Contrepèterie : L’art de décaler les sons.

Tuteur : Pascal LAFOURCADE

Réalisé par : Anthony COSTA, Corentin CAMPIDELLI, Adrien BLAY, Thomas GOUNON, Pierre ERBLAND, Jean-Baptiste VAYSSADE

Année Universitaire 2019/2020
IUT de Clermont-Ferrand, Département Informatique.

REMERCIEMENTS

Avant d'entamer ce rapport, nous profitons de l'occasion pour remercier tout d'abord notre professeur Monsieur Pascal Lafourcade qui n'a pas cessé de nous encourager pendant la durée du projet, ainsi pour sa générosité en matière de formation et d'encadrement.

Nous le remercions également pour l'aide et les conseils concernant les missions évoquées dans ce rapport, qu'il nous a apporté lors des différents suivis, et la confiance qu'il nous a témoignée.

Nous tenions également à lui dire merci pour nous avoir prêté son livre sur les contrepèteries durant la durée du projet : *La Bible du contrepet de Joël MARTIN*. Ce livre nous a beaucoup aidé notamment pour comprendre les différents types de contrepèteries qui existaient ainsi que par ses nombreux exemples de contrepèteries qui nous aurons énormément servi dans notre projet.

Nous le remercions aussi de nous héberger notre site WEB sur son adresse personnelle (http://sancy.univ-bpclermont.fr/~lafourcade/contrepeteries/). Cela nous a permis d'avancer dans le projet en pouvant tester notre code côté client.

SOMMAIRE

1. INTRODUCTION

    1.1. Présentation du projet
    1.2. Objectif du projet

2. MÉTHODOLOGIE

    2.1. Choix des langages
    2.2. Méthode de résolution
    2.3 Méthode de travail

3.RÉALISATION DU PROJET

    3.1. La partie Python

        3.1.1. Obtention des dictionnaires de mots
        3.1.2. Aide à la contrepèterie
            3.1.2.1. 1er algorithme
            3.1.2.2. Test de performance
            3.1.2.3. 2ème Algorithme
            3.1.2.4. 3ème Algorithme
            3.1.2.5. Fonctions utiles

    3.2. La partie WEB

        3.2.1. La transition du python à Javascript
        3.2.2. Création du site vitrine
            3.2.1.1. Partie HTML, CSS, Bootstrap
            3.2.1.2. Partie JavaScript
        3.2.3. Chargement dictionnaire côté client

4.CONCLUSION

5.RÉSUMÉ EN ANGLAIS

6. WEBOGRAPHIE

7. LEXIQUE

8. ANNEXE

1. INTRODUCTION

1.1. Présentation du projet

Dans le cadre du projet tuteuré de seconde année de DUT Informatique, nous avons obtenu le sujet « Contrepèteries, l'art de décaler les sons » dont le tuteur est M.LAFOURCADE. Ce projet n'est donc pas dépendant d'une entreprise. Afin de réaliser ce projet, notre équipe est constituée de six personnes : ERBLAND Pierre, COSTA Anthony, CAMPIDELLI Corentin, BLAY Adrien, VAYSSADE Jean-Baptiste et GOUNON Thomas.

Avant de détailler le sujet et les objectifs de ce projet, il nous semble important de définir ce qu'est une contrepèterie. Le Larousse définit une contrepèterie ainsi : « Inversion de l'ordre des syllabes, des lettres ou des mots qui, modifiant le sens, produit des phrases burlesques ou grivoises. ».
Une contrepèterie est donc un changement de deux ou plusieurs phonèmes dans une phrase. L'objectif final est d'obtenir une nouvelle phrase avec un sens complètement différent, cette nouvelle phrase n'est pas forcément orthographiquement correcte mais elle est oralement compréhensible. Par exemple : « Il y a 2 sortes d'orateurs... les courts et les longs » qui donne : « Il y a 2 sortes d'orateurs... les lourts et les congs ».

