création d'un contexte stream avec les options de la requête  
$context = stream_context_create($options);
```

```
// Envoi de la requête à l'API de traduction et stockage de la réponse  
$result = file_get_contents($url, false, $context);
```

On lui donne les options nécessaires au bon fonctionnement de celle-ci, notamment avec la méthode POST. On la transforme en contexte stream qui permet simplement la bonne lecture des données par l'API. Et l'on fini par tout envoyer à l'API et à stocker le résultat dans notre variable result.

A partir de là, la suite est simple.

```
$json = json_decode($result, true);  
  
// Récupération du mot traduit en anglais à partir du tableau associatif  
$mot_anglais = $json['data']['translations'][0]['translatedText'];  
  
// Insérer le mot en français et le mot en anglais dans la base de données  
$query = "INSERT INTO mots (mot_en_francais, mot_en_anglais) VALUES (:mot_francais,  
:mot_anglais)";  
$stmt = $db->prepare($query);  
$stmt->bindValue(':mot_francais', $mot_francais);  
$stmt->bindValue(':mot_anglais', $mot_anglais);  
$stmt->execute();
```

Comme il nous l'es montré, il suffit de décoder le résultat obtenu afin de récupérer le mot voulu (la première traduction qui vient) et de l'insérer dans notre BDD avec une simple requête SQL.

Et voilà, grâce a cela, maintenant notre page nous permet, en insérant un mot français dans celle-ci, de le traduire et de l'insérer dans la BDD pour le quiz.

Mot en français :

☐  Éditer  Copier  Supprimer 24 Eggplant Aubergine

5 Gestion de la maintenance (corrective / évolutive)

5.1 Mise à jour de la documentation du SI

Toute documentation nécessaire a l'utilisation de l'API google traduction peut être trouvé à ce [lien](#) menant à la documentation officielle et mise a jour de cloud translate.

5.2 Procédure de correction d'un dysfonctionnement

En cas de dysfonctionnement, pas de réelle procédure a suivre. La partie la plus importante restera la lecture et la compréhension des messages d'erreurs.

5.3 Gestion des tests de mise à jour

Pour tester et être sûr en cas de mise a jour, il faut toujours garder une version stable sur un Git afin de toujours pouvoir y retourner en cas de problème.

6 Bilan du projet

6.1 Validation des exigences point par point

Faire un quiz anglais français avec plusieurs réponses possible.

Avoir la capacité d'insérer des mots supplémentaires petit à petit.

Apprendre à utiliser une API.

6.2 Axes d'amélioration

Potentiellement mise en place de sessions de quiz pour un utilisateur précis avec des mots précis.

6.3 Compétences acquises

L'utilisation et la configuration d'une API.

L'utilisation de jQuery pour envoyer des formulaires et traiter des données.