Université Grenoble Alpes

Projet Programmation Web et TAL
Master 1 IdL – semestre 2 - année 2024-2025

Cahier des charges

Marion LETEILLIER
Naja Sthael FERREIRA

Présentation du projet

Contexte

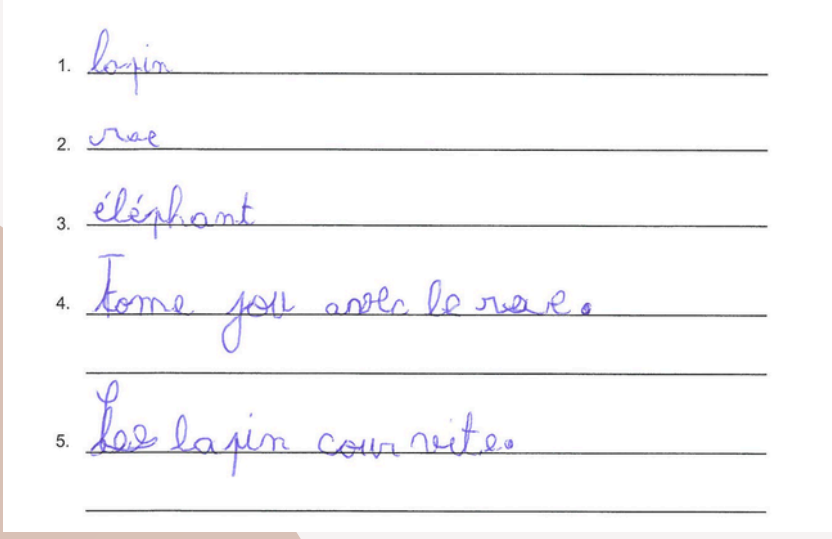
Lors de projets visant à améliorer les compétences orthographiques des élèves, des dictées (mots, phrases, textes) ont été recueillies sous format papier. Pour faciliter leur analyse, une application web sera développée pour automatiser la saisie, l'évaluation et l'interprétation des dictées.

Objectifs

L'objectif est de créer une application Web d'analyse orthographique permettant :

- La définition et l'enregistrement de dictées de mots
- La saisie et l'analyse automatique des dictées
- L'interrogation et l'exploitation des données sous plusieurs critères
- L'exportation des résultats en Excel

Exemple d'une dictée



Expression des besoins

Besoins fonctionnels

Gestion des dictées : Création, modification
Saisie des réponses des élèves : Manuelle
Analyse automatique : Comparaison avec les réponses correctes, détection des erreurs orthographiques
Interrogation des données : Recherche par type de dictée (mots), niveau des élèves (CE2, CM1), catégorie grammaticale (verbe, adjectif, nom..), date de passation (jour, mois, année), évolution temporelle (d'une année à l'autre)
Visualisation des résultats : Génération de tableaux pour représenter les erreurs et progrès
Export des résultats : Formats Excel
Intégration d'un dictionnaire pour vérifier l'orthographe

Technologies utilisées

Frontend : HTML, CSS, JavaScript, JQuery
Backend : PHP, Ajax
Base de données : MySQL
API intégrée : LanguageTool

Présentation du projet

Ce projet consiste en la création d'une application web dédiée à la gestion de dictées scolaires. Il s'adresse aux enseignants de niveau élémentaire et a pour objectif de faciliter la création et la passation de dictées ainsi que la correction automatisée et l'analyse des réponses. L'enseignant peut rédiger une dictée via une interface en saisissant un titre et un contenu. Celle-ci est enregistrée dans une base de données MySQL et devient ensuite accessible. L'enseignant choisi leur nom, leur niveau, une dictée et saisi les réponses des élèves. L'application se charge ensuite de comparer cette réponse avec le texte original afin de détecter et classer les erreurs.

Une première évaluation globale est réalisée à l'aide de l'algorithme de Levenshtein, qui mesure le nombre de modifications nécessaires pour transformer la réponse de l'élève en texte correct. Cette distance permet de calculer un pourcentage d'erreur global ainsi qu'une note estimée sur 10. Ce premier indicateur donne une vue d'ensemble de la qualité de la copie. Ensuite, une analyse détaillée mot par mot est réalisée. Le texte de la dictée et la réponse sont découpés en mots tout en tenant compte des apostrophes (par exemple, « l'élève » devient deux mots : « l' » et « élève »). Chaque mot attendu est comparé à ceux saisis par l'élève. Si une correspondance exacte n'est pas trouvée, l'algorithme tente d'identifier des fautes spécifiques. Cela repose sur un lexique présent dans la base de données, où chaque mot est associé à une catégorie grammaticale (nom, verbe, déterminant...), à un lemme (forme de base) et à une désinence si elle existe.