Dans notre projet, on peut retrouver plusieurs types de contrepèteries :

- Le premier type est lorsque nous échangeons la première lettre d'un mot :

Exemple : Nous sommes des souillons, cachons-le bien.
Réponse : Nous sommes des couillons, sachons-le bien.

- Ensuite, le second type correspond à lorsque nous allons permuter deux lettres à l'intérieur de mots :

Exemple : Papi est gras
Réponse : Papa est gris

- L'avant dernier type de contrepèterie est une permutation de sons entre deux mots :

Exemple : Il fait beau et chaud.
Réponse : Il fait chaud et beau.

- Enfin, nous avons la permutation de sons entre plusieurs mots :

Exemple : Chaque dimanche, je m'en vais en Loire car ma Muse aime les jolis châteaux.
Réponse à vous de trouver.

1.2. Objectif du projet

Le sujet « Contrepèteries, l'art de décaler les sons » étant très vaste, il est important de souligner les attentes de notre tuteur. Ainsi, le projet avait deux buts principaux.
Pour ce premier objectif, nous devions à partir d'une phrase saisie par l'utilisateur, vérifier si elle contenait une contrepèterie ou non. Si oui, nous devions l'afficher à l'utilisateur. Par exemple si l'utilisateur rentre : La cuvette est remplie de bouillon. Notre algorithme va donc vérifier si la phrase contient une contrepèterie, ici c'est le bien le cas donc nous allons afficher la contrepèterie qui est : La buvette est remplie de couillon.
Le second objectif est de générer des contrepèteries, si possible, à partir d'une phrase donnée par l'utilisateur. Ces deux objectifs étant destinés à un utilisateur, il est donc nécessaire d'avoir un site Web afin que cet utilisateur puisse les utiliser.
Aussi, des objectifs supplémentaires nous ont été demandés.
Le premier est une aide à la contrepèterie, sur le site web, nous avons une page où lorsque l'utilisateur saisit un mot, nous affichons la liste des mots qui peut former une contrepèterie avec ce dernier.
Nous devions aussi réaliser un jeu sous forme de résolution de contrepèteries ayant un but pédagogique.

2. MÉTHODOLOGIE

2.1. Choix des langages

Afin de résoudre ce sujet nous avons dû faire des choix au niveau des langages que nous allions utiliser. Tout d'abord, notre site allait être codé en HTML5 et CSS3 avec le Framework Bootstrap ainsi que JavaScript pour les animations notamment dans le jeu. Ensuite, pour le développement de nos algorithmes, nous avons dans un premier temps choisi Python afin de programmer la logique des algorithmes et de tester leurs fonctionnalités sur un langage que nous connaissions tous, pour ensuite réécrire la logique de ces algorithmes en JavaScript, qui est efficace côté client. Nous sommes conscients que l'utilisation de deux langages pour coder les mêmes algorithmes n'était pas la solution optimale mais étant donné qu'aucun membre du groupe n'avait de réelles connaissances en JavaScript, nous avons fait le choix d'avoir des tests et des résultats rapidement avec Python.

2.2. Méthode de résolution

Nous avons traité les objectifs un à un tout en sachant que les algorithmes utilisés d'un objectif à l'autre allaient être similaires. C'est pourquoi nous avons préféré séparer les algorithmes en fonction des types de contrepèterie. Nous avons donc défini quatre types de contrepèterie et à chaque type correspond à un algorithme. Ainsi le premier type correspond à une contrepèterie à la permutation se fait entre la première lettre de deux mots. Le second type est similaire au premier mais la permutation se fait n'importe où dans le mot. Le troisième concerne la permutation de deux

