Documentation :
Site E-Calm

GIROUD Océane, LIU Jingyu, MARTIN Célia
Master 2, Sciences du Langage, parcours Industrie de la Langue

TABLE DES MATIERES

I. Organisation générale des fichiers

L'organisation des fichiers du site E-Calm suit l'architecture **MVC** (Model View Controller). Il n'y a donc pas de surprise, les dossiers sont classés de la manière suivante :

- Dossier **model** : traitements en php des informations ;
- Dossier **view** : affichage des informations pour l'utilisateur en html ;
- Dossier **controller** : récupère les informations données par le modèle et les envoie à la vue correspondante ;
- Fichier **index.php** : routeur, appelle le contrôleur adéquat en fonction des actions de l'utilisateur ;
- Dossier **public** : contient le reste des fichiers utiles à savoir :
    - Dossier **js** : traitements javascript (jquery/ajax);
    - Dossier **css** : styles des pages html ;
    - Dossier **assets** : rassemble les images (dossier **img**), les polices de caractères utilisées (dossier **font**), et dans notre cas particulier, les scans des copies des élèves (dossier **scan**) que l'utilisateur peut consulter depuis le site.

II. index.php : le routeur

Le routeur, comme précisé précédemment, est le chef d'orchestre. Il organise les actions effectuées par l'utilisateur, et appelle les contrôleurs correspondant à ces actions qui se chargeront de réaliser les traitements adéquats et d'envoyer les résultats aux vues. Les actions possibles de l'utilisateur gérées par le routeur sont les suivantes:

- Action **showResults** : appelle le contrôleur permettant de sélectionner dans la base de données les lignes qui correspondent aux critères de la partie "Interrogation des données" (en fonction des critères spécifiques aux verbes, aux adjectifs ou aux critères généraux) ;
- Action **showStats** : appelle le contrôleur permettant de calculer le tableau de statistiques sélectionné dans la partie "Statistiques descriptives" ;
- Action **deleteData** : appelle le contrôleur qui permet de supprimer des données dans la base de données (action possible seulement en tant que gestionnaire, il faut donc un compte sur le site) ;
- Action **addData** : appelle le contrôleur qui permet d'ajouter des données dans la base de données (action possible seulement en tant que gestionnaire) ;

- Action **connectionPage** : appelle le contrôleur qui redirige l'utilisateur vers la page de connexion ;
- Action **connection** : appelle le contrôleur qui vérifie l'identité de l'utilisateur et le connecte s'il est présent dans la base de données ;
- Action **disconnection** : appelle le contrôleur qui détruit la session en cours (déconnecte l'utilisateur) ;
- Pour toutes les **autres actions** : appelle le contrôleur qui redirige l'utilisateur vers la page d'accueil.
- Si une **page n'est pas trouvée** : renvoie un "Erreur 404".

III. Dossier model : traitement des informations

Ce dossier contient tous les scripts de traitement côté serveur. C'est dans ce dossier que sont rangés les scripts qui permettent d'interroger la base de données. Chaque fichier .php contient une classe.

1. *Classe **DataBase***

Cette classe réunit les fonctions permettant la connexion à la base de données et son interrogation.

a. Fonction **connection()**

Cette fonction permet la connexion à la base de données.

- *Retourne* : PDO.
- Jette une Exception si la connexion n'est pas possible.

b. Fonction **getData($request)**

Cette fonction se connecte à une base de données, exécute la requête SQL passée en paramètre et retourne un tableau des éléments ainsi sélectionnés.

- *Paramètre* string $request : une requête SQL.
- *Retourne* un array avec les lignes extraites de la base de données.
- Jette une Exception si les données n'ont pas été récupérées.

c. Fonction **addOrDelData($request)**

Cette fonction se connecte à la base de données, exécute la requête passée en paramètre et retourne le nombre de lignes affectées par la requête. Elle est utilisée principalement dans les scripts permettant la suppression et l'ajout de données.

- *Paramètre* string $request : une requête SQL.
- *Retourne* un entier qui est le nombre de lignes affectées par la requête.
- Jette une Exception si les données n'ont pas été récupérées.

2. *Classe **User***

Cette classe contient toutes les fonctions relatives à l'utilisateur.

Require : classe DataBase.

Propriétés :

- protégée $login : Identifiant de l'utilisateur.
- protégée $psw : Mot de passe de l'utilisateur.

a. Fonction **verifyUser()**

Cette fonction vérifie que l'identifiant entré par l'utilisateur est présent dans la base. S'il est présent, elle vérifie que le mot de passe est correct. Elle retourne *true* si la personne est effectivement présente dans la base de données, elle envoie un message d'erreur si ce n'est pas le cas.

- *Retourne* un booléen ou une phrase précisant si la partie fausse est le mot de passe ou l'identifiant.

b. Fonction **destroySession()**

Cette fonction détruit la session en cours pour permettre la déconnexion de l'utilisateur.

- *Ne retourne rien.*

3. *Classe **Export***

Cette classe contient la fonction nécessaire à la création de l'exemplier.

Require : classe DataBase.

Propriétés :

- protégée $word : lemme sélectionné par l'utilisateur.
- protégée $nbLine : nombre de lignes sélectionnées par l'utilisateur.

a. Fonction **sentences()**

Cette fonction permet de sélectionner un nombre de phrases d'exemples choisies par l'utilisateur contenant le lemme sélectionné par ce dernier.

- *Retourne* string $finalSentencesList : La liste des phrases contenant le mot choisi par l'utilisateur. Le nombre de lignes est sélectionné par l'utilisateur.

4. *Dossier AddDelData*

a. Classe **DeleteData**

Cette classe permet la suppression des données présentes sur la base de données à partir de paramètres choisis par l'utilisateur.

Require : classe DataBase.

Propriétés :

- protégée $corpus : Le nom d'un corpus choisi par l'utilisateur parmi les trois corpus du projet E-calm (Scoledit, Ecriscol ou Resolco).
- protégée $level : Le niveau pour lequel appliquer la suppression (du CP au M2).

i. Fonction **deleteData()**

Cette fonction permet de sélectionner dans la base de données toutes les lignes qui correspondent au corpus et au niveau sélectionnés par l'utilisateur et de les supprimer.

- *Retourne* int $nbLineAffected : le nombre de lignes supprimées lors de l'application de la fonction.

ii. Fonction **normalizeCorpusName()**

Cette fonction permet de passer du mot désignant le corpus à supprimer (Scoledit, Ecriscol ou Resolco) à une expression régulière permettant d'identifier ce corpus dans l'identifiant IdTok présent dans la base de données (Id unique décrivant les mots présents dans la base de données). En effet, IdTok contient la première lettre du corpus (S, E ou R), c'est cette lettre que les expressions régulières permettent de trouver pour sélectionner le corpus adéquat dans la base de données.

- *Ne retourne rien*.

b. Classe **InsertData**

Cette classe permet d'ajouter à la base de données les lignes présentes dans un fichier csv fourni par l'utilisateur.

Require : classe DataBase.

i. Fonction **inputCSV($handle)**

Cette fonction permet de charger les données présentes dans le fichier csv fourni par l'utilisateur.

- *Paramètre* $handle.
- *Retourne* un array : Tableau des lignes du fichier csv.

ii. Fonction **addCSV($filePath)**

La fonction addCSV() permet d'écrire dans la base de données les lignes du fichier csv donné par l'utilisateur. Elle génère une erreur si le fichier est vide.

- *Paramètre* string $filePath : le chemin vers le fichier csv (chemin temporaire sur le serveur).
- *Retourne* int $nbLineAffected : le nombre de lignes affectées par l'ajout.
- Jette une Exception.

5. *Dossier **SearchByCriteria***

a. Classe **Criterion**

Cette classe permet de récupérer des données dans la base de données en fonction des critères sélectionnés par l'utilisateur. Cette classe ne fonctionne pas pour les catégories de verbes et adjectifs, elle en est par contre la classe mère (se référer aux classes filles correspondantes : CriterionAdjective et CriteronVerb).

Require : classe DataBase.

Propriétés :

- protégée $corpus : Le corpus sélectionné parmi Scoledit, Ecriscol ou Resolco.
- protégée $level : Le niveau sélectionné entre le CP et le M2.
- protégée $pos : La catégorie grammaticale (Part Of Speech) sélectionnée.
- protégée $errStatus : Le statut d'erreur sélectionné (Normé, Phono, ApproxGraphique, ApproxPhono, ApproxArchiPhono, Non normé, Non pertinent, Tous).
- protégée $segmStatus : Le statut de segmentation (Normé, HyperSeg, HypoSeg, HypoHyperSeg, Non pertinent, Omis, Inséré, Tous).
- protégée $lemma : Le lemme sélectionné.

i. Fonction **getResults()**

Cette fonction permet de retourner un tableau en fonction des critères généraux sélectionnés par l'utilisateur (corpus, niveau, lemme, catégorie grammaticale, statut d'erreur et de segmentation). Elle donne un tableau contenant les lignes retournées par la requête SQL faite à la base de données (un mot par ligne avec ses informations). Elle ajoute également à chaque fin de ligne (donc pour chaque mot sélectionné) le scan associé.

- *Retourne* un array.

ii. Fonction **normalizeCriterions()**

Les données provenant de la page HTML sont dans un format agréable à lire pour l'utilisateur (pas d'abréviation pour les Part of Speech par exemple), cette fonction permet de

transcrire ces données pour qu'elles correspondent à la manière dont elles sont référencées dans la base de données (les Part of Speech sont de la forme de ceux de treetagger donc les adverbes seront des ADV par exemple). Cela permettra de créer la requête SQL.

- *Ne retourne rien.*

iii. Fonction **addScanLink($result)**

A chaque mot présent dans la base de données correspond ou non un scan de la copie de l'élève. Cette fonction permet de vérifier que le scan correspondant existe : si c'est le cas on crée un lien (balise <a href> en html) vers ce scan, sinon on précise à l'utilisateur que le scan est indisponible. Elle retourne un tableau contenant un mot par ligne avec ses informations telles qu'elles sont dans la base de données avec le lien vers le scan à la fin de chaque ligne.

- *Paramètre* array $result : tableau contenant les lignes correspondant à la recherche de l'utilisateur (résultat de la fonction getResults).
- *Retourne* un array contenant le même tableau qu'en entrée auquel on a ajouté le scan à la fin de chaque ligne.

b. Classe **CrirerionAdjectives** (fille de Criterion)

Cette classe permet de récupérer des données dans la base de données en fonction des critères sélectionnés par l'utilisateur. Cette classe ne fonctionne que pour les adjectifs (pour les verbes et les autres catégories se référer aux classes correspondantes : Criterion et CriteronVerb).

Require : classe DataBase.

Propriétés : Cette classe partage ses propriétés avec sa classe mère Criterion. Aux propriétés de cette dernière s'ajoutent les suivantes :

- privée $genre : Genre de l'adjectif sélectionné.
- privée $numbre : Nombre de l'adjectif sélectionné.
- privée $errGenre : Présence ou non d'erreurs sur le genre (0, 1 ou Tous).
- privée $errNumber : Présence ou non d'erreurs sur le nombre (0, 1 ou Tous).
- privée $base : Base sélectionnée.

i. Fonction **getResultsAdjective()**

Cette fonction permet de créer un tableau contenant les lignes retournées par la requête SQL envoyée à la base de données en fonction des critères sélectionnés par l'utilisateur. La requête est spécifique aux adjectifs (elle contient les champs de genre, nombre, erreur de genre, erreur de nombre et base en plus des champs des critères généraux).

- *Retourne* un array contenant le résultat de la requête.

ii. Fonction **normalizeCriterions()**

Cette fonction est la même que celle présente dans la classe mère Criterion, mais elle normalise en plus les informations spécifiques aux adjectifs (genre, nombre, erreur de genre, de nombre et base).

- *Ne retourne rien.*

c. Classe **CriterionVerbs** (fille de Criterion)

Cette classe permet de récupérer des données dans la base de données en fonction des critères sélectionnés par l'utilisateur. Cette classe ne fonctionne que pour les verbes (pour les adjectifs et les autres catégories se référer aux classes correspondantes : Criterion et CriteronAdjective).

Require : classe DataBase.

Propriétés : Cette classe partage ses propriétés avec sa classe mère Criterion. Aux propriétés de cette dernière s'ajoutent les suivantes :

- privée $tense : Tiroir verbal sélectionné (Conditionnel, Futur, Impératif, Imparfait, Infinitif, Participe présent, Présent, Passé simple, Subjonctif imparfait, Subjonctif présent).
- privée $person : Personne sélectionnée (1S, 2S, 3S, 1P, 2P, 3P).
- privée $typeErr : Présence d'une erreur soit sur la base, soit sur la désinence soit sur les deux (Erreur Base, Erreur Désinence, Erreur Base et Désinence).
- privée $base : Base sélectionnée.
- privée $ending : Désinence sélectionnée.

i. Fonction **getResultsVerbs()**

Cette fonction permet de créer un tableau contenant les lignes retournées par la requête SQL envoyée à la base de données en fonction des critères sélectionnés par l'utilisateur. La requête est spécifique aux verbes (elle contient les champs de tiroir verbal, personne, type d'erreur, désinence et base en plus des champs des critères généraux).

- *Retourne* un array des lignes de la requête.

ii. Fonction **normalizeCriterion()**

Cette fonction est la même que celle présente dans la classe mère, mais elle normalise en plus les informations spécifiques aux verbes (genre, nombre, erreur de genre, de nombre et base).

- *Ne retourne rien.*

6. Dossier Statistics

a. Classe **FailureAndSuccessTenses**

Cette classe ne contient qu'une seule fonction. Elle permet de créer le tableau : Répartition des échecs et réussites pour les tiroirs verbaux les plus employés.

Require : classe DataBase.

i. Fonction **createTabFailureSuccess ($verbGroup)**

Cette fonction retourne un tableau indiquant les pourcentages de formes correctes et de formes incorrectes qui permettent de restituer la forme sonore, et de formes incorrectes ne permettant pas de restituer la forme sonore. Elle sélectionne dans la base de données les verbes et les classe en fonction de ces réussites et échecs (comparaison avec le segment normé, le segment transcrit, la phonologie normée et la phonologie transcrite.

Elle permet à l'utilisateur de choisir d'afficher ces informations pour les verbes en -er, non en -er ou pour tous les verbes.

- *Paramètre* $verbGroup : "er" | "nonEr" | Tous_les_verbes

- *Retourne* un array.

b. Classe **NbVerbalForms**

Cette classe ne contient qu'une seule fonction. Elle permet de créer le tableau : Nombre de formes verbales analysées.

Require : DataBase.

i. Fonction **createTabNbVerbalForms()**

Cette fonction sélectionne dans la base de données les verbes correspondant aux informations suivantes pour chaque niveau : Nombre total de formes verbales (hors participes passés), Nombre de formes verbales analysées (c'est-à-dire les formes qui ont été analysées correctement par Treetagger, donc les verbes dont la base n'est pas égale à "#" ou à "_"), elle calcule ensuite le pourcentage de formes analysées.

- *Retourne* un array.

c. Classe **NbWordProd**

Cette classe n'a qu'une seule fonction qui permet de générer le tableau : Nombre de mots des productions.

Require : classe DataBase.

i. Fonction **createTabNbWordsProd()**

Cette fonction sélectionne les mots dans la base de données et les compte (hors ponctuations et balises) : elle crée un tableau final avec le nombre de mots par niveau, la longueur moyenne des productions par niveau et la taille minimum et maximum de ces productions par niveau.

- *Retourne* un array.

d. Classe **POSRepartitionByLevel**

Cette classe n'a qu'une seule fonction et permet de générer le tableau : Répartition des POS en fonction du niveau.

Require : classe DataBase.

i. Fonction **createTabPOSRepartitionByLevel()**

Cette fonction sélectionne les mots qui ont pu être analysés par Treetagger dans la base de données et calcule la répartition des catégories grammaticales en fonction des niveaux. Les résultats sont exprimés en pourcentage et enregistrés dans un tableau.

- *Retourne* un array.

e. Classe **StandardizedBaseEndingProportion**

Cette classe n'a qu'une seule fonction et permet de créer le tableau des proportions de bases et désinences normées et non normées.

Require : classe DataBase.

i. Fonction **createTabStandardizedBaseEndingProportion($verbGroup, $tense)**

Cette fonction prend en entrée les choix de l'utilisateur concernant le groupe du verbe et son temps (choix entre les verbes en -er ou non et entre les quatre temps [infinitif, présent, imparfait, passé simple]) et retourne un tableau contenant les informations : nombre de base / désinences erronées et nombre de base / désinence normées, en fonction des informations contenues dans la base de données.

- *Paramètre* string $verbGroup : Choix du groupe du verbe (-er | non -er | tous_les_verbes).
- *Paramètre* string $tense : Choix du temps (Infinitif | Présent | Imparfait | Passé Simple | tous_les_temps).
- *Retourne* un array.

f. Classe **StandardizedBaseOrEnding**

Cette classe n'a qu'une seule fonction et permet de créer le tableau des répartitions des formes verbales selon si leur base et/ou leur désinence sont normées.

Require : classe DataBase.

i. Fonction **createTabStandardizedBaseOrEnding($verbGroup, $tense)**

Cette fonction crée un tableau qui donne le pourcentage pour chaque niveau de forme verbale normée, de formes ayant une erreur sur la base, sur la désinence et sur les deux, en fonction des informations contenues dans la base de données.

L'utilisateur peut choisir d'afficher le tableau en fonction d'un temps (Infinitif, Imparfait, Présent, Passé Simple) et d'un type de verbe (en -er, non en -er ou tous).

- *Paramètre* string $verbGroup : Choix du groupe du verbe (-er | non -er | tous_les_verbes).
- *Paramètre* string $tense : Choix du temps (Infinitif | Présent | Imparfait | Passé Simple | tous_les_temps).
- *Retourne* un array.

g. Classe **TenseRepartition**

Cette classe n'a qu'une seule fonction et permet de générer le tableau : Répartition des tiroirs verbaux.

Require : classe DataBase.

i. Fonction **createTabTenseRepartition()**

Cette fonction sélectionne les verbes dans la base de données retourne un tableau montrant la répartition des tiroirs verbaux en fonction des niveaux en pourcentage.

- *Retourne* un array.

h. Classe **VerbalFormsRepartionBaseAndEndingPhono**

Cette classe ne possède qu'une seule fonction et permet de générer le tableau : Répartition des formes verbales non normées selon si leur base et/ou leur désinence respectent ou non la phonologie.

Require : classe DataBase.

i. Fonction **createTabVerbalFormsRepartitionBaseAndEndingPhono($verbGroup, $tense)**

Cette fonction sélectionne dans la base de données les informations relatives à la phonologie et crée un tableau montrant le pourcentage de réussites et d'échecs des bases et désinences au niveau phonologique.

- *Paramètre* string $verbGroup Choix du groupe du verbe (-er | non -er | tous_les_verbes).
- *Paramètre* string $tense Choix du temps (Infinitif | Présent | Imparfait | Passé Simple | tous_les_temps).
- *Retourne* un array.

IV. Dossier view : affichage des informations

Ce dossier est consacré à l'affichage html. Il sert donc la partie client.

*1. Fichier **home.php** : page d'accueil*

Ce fichier contient la structure html de la page d'accueil qui est la page centrale du site où la très grande majorité des actions sont effectuées. Il est donc divisé en plusieurs parties, correspondant aux différentes actions possibles. On y trouve donc, en dehors de l'entête où se trouve notamment le bouton de connexion :

- Les boutons de **navigation** : Données, Statistiques descriptives et Ajouter et supprimer des données (ce dernier bouton est disponible seulement quand un des gestionnaires est connecté) ;

- La partie permettant la **sélection des critères** pour l'affichage des informations prenant la forme d'un formulaire dont les informations sont envoyées au serveur via Ajax qui se compose elle-même de trois sous-parties :
    - Sous-partie permettant la sélection des critères généraux, commune à **toutes les catégories grammaticales** ;
    - Sous-partie permettant la sélection de critères plus avancés réservés aux **verbes** ;
    - Sous-partie permettant la sélection de critères plus avancés réservés aux **adjectifs** ;
- La partie permettant la sélection du **tableau de statistiques** à afficher avec des critères spécifiques à certains tableaux qui sont affichés au moment de leur sélection (groupe verbal / tiroir verbal), également sous la forme d'un formulaire envoyé au serveur via Ajax ;
- La partie comprenant le bouton permettant le **téléchargement** des résultats de la requête ou des statistiques, ainsi que le bouton permettant le téléchargement d'un **exemplier** contenant des exemples de l'utilisation du lemme. Les deux téléchargements se font avec l'envoi d'un formulaire, les fichiers téléchargés sont sous la forme de tableaux enregistrés au format tsv ;
- La dernière partie correspond à l'affichage des résultats des différentes opérations.

2. *Fichier **userConnection.php** : page de connexion*

Outre le même en-tête que la page d'accueil, cette page permet exclusivement aux utilisateurs de se connecter à l'aide de leur identifiant et de leur mot de passe. Elle ne se compose donc que d'un formulaire envoyé via Ajax au serveur. S'il y a une erreur, elle est retournée dans la div ayant pour id "alerts", sinon l'utilisateur est connecté et redirigé vers la page d'accueil.

3. *Fichier **error.php** : affichage des erreurs*

Cette page permet seulement d'uniformiser les messages d'erreur avec le reste du site et permet à l'utilisateur de cliquer sur un bouton pour être redirigé vers la page d'accueil sans avoir à faire retour ou à recharger le site.

V. Dossier controller : lien entre les informations et la vue

1. *Classe **HomeController***

Cette classe contient tous les contrôleurs utilisés dans la page d'accueil (hors gestionnaire).

Require :

- Classe Criterion.php ;
- Classe CriterionVerb.php ;
- Classe CriterionAdjective.php ;
- Classe FailureAndSuccessTenses.php ;
- Classe NbVerbalForms.php ;
- Classe NbWordProd.php ;
- Classe POSRepartitionByLevel.php ;
- Classe VerbalFormsRepartitionBaseAndEndingPhono.php ;
- Classe StandardizedBaseEndingProportion.php ;
- Classe StandardizedBaseOrEnding.php ;
- Classe TenseRepartition.php.

a. Fonction **home()**

Ce contrôleur permet de rediriger l'utilisateur vers la page d'accueil.

Ne retourne rien.

b. Fonction **showResultsCriteria($corpus, $level, $pos, $errStatus, $segmStatus, $lemma)**

Contrôleur pour les critères généraux : Permet de retourner les résultats de la requête (interrogation simple, exclut les verbes et les adjectifs) faite par l'utilisateur au format JSON pour un traitement avec Ajax.

- *Paramètre* $corpus : Le corpus sélectionné parmi Scoledit, Ecriscol ou Resolco.
- *Paramètre* $level : Le niveau sélectionné entre le CP et le M2.
- *Paramètre* $pos : La catégorie grammaticale (Part Of Speech) sélectionnée.

- *Paramètre* $errStatus : Le statut d'erreur sélectionné (Normé, Phono, ApproxGraphique, ApproxPhono, ApproxArchiPhono, Non normé, Non pertinent, Tous).
- *Paramètre* $segmStatus : Le statut de segmentation (Normé, HyperSeg, HypoSeg, HypoHyperSeg, Non pertinent, Omis, Inséré, Tous).
- *Paramètre* $lemma : Le lemme sélectionné.
- *Ne retourne rien.*

c. Fonction **showResultsCriteriaVerbs($corpus, $level, $pos, $errStatus, $segmStatus, $lemma, $tense, $person, $typeErr, $base, $ending)**

Contrôleur pour les critères des verbes : Permet de retourner les résultats de la requête faite par l'utilisateur pour les verbes au format JSON pour un traitement avec Ajax.

- *Paramètre* $corpus : Le corpus sélectionné parmi Scoledit, Ecriscol ou Resolco.
- *Paramètre* $level : Le niveau sélectionné entre le CP et le M2.
- *Paramètre* $pos : La catégorie grammaticale (Part Of Speech) sélectionnée.
- *Paramètre* $errStatus : Le statut d'erreur sélectionné (Normé, Phono, ApproxGraphique, ApproxPhono, ApproxArchiPhono, Non normé, Non pertinent, Tous).
- *Paramètre* $segmStatus : Le statut de segmentation (Normé, HyperSeg, HypoSeg, HypoHyperSeg, Non pertinent, Omis, Inséré, Tous).
- *Paramètre* $lemma : Le lemme sélectionné.
- *Paramètre* $tense : Tiroir verbal sélectionné (Conditionnel, Futur, Impératif, Imparfait, Infinitif, Participe présent, Présent, Passé simple, Subjonctif imparfait, Subjonctif présent).
- *Paramètre* $person : Personne sélectionnée (1S, 2S, 3S, 1P, 2P, 3P).
- *Paramètre* $typeErr : Présence d'une erreur soit sur la base, soit sur la désinence soit sur les deux (Erreur Base, Erreur Désinence, Erreur Base et Désinence).
- *Paramètre* $base : Base sélectionnée.
- *Paramètre* $ending : Désinence sélectionnée.
- *Ne retourne rien.*

d. Fonction **showResultsCriteriaAdjectives($corpus, $level, $pos, $errStatus, $segmStatus, $lemma, $genre, $numbre, $errGenre, $errNumber, $base)**

Contrôleur pour les critères des adjectifs : Permet de retourner les résultats de la requête faite par l'utilisateur pour les adjectifs au format JSON pour un traitement avec Ajax.

- *Paramètre* $corpus : Le corpus sélectionné parmi Scoledit, Ecriscol ou Resolco.
- *Paramètre* $level : Le niveau sélectionné entre le CP et le M2.
- *Paramètre* $pos : La catégorie grammaticale (Part Of Speech) sélectionnée.
- *Paramètre* $errStatus : Le statut d'erreur sélectionné (Normé, Phono, ApproxGraphique, ApproxPhono, ApproxArchiPhono, Non normé, Non pertinent, Tous).
- *Paramètre* $segmStatus : Le statut de segmentation (Normé, HyperSeg, HypoSeg, HypoHyperSeg, Non pertinent, Omis, Inséré, Tous).
- *Paramètre* $lemma : Le lemme sélectionné.
- *Paramètre* $genre : Genre de l'adjectif sélectionné.
- *Paramètre* $numbre : Nombre de l'adjectif sélectionné.
- *Paramètre* $errGenre : Présence ou non d'erreurs sur le genre (0, 1 ou Tous).
- *Paramètre* $errNumber : Présence ou non d'erreurs sur le nombre (0, 1 ou Tous).
- *Paramètre* $base : Base sélectionnée.
- *Ne retourne rien.*

e. Fonction **showStatsTab($tabName, $verbGroup, $tense)**

Contrôleur pour les différents tableaux de statistiques. Il retourne le tableau sélectionné par l'utilisateur au format JSON pour un traitement avec Ajax.

- *Paramètre* $tabName : Nom du tableau à générer.
- *Paramètre* $verbGroup : Groupe verbal pour lequel afficher le tableau.
- *Paramètre* $tense : Tiroir verbal pour lequel afficher le tableau.
- *Ne retourne rien.*
- Jette une *Exception* si le tableau ne peut être créé.

2. *Classe **ManagerController***

Cette classe contient les contrôleurs pour la partie gestionnaire (ajout et suppression des données).

Require :

- Classe DeleteData.php
- Classe InsertData.php

a. Fonction **delete($corpus, $level)**

Contrôleur pour la suppression des données. Supprime les données correspondant au niveau et au corpus entrés par l'utilisateur et redirige l'utilisateur vers la page d'accueil en générant une alerte montrant le nombre de lignes affectées par l'opération.

- *Paramètre* $corpus : Corpus à supprimer.
- *Paramètre* $level : Niveau à supprimer.
- *Ne retourne rien.*

b. Fonction **add($file)**

Contrôleur pour l'ajout de données à partir d'un fichier csv. Il prend en entrée un fichier csv, ajoute ses lignes à la base de données et redirige l'utilisateur vers la page d'accueil en générant une alerte montrant le nombre de lignes affectées par l'opération.

- *Paramètre* $file : Fichier csv à ajouter.
- *Ne retourne rien.*

3. *Classe **UserController***

a. Fonction **connectionPage()**

Ce contrôleur dirige l'utilisateur vers la page de connexion.

- *Ne retourne rien.*

b. Fonction **login($login, $mdp)**

Ce contrôleur permet de vérifier l'identité de la personne qui se connecte et renvoie la réponse (true si la connexion a eu lieu, sinon un message d'erreur à afficher dans la vue) au format JSON pour un traitement avec Ajax.

- *Paramètre* $login : Identifiant de l'utilisateur.
- *Paramètre* $mdp : Mot de passe de l'utilisateur.
- *Ne retourne rien.*

c. Fonction **logout()**

Ce contrôleur permet de supprimer la session en cours, donc de déconnecter l'utilisateur et de le rediriger vers la page d'accueil.

- *Ne retourne rien.*

*4. Classe **ExportController***

Cette classe contient le controller nécessaire à l'exportation de l'exemplier.

Require :

- Classe Export.php

a. Fonction **Export()**

Ce contrôleur permet l'exporation de l'exemplier. Il permet de retourner les phrases d'exemple à partir du mot choisi par l'utilisateur au format JSON pour un traitement avec ajax.

- *Paramètre* $word : mot sélectionné par l'utilisateur.
- *Paramètre* $nbLine : nombre de lignes à afficher sélectionné par l'utilisateur.
- *Ne retourne rien.*

VI. Dossier public

1. *Dossier **assets***

Ce dossier contient les polices utilisées dans le fichier css, les images affichées dans la page html et enfin un dossier des scans des copies d'élèves disponibles. Ce dernier dossier est utilisé dans la fonction addScanLink() pour vérifier l'existence du scan et ajouter un lien vers celui-ci dans les informations retournées par les fonctions getResults(), getResultsVerbs() et getResultsAdjectives() pour la sélection des données dans la base de données en fonction des critères d'interrogation sélectionnés par l'utilisateur.

Le dossier assets pourra contenir le reste des documents utiles au site.

2. *Dossier **css***

Ce dossier contient les styles utilisés dans les différentes pages html. Le fichier style.css en est le seul fichier pour le moment et se veut au maximum divisé par pages auxquelles s'appliquent les styles ou par élément (style des boutons au passage de la souris etc.).

3. *Dossier **js***

Le dossier js comporte tous les scripts javascript associés aux différentes pages html. Il ne contient actuellement qu'un seul fichier.

a. Fichier script.js

Ce fichier se compose de toutes les fonctions et évènements utilisés côté client. Les fonctions sont regroupées par rapport à leur utilité dans la page.

i. Navigation entre les volets

On retrouve tout d'abord les évènements correspondant à la **navigation entre les différentes fonctionnalités du site** : notamment entre les boutons "*Interrogation des*

données", "*Statistiques descriptives*" et "*Ajout et Suppression des données*" (pour le gestionnaire lorsqu'il est connecté). Ces fonctions permettent ainsi de montrer le contenu correspondant au volet sélectionné et de cacher les autres, de montrer les boutons utilisables sur ces pages (on peut télécharger les résultats des données et statistiques mais pas pour l'ajout et la suppression de données, on peut également télécharger un exemplier seulement pour les données etc.) et de donner une couleur au bouton qui est actuellement sélectionné pour que l'utilisateur sache dans quel volet il se trouve.

ii. Partie Interrogation des données

Les évènements suivants permettent de **gérer les actions qui se déroulent dans le volet "Interrogation des données"**. C'est dans cette partie que l'on peut **gérer l'affichage des critères avancés** pour les verbes et les adjectifs et les cacher pour les autres catégories, et que l'on trouve également la fonction qui permet d'**envoyer les critères sélectionnés au routeur via AJAX** et de **récupérer le tableau de données** récupéré dans la base de données. Cette dernière fonction permet notamment de **donner les couleurs aux parties vraies et fausses pour les adjectifs et les verbes** affichés, de **normaliser l'affichage des données** (par exemple ne pas écrire "ADV" qui est la catégorie de Treetagger enregistrée dans la base de données, mais plutôt "Adverbe") en utilisant la fonction **normalizeTense(posMessage)** pour la normalisation des tiroirs verbaux. Ainsi, le résultat de la requête AJAX permet de construire intégralement le tableau de résultats à afficher à l'utilisateur.

iii. Partie Statistiques descriptives

La partie qui suit permet de **gérer les actions qui se déroulent dans le volet "Statistiques descriptives"**. La première fonction permet d'**afficher les critères associés au tableau sélectionné** par l'utilisateur (par exemple elle affiche le checkbox permettant de choisir le groupe de verbe sur lequel appliquer le tableau lorsque le tableau "Répartition des échecs et réussites pour les tiroirs verbaux les plus employés" est sélectionné par l'utilisateur). La seconde fonction, tout comme pour la partie "Interrogation des données", permet d'**envoyer au routeur les informations relatives au tableau sélectionné par l'utilisateur via AJAX** pour permettre leur traitement en php. Elle envoie donc le tableau, le groupe verbal et le tiroir verbal sélectionnés au routeur, le résultat de la requête AJAX étant

alors le tableau de statistiques ainsi sélectionné et calculé sur les données présentes dans la base de données. Chaque tableau est alors traité pour l'affichage en html.

iv. Bouton téléchargement des résultats

Vient ensuite la fonction qui permet de **télécharger les données interrogées ou le tableau généré lorsque l'utilisateur clique sur le bouton "Télécharger le résultat"**. Elle **récupère le contenu de la balise html <table>**, **convertit ce contenu en string** pour qu'il soit utilisable par la fonction **strDownload(string, fileName)** qui permet le **téléchargement du contenu récupéré dans un fichier au format tsv**.

v. Bouton téléchargement d'un exemplier

Les évènements suivants concernent le **bouton "Télécharger un exemplier"**. La première fonction **affiche la fenêtre permettant la sélection du mot dont on veut faire un exemplier et du nombre de lignes** que doit contenir le fichier. Elle extrait les mots présents dans la balise html <table> et en fait un menu déroulant (sans les doublons). La fonction suivante récupère la valeur de ce menu déroulant et le nombre de lignes désirées, les envoie au routeur via AJAX, le résultat est un string contenant les lignes voulues séparées d'un saut de ligne. Ces lignes sont téléchargées sur l'ordinateur de l'utilisateur grâce à la fonction strDownlad(string, fileName). Cette partie est terminée par la fonction qui permet de **fermer la fenêtre à l'aide de la croix** présente dans le coin à gauche.

vi. Page de connexion

L'avant-dernier évènement permet à l'utilisateur de **se connecter** pour effectuer les actions d'ajout et suppression des données. Elle récupère l'identifiant et le mot de passe renseignés par l'utilisateur, les envoie au routeur via AJAX. Le résultat est soit le message d'erreur retourné par le script php qui vérifie l'identité de l'utilisateur soit la valeur "true". Si on a un message d'erreur il est affiché à l'utilisateur, sinon il est connecté et renvoyé à la page d'accueil.

vii. Affichage de la zone d'ajout et suppression des données

Le dernier évènement a lieu lorsque la personne **clique sur le bouton "Ajout et suppression des données" : elle permet de construire les formulaires nécessaires à ces deux actions**. On construit ces formulaires en utilisant javascript et non en les incluant au html directement et en les cachant pour ne pas que les utilisateurs puissent y avoir accès en affichant le code source avec f12.

viii. Fonction **strDownload(text, fileName)**

La fonction qui se trouve à la fin du fichier est la **fonction strDownload(text, fileName)** qui prend donc en entrée un string et un nom de fichier, et télécharge ce string sur l'ordinateur de l'utilisateur dans un fichier portant le nom donné en entrée. Elle ne retourne rien.