Une faute de désinence est détectée si le mot est presque correct mais contient une terminaison incorrecte. Une faute d'accent est repérée si le mot est identique sans les accents. Une erreur de segmentation est notée lorsqu'un mot comme « l'élève » est fusionné incorrectement en « lélève ». Il peut aussi détecter les erreurs d'omission (mot manquant), d'orthographe (si la similarité est forte mais incorrecte), ou encore les fautes de ponctuation comme l'oubli d'un point. Chaque erreur est ensuite catégorisée selon sa nature grammaticale (NOM, VERBE, etc.). Les résultats de l'analyse sont affichés puis enregistrés dans la base de données avec la date. L'enseignant peut ensuite consulter les performances via un module de statistiques : soit par dictée (taux d'erreurs par élève), soit par élève (progression individuelle), avec la possibilité d'exporter les données au format Excel.

Une intégration avec l'outil *LanguageTool* permet en complément d'analyser automatiquement les fautes grammaticales et orthographiques. Lorsque des erreurs de conjugaison sont détectées, un lien vers le site *Le Conjugueur* est proposé afin de consulter rapidement les conjugaisons correctes du verbe concerné.

Conception de la base de données

Exemple de schéma relationnel

Élèves	Dictées	Réponses	Erreurs
ID	⬅ID	ID	ID
Nom	Type	⬅Elève ID	Réponse ID
Prénom	Date	Dictée ID	⬅Type
Niveau		Texte	Catégorie
		Erreur	Correction

Un élève (Élèves) fait une dictée (Dictées) et sa réponse est stockée dans Réponses. Si des erreurs sont repérées, elles sont enregistrées dans Erreurs.

Déroulement du projet

Planification

28 février 2025 — Rédaction du mini-cahier des charges V1

30 mars 2025 — Rédaction du mini-cahier des charges V2: Spécification des besoins et ajout de HTML/CSS

avril 2025 — Conception de l'interface et de la base de données

avril 2025 — Développement des fonctionnalités et tests

avant le 9 mai 2025 — Rendu final avec le cahier des charges

Les différentes pages Php

INDEX.PHP

Cette page présente l'interface d'accueil de l'application de dictées. Elle est conçue pour accueillir les enseignants, en leur présentant de manière claire les trois principales fonctionnalités : créer une dictée, faire une dictée, et consulter les statistiques des élèves ou dictées.

Il y a un style CSS intégré dans la balise <*style*> avec une esthétique douce et pastel : tons violets, coins arrondis, ombrages légers et boutons interactifs.

La structure est organisée en trois grandes parties. En haut, un en-tête <*header*> affiche le titre de l'application dans une grande police avec un fond coloré. Juste en dessous, un encadré rose clair affiche un message de bienvenue destiné à l'utilisateur. Ensuite, la section centrale <*section*> contient trois blocs <*article*>, chacun dédié à une action spécifique. Le premier bloc permet de rédiger une nouvelle dictée en redirigeant vers la page *create_dictee.php*. Le second permet de commencer une passation de dictée, en allant vers *passation.php*. Enfin, le troisième bloc mène à la page *stats.php*, qui affiche les résultats et analyses des dictées passées.

Chaque bloc est présenté avec un titre, une courte description, et un bouton violet (avec effet de survol) qui permet d'accéder à la fonctionnalité concernée.

Les différentes pages Php

CONNEXION.PHP

Ce script PHP est conçu pour établir une connexion sécurisée à une base de données MySQL à l'aide de l'extension PDO (PHP Data Objects). Il utilise un bloc try-catch pour gérer les erreurs : si la connexion fonctionne, un message de confirmation est affiché, sinon une erreur détaillée est renvoyée.

À l'intérieur du bloc try, l'objet $pdo est instancié avec les paramètres nécessaires pour accéder à la base : l'adresse du serveur (localhost), le nom de la base (leteillierm), l'identifiant utilisateur (leteillierm) et le mot de passe.

CREATE_DICTEE.PHP

Cette page commence par définir la structure HTML, un encodage UTF-8 et une feuille de style CSS intégrée. Le contenu principal est un formulaire de création de dictée. Un en-tête <*header*> en haut de la page rappelle le nom de l'application. Ensuite, une section blanche au centre affiche le titre "Créer une nouvelle dictée" suivi du formulaire. Ce formulaire contient deux champs à remplir : le titre de la dictée (par exemple *"L'hiver"*), et le contenu de la dictée, qui attend une liste de mots séparés par des virgules (ex. : *neige; bonnet; une écharpe; le manteau; Il met son manteau chaud.*). Un bouton "Enregistrer" permet d'envoyer les données.

Lorsque l'utilisateur soumet le formulaire, les informations sont envoyées via la méthode POST à la page *save_dictee.php* qui sauvegarde la dictée dans la base de données. Les champs *required* obligent l'utilisateur à remplir les deux zones avant de pouvoir valider.

Les différentes pages Php

SAVE_DICTEE.PHP

Dès le début du fichier, la page inclut *connexion.php* qui établit une connexion sécurisée à la base de données. Elle récupère ensuite deux données envoyées en méthode POST depuis un formulaire : le titre et le contenu de la dictée. Ces deux éléments sont ensuite insérés dans la table *dictees* de la base via une requête SQL paramétrée (INSERT INTO).

Une fois l'insertion réussie, le reste du fichier est du HTML avec du CSS intégré. Il y a une page de confirmation indiquant à l'utilisateur que la dictée a bien été enregistrée. Un bloc central vert s'affiche pour confirmer l'enregistrement. En dessous, un bouton invite l'utilisateur à revenir au menu principal *index.php*.

PASSATION.PHP

Cette page commence par inclure le fichier *connexion.php*. Elle propose à l'enseignant un formulaire pour sélectionner un niveau d'élève, puis un élève, une dictée, une date de passation, et enfin pour saisir la réponse de l'élève. Ce formulaire est ensuite envoyé à la page *analyser_reponse.php* qui va comparer la réponse à la dictée et enregistrer les résultats.

Le formulaire placé en haut permet de filtrer les élèves par niveau (*CE2, CM1*). Il utilise la méthode GET et recharge la page à chaque changement de niveau pour afficher la liste correspondante d'élèves dans la sélection. Le formulaire principal comprend : une liste déroulante des élèves (filtrée si un niveau est sélectionné), une liste des dictées enregistrées dans la base, un champ pour choisir la date de la dictée et un champ texte pour entrer la réponse de l'élève.

L'envoi du formulaire se fait par la méthode POST pour protéger les données, et tous les champs sont requis *required* afin de s'assurer que l'ensemble des informations demandées est bien renseigné.

Les différentes pages Php

ANALYSER_REPONSE.PHP

La première ligne *connexion.php* connecte à la base de données via un fichier externe. Le code vérifie que toutes les données essentielles du formulaire sont bien présentes (*id_eleve, id_dictee, reponse*). Si une donnée est manquante, le script s'arrête immédiatement avec un message d'erreur. La fonction *Normalizer* est utilisée pour la gestion des accents. Si l'extension n'est pas activée sur le serveur, un message d'erreur s'affiche car certaines opérations Unicode seraient impossibles.

Une série de fonctions est définie pour faciliter l'analyse des erreurs : *tokenize()* : transforme un texte en une liste de mots. Elle gère la casse (met en minuscules), remplace les apostrophes typographiques par une apostrophe standard, sépare les contractions (l'élève devient l' et élève) puis découpe le texte avec une expression régulière. *get_radical()* extrait la racine du mot en supprimant sa désinence (pour identifier les erreurs de conjugaison ou de flexion). *is_segmentation_error()* détecte si une contraction a été mal transcrite (ex : l'élève → leleve). *remove_accents()* et *has_accent()* : servent à comparer les mots en ignorant les accents, pour repérer les fautes comme foret au lieu de forêt.

À partir de l'identifiant de dictée, le code va chercher dans la base le texte original de la dictée. Il applique la fonction *tokenize()* sur ce texte et sur la réponse de l'élève pour obtenir deux listes de mots à comparer.

La fonction *analyse_erreur_leven()* utilise l'algorithme de Levenshtein pour mesurer la différence globale entre le texte original et la réponse. Cela donne un pourcentage d'erreur global, à partir duquel une note sur 10 est calculée.

Le script charge ensuite tous les mots du lexique (avec leurs catégories, lemmes, désinences) depuis la table *lexique*. Chaque mot du texte original est comparé à chaque mot de la réponse élève. Si une correspondance parfaite est trouvée, le mot est validé. Sinon, le script tente d'identifier la meilleure correspondance et note les erreurs détectées : orthographe (erreur avec plus de 70% de ressemblance), accent (seule différence = absence d'accent), désinence, segmentation (mot contracté mal géré) ou enfin omission si aucun mot correspondant n'est trouvé.

Les différentes pages Php

Les erreurs sont classées par catégorie grammaticale, ce qui permet une analyse plus précise (ex : erreurs de noms, de verbes, etc.). Le script vérifie aussi la présence du point final (.). Le rapport est ensuite formaté en texte avec un décompte des erreurs par catégorie suivi de la liste des erreurs. Toutes les données sont enregistrées dans la table *reponses* : identifiants de la dictée et de l'élève, réponse saisie, rapport d'erreurs et date de passation.

STATS.PHP

En haut du fichier, le script inclut le fichier *connexion.php*. Une fonction *calculer_stats()* utilise l'algorithme de Levenshtein pour mesurer la différence entre la dictée originale et la réponse d'un élève. La distance est convertie en pourcentage d'erreurs, puis en une note sur 10.

Deux requêtes sont exécutées pour récupérer toutes les dictées (id, titre) et tous les élèves (id, nom, niveau). Ces listes sont utilisées pour remplir deux menus déroulants (un pour les dictées, un pour les élèves).

Deux formulaires indépendants sont proposés : l'un pour afficher les résultats par dictée, l'autre par élève. Lorsqu'une sélection est faite, le formulaire est automatiquement soumis *onchange="this.form.submit()"* pour recharger la page avec les données correspondantes.

Si une dictée est sélectionnée *id_dictee*, le script récupère les réponses de tous les élèves pour cette dictée. Pour chaque réponse, il recalcule le pourcentage d'erreurs et la note à l'aide de *calculer_stats()* et les affiche dans un tableau : nom de l'élève, erreurs détectées, date, note, pourcentage d'erreurs. Un bouton permet aussi d'exporter les résultats de cette dictée en Excel via *export_par_dictee.php*.

De la même manière, si un élève est sélectionné *id_eleve*, toutes ses réponses à différentes dictées sont affichées. Le titre de chaque dictée, la réponse de l'élève, les erreurs, la date et les notes calculées sont montrés. L'utilisateur peut également exporter ces résultats au format Excel via *export_par_eleve.php*. Un bouton "*Retour au menu*" est proposé à la fin de la page pour revenir à l'accueil de l'application *index.php*.

Les différentes pages Php

EXPORT_PAR_ELEVE.PHP

Le script commence par inclure le fichier *connexion.php* pour se connecter à la base de données. Ensuite, une fonction *calculer_stats()* est définie : elle utilise la distance de Levenshtein pour évaluer la similarité entre le texte original de la dictée et la réponse de l'élève. Le pourcentage d'erreur est calculé puis une note sur 10 est déduite de cette erreur.

L'élève est identifié via l'URL *$_GET['id_eleve']*. Si cet ID est absent, on redirige vers la page des statistiques. Une fois l'ID validé, une requête récupère les informations de l'élève : son nom et sa classe (niveau) stockées dans les variables $prenom et $classe pour nommer le fichier exporté.

Le script prépare une requête SQL pour récupérer toutes les réponses de cet élève avec les titres de dictées et à la date de saisie. Cela permet d'avoir un historique trié par date. Les en-têtes HTTP spécifient que le contenu doit être interprété comme un fichier Excel *Content-Type: application/vnd.ms-excel* et forcent le téléchargement *Content-Disposition: attachment*. Le BOM UTF-8 est ajouté avec *\xEF\xBB\xBF* pour garantir que les caractères accentués soient bien affichés dans Excel.

Un tableau HTML est généré. Pour chaque ligne de réponse, le contenu original de la dictée est récupéré. La fonction *calculer_stats()* est utilisée pour calculer le % d'erreur et la note. Les données sont insérées dans une ligne du tableau : titre de la dictée, réponse, erreurs détaillées (<pre> pour conserver les retours à la ligne), date, note et pourcentage.

Une fois toutes les lignes générées, le tableau est fermé. Le fichier est transmis au navigateur qui va le proposer au téléchargement.

Les différentes pages Php

EXPORT_PAR_DICTEE.PHP

Tout comme le fichier *export_par_eleve.php,* ce script a pour but de générer un fichier Excel téléchargeable mais qui ici présente les résultats des élèves à une dictée précise.

Le script vérifie ensuite que l'identifiant d'une dictée a bien été fourni dans l'URL. Si ce n'est pas le cas, il redirige l'utilisateur vers la page de statistiques. Une fois l'ID validé, il prépare une requête SQL pour récupérer toutes les réponses liées à cette dictée, incluant le nom de l'élève, le titre et contenu de la dictée, la réponse soumise, les erreurs détectées et la date de saisie.

Une requête séparée permet de récupérer le titre exact de la dictée pour le nom du fichier exporté. Ce titre est nettoyé pour être compatible avec les fichiers (remplacement des espaces par des underscores).

CORRECTION_EXTERNE.PHP

Elle propose aux enseignants de soumettre les réponses des élèves aux dictées afin d'en vérifier la correction grammaticale et orthographique. Un formulaire propose un champ <*textarea*> où l'utilisateur peut saisir un texte à corriger.

Lorsque l'enseignant soumet un texte via le formulaire, le script PHP envoie ce texte à l'API de LanguageTool, un outil en ligne de correction grammaticale. La requête est faite par cURL en mode POST avec les paramètres text (le texte entré) et language défini sur 'fr' pour le français. Une fois la réponse reçue en JSON, le script l'analyse si des erreurs sont détectées : une section intitulée "Suggestions" apparaît alors. Chaque mot ou groupe de mots fautif est affiché en gras suivi des suggestions de correction en italique. Si l'erreur est de type grammatical (un verbe mal conjugué), un lien est proposé vers *Le Conjugueur* du Figaro pour obtenir la conjugaison correcte du verbe. Si aucune erreur n'est détectée, un message vert indique que le texte est correct.

La base de données

Les différentes tables

dictees

Elle contient des enregistrements de dictées. Chaque ligne représente une dictée unique identifiée par un numéro (id), accompagnée d'un titre et du contenu textuel de la dictée. Ce contenu peut être une simple liste de mots ou un petit texte. L'objectif principal de cette table est de fournir une base de référence pour les évaluations : chaque dictée saisie ici est ensuite utilisée pour comparer les réponses des élèves et identifier leurs erreurs à l'aide d'un traitement automatique (calcul de distance de Levenshtein, analyse du lexique).

Les dictées sont ajoutées à cette table via une interface utilisateur depuis la page *create_dictee.php* dans l'application. Cette page permet à un enseignant de rédiger un nouveau titre et d'entrer le contenu de la dictée puis de l'enregistrer en base de données. Il est également possible d'ajouter ou de modifier directement une dictée en utilisant une requête SQL. Cette méthode est plus rapide et utile lorsqu'on veut corriger ou insérer plusieurs dictées en bloc.

Chaque dictée est construite autour d'un thème – par exemple, « L'hiver », « Les animaux », ou « L'école ».

eleves

Cette table contient trois colonnes principales : id, nom, et niveau. La colonne id est un identifiant unique pour chaque élève et utilisé comme clé primaire. La colonne nom contient le prénom de l'élève, tandis que niveau indique sa classe scolaire (CM1 ou CE2).

Elle est alimentée directement via une requête SQL dans phpMyAdmin.

La base de données

Les différentes tables

lexique

Elle contient plusieurs colonnes : mot, categorie, lemme et desinence. La colonne *mot* stocke chaque mot tel qu'il peut apparaître dans une dictée. La colonne *categorie* indique la nature grammaticale du mot (par exemple nom, verbe, adjectif, ponctuation). La colonne *lemme* représente la forme de base du mot (par exemple « courir » pour « court »). Enfin, la colonne *desinence* contient la terminaison grammaticale du mot, utilisée pour détecter les erreurs de conjugaison ou de flexion (par exemple, t pour « court », s pour « les »).

Cette table est essentielle pour identifier précisément les types d'erreurs que fait un élève dans une dictée. Elle permet de distinguer une faute d'orthographe d'une erreur de ponctuation ou de désinence. Lorsqu'un mot est analysé, il est comparé à cette table pour savoir s'il est connu, dans quelle catégorie il se situe, et comment il se forme.

Ce lexique peut être enrichi par des requêtes SQL.

reponses

Cette table stocke toutes les réponses des élèves aux différentes dictées, ainsi que l'analyse des erreurs détectées automatiquement par le système. Elle contient plusieurs colonnes : *id* identifiant unique de chaque réponse, *id_dictee* identifie à quelle dictée cette réponse est liée en lien avec la table *dictees*, *id_eleve* identifie quel élève a fourni cette réponse en lien avec la table *eleves, reponse* contient le texte exact que l'élève a écrit, *erreurs* regroupe les fautes détectées par catégorie (nom, verbe, ponctuation..) avec une explication et *date_saisie* indique la date à laquelle la dictée a été faite.

Les données de cette table sont ajoutées automatiquement depuis la page de saisie *passation.php*.