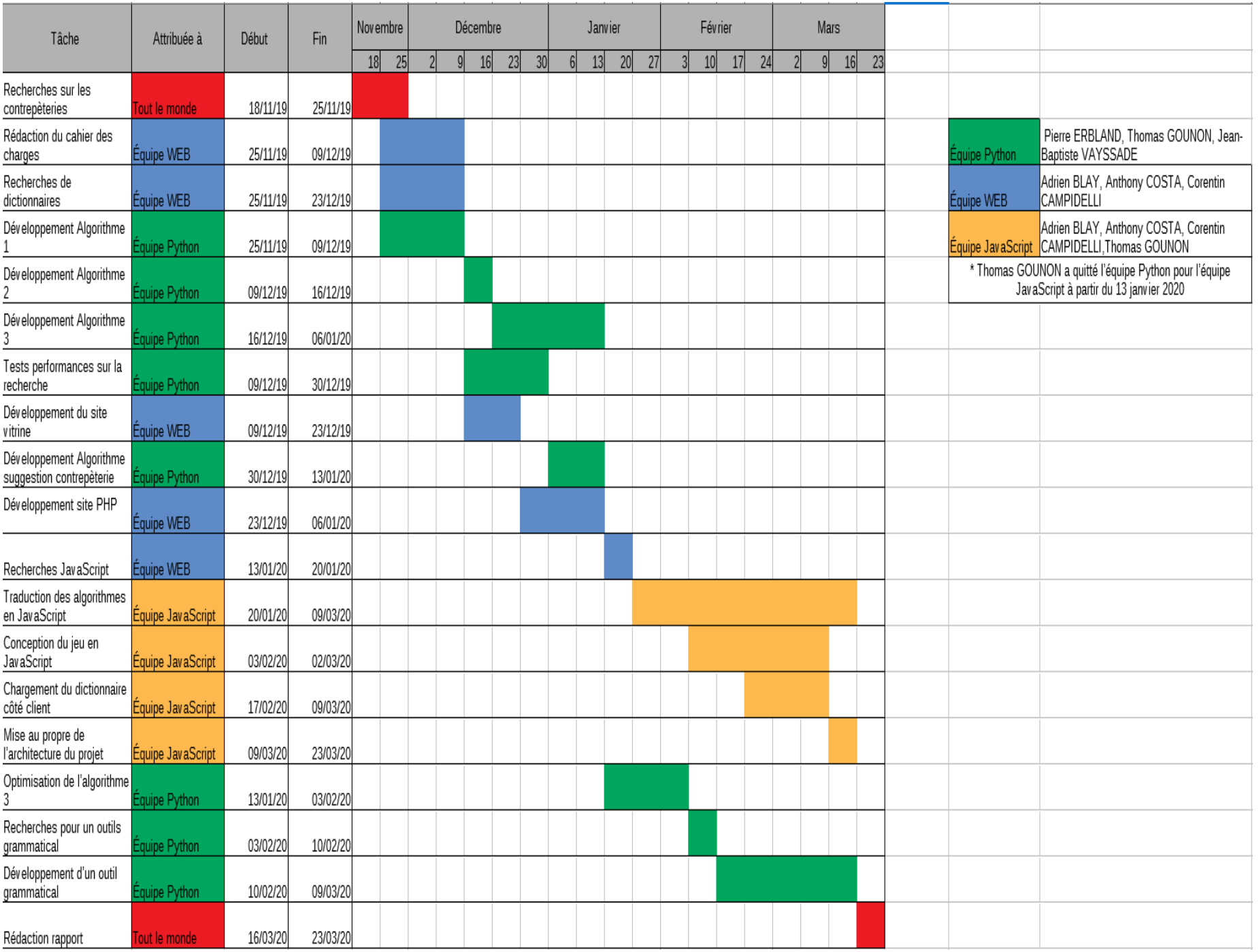
sons entre les mots. Enfin notre dernier type permet de gérer les contrepèteries ayant plus de deux permutations. Notre objectif est de faire fonctionner ces algorithmes un par un afin de toujours avoir des résultats. Une fois que ces algorithmes seront fonctionnels en Python, nous les traduirons en JavaScript afin qu'ils soient utilisables côté client depuis le site WEB.

2.3 Méthode de travail

Afin de mener à bien ce projet, nous avons formé des équipes de travail pour faciliter l'avancée du projet. Ainsi une équipe " Python ", composée de Pierre ERBLAND, Jean-Baptiste VAYSSADE et Thomas GOUNON et une équipe " WEB ", composée de Corentin CAMPIDELLI, Adrien BLAY et Anthony Costa ont vu le jour. Ces équipes ne sont évidemment pas fixes et des changements ont été effectués. Une fois le développement des algorithmes Python terminés, une équipe JavaScript a été créée sur la base de l'équipe WEB plus Thomas GOUNON. Cette équipe doit traduire les algorithmes Python existants en JavaScript ainsi que créer le jeu.

Vous trouverez ci-joint notre diagramme de Gantt réel avec les tâches effectuées par les différentes équipes. Nous n'avons pas de diagramme de Gantt prévisionnel car nous avons travaillé en utilisant la méthode Agile. Cette méthode consiste à diviser le projet principal en sous-projet traités indépendamment et où l'acceptation du changement est une valeur fondamentale. C'est pourquoi nous avions une idée du déroulement du projet mais de façon précise. La planification des séances se faisait la semaine précédente, cela nous a permis de se focaliser sur du court terme et de ne pas avoir de pression concernant le résultat final. Nous nous sommes cependant fixer des dates limites pour accomplir certaines tâches afin de rester dans le temps imparti. Par exemple, notre objectif pour les algorithmes Python était d'avoir terminé pour la première soutenance du 21 janvier 2020.

Diagramme de Gantt



Tâche	Attribuée à	Début	Fin
Recherches sur les contrepèteries	Tout le monde	18/11/19	25/11/19
Rédaction du cahier des charges	Équipe WEB	25/11/19	09/12/19
Recherches de dictionnaires	Équipe WEB	25/11/19	23/12/19
Développement Algorithme 1	Équipe Python	25/11/19	09/12/19
Développement Algorithme 2	Équipe Python	09/12/19	16/12/19
Développement Algorithme 3	Équipe Python	16/12/19	06/01/20
Tests performances sur la recherche	Équipe Python	09/12/19	30/12/19
Développement du site vitrine	Équipe WEB	09/12/19	23/12/19
Développement Algorithme suggestion contrepèterie	Équipe Python	30/12/19	13/01/20
Développement site PHP	Équipe WEB	23/12/19	06/01/20
Recherches JavaScript	Équipe WEB	13/01/20	20/01/20
Traduction des algorithmes en JavaScript	Équipe JavaScript	20/01/20	09/03/20
Conception du jeu en JavaScript	Équipe JavaScript	03/02/20	02/03/20
Chargement du dictionnaire côté client	Équipe JavaScript	17/02/20	09/03/20
Mise au propre de l'architecture du projet	Équipe JavaScript	09/03/20	23/03/20
Optimisation de l'algorithme 3	Équipe Python	13/01/20	03/02/20
Recherches pour un outils grammatical	Équipe Python	03/02/20	10/02/20
Développement d'un outil grammatical	Équipe Python	10/02/20	09/03/20
Rédaction rapport	Tout le monde	16/03/20	23/03/20

Équipe Python	Pierre ERBLAND, Thomas GOUNON, Jean-Baptiste VAYSSADE
Équipe WEB	Adrien BLAY, Anthony COSTA, Corentin CAMPIDELLI
Équipe JavaScript	Adrien BLAY, Anthony COSTA, Corentin CAMPIDELLI,Thomas GOUNON

* Thomas GOUNON a quitté l'équipe Python pour l'équipe JavaScript à partir du 13 janvier 2020

3. RÉALISATION DU PROJET

3.1 : La partie Python

3.1.1. Obtention des dictionnaires de mots

Afin de résoudre les objectifs de ce projet, nous avons forcément besoin de dictionnaires de mots afin de tester des permutations de sons. Nous nous limitons dans un premier temps à des dictionnaires uniquement composés de mots, qui convenait au développement des premiers algorithmes. Cependant, pour que les permutations entre les sons fonctionnent, nous avions besoin des phonèmes présents dans la phrase entrée par l'utilisateur. Malheureusement, nous nous sommes d'abord heurtés à la difficulté de la langue française et il était impossible de trouver un dictionnaire de mots avec sa prononciation. Nous avons donc utilisé un dictionnaire en anglais pour tester nos algorithmes. Celui-ci a été généré grâce au module python Big Phoney qui pour chaque mot donné en paramètre, renvoie ses phonèmes.

```
phoney.phonize('pterodactyl')  # --> 'T EH2 R OW0 D AE1 K T AH0 L'
```

Chaque caractère retourné correspond donc à un son différent.

Ce dictionnaire est pertinent et permet d'obtenir des résultats pertinents. Cependant, ne pas utiliser la langue française pour notre projet aurait été un échec. C'est pourquoi nous avons simplement généré notre propre dictionnaire de mots et phonétique en parsant le site fr.wiktionary.org/.

```
1   Initialisation
2   Pour chaque mot dans le dictionnaire
3       Récupérer la page fr.wiktionary.org/wiki/mot
4           Garder uniquement la prononciation du mot
5           Stocker la prononciation dans le dictionnaire phonétique
6   Fin pour
```

Notre algorithme prenait en compte les erreurs suivantes :

- Aucune page n'existe pour ce mot
- Aucune prononciation n'est mentionnée pour ce mot

Dans les 2 cas précédents, le mot n'est pas gardé.

- Plusieurs prononciations sont mentionnées

Garder simplement la première occurrence.

Le temps de réponse à la suite d'une requête avoisinait la seconde, notre dictionnaire possédait plus de 300 000 mots. Cela aurait pris du temps. Nous avons rapidement réalisé une deuxième version avec des threads. 10 threads simultanés semblaient optimaux, pour avoir une balance entre le temps d'exécution et le risque de se faire blacklister. Enfin, le script était exécutable, puisqu'il récupérait le dernier mot parser en regardant dans le dictionnaire. Et arrêtable à tout moment car il écrivait directement dans le fichier du dictionnaire à chaque itération.

a	É'
Ã	a
abaissa	a bÉ› sa
abaissable	a bÉ› sabl
abaissables	a bÉ› sabl
abaissai	a bÉ› se
abaissaient	a bÉ› sÉ›
abaissais	a bÉ› sÉ›
abaissait	a bÉ› sÉ›
abaissÃ¢mes	a bÉ› sam
abaissant	a bÉ› sÉ'Ìƒ
abaissante	a be sÉ'Ìƒ
abaissantes	a be sÉ'Ìƒ
abaissants	a be sÉ'Ìƒ
abaissas	a bÉ› sa

Capture de notre dictionnaire après avoir été parsé

3.1.2. Aide à la contrepèterie

Nous avons également développé une aide à la contrepèterie, cette aide est disponible pour l'utilisateur. Celui-ci entre un mot et notre algorithme lui liste tous les mots susceptibles d'être complémentaire au premier pour former une contrepèterie. Pour cela, nous utilisons notre dictionnaire phonétique. Nous prenons la liste des sons du premier mot et nous les échangeons avant les sons des autres mots du dictionnaire. Ainsi, deux nouveaux mots sont formés, nous regardons donc si ces deux mots existent : s'ils sont dans le dictionnaire. Si oui, le mot entré par l'utilisateur et celui testé par notre algorithme sont complémentaire et le mot va s'afficher. Ainsi, une fois l'algorithme terminé, l'utilisateur dispose d'une liste de mots qu'il pourra utiliser dans une phrase afin de former une contrepèterie.

3.1.2.1. 1er algorithme

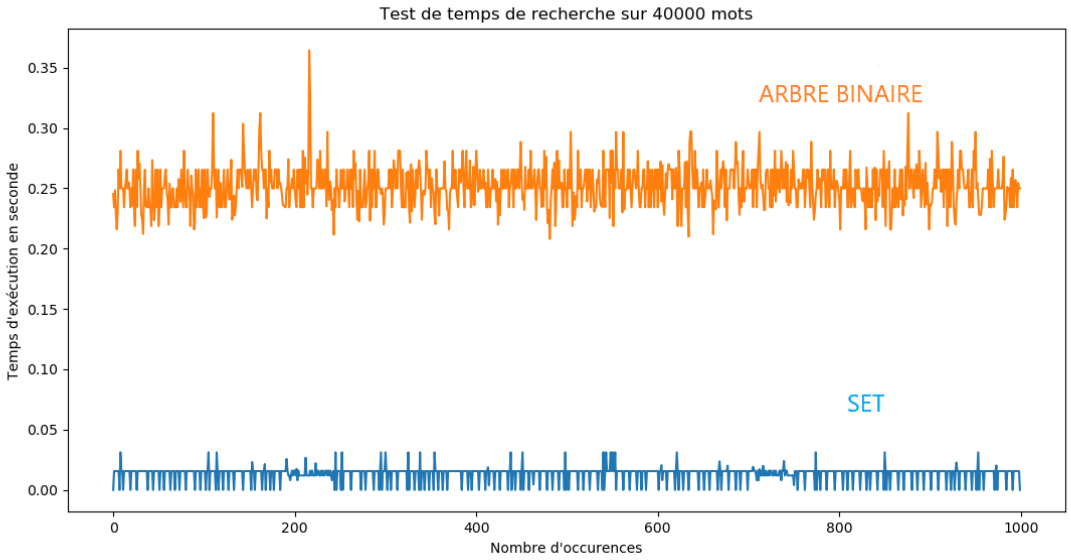
Le premier algorithme vise à vérifier si une phrase présente une contrepèterie en échangeant la première lettre de deux mots. Pour cela, nous avons utilisé un dictionnaire de mot français. La logique de notre algorithme est simple :

```
1   Initialisation(dictionnaire, phrase, output)
2   Pour chaque mot dans la phrase faire
3       Échange la première du mot avec la première lettre des autres mots
4           Si les nouveaux mots existent dans le dictionnaire
5               Ajouter la nouvelle phrase dans output
6   Fin pour
7   Retourner output
```

Cet algorithme a été rapide à mettre en place mais celui-ci était important afin de voir si les tests n'étaient pas trop longs. Nous avons pu voir que l'algorithme était très rapide donc notre logique pouvait être appliquée aux algorithmes plus complexes. Cependant, nous voulons que nos algorithmes soient le plus optimisé possible, c'est pourquoi nous avons effectué des tests de performance.

3.1.2.2. Test de performance

Afin de gagner du temps sur l'exécution de nos algorithmes, nous avons effectué des tests sur ceux-ci.
Notre premier choix pour tenter d'optimiser l'algorithme est dans la logique. Nous n'avons pas vu de réel progrès à faire au niveau de la logique de l'algorithme si ce n'est limiter les tests d'échanges en fonction des lettres testées. Au lieu de tester un changement avec toutes les autres lettres de début de mot, nous testons le changement qu'avec des lettres compatibles avec la seconde lettre du mot. Cependant nous nous sommes rendu compte que cela n'améliorait que très peu les performances de notre algorithme. C'est pourquoi nous n'avons finalement pas changé notre logique, afin de garder la même logique pour tous nos algorithmes.
Notre second choix d'optimisation portait sur la structure de stockage de notre dictionnaire. En effet, la recherche dans notre dictionnaire est ceux qui coûte le plus dans notre algorithme. Optimiser la recherche dans notre dictionnaire allait donc améliorer les performances de notre algorithme. C'est pourquoi nous avons effectué des tests sur deux structures : les arbres binaires et les SET Python. Nous avons effectué une pré-sélection car nous savions que d'autres structures classiques étaient moins efficaces que les arbres binaires pour la recherche, c'est pourquoi nous n'avons pas effectué de test sur les tableaux, les listes linéaires ou les dictionnaires. Nos choix de structure portaient uniquement sur le temps de recherche d'un élément de celle-ci étant donné que nous effectuons beaucoup de recherches dans nos algorithmes. Ainsi, après avoir chargé les structures avec le même dictionnaire, nous avons effectué des recherches sur chacune des structures avec des éléments existants dans le dictionnaire et d'autres non présents.



Les résultats furent sans appel étant donné que la recherche dans un SET était en moyenne deux fois plus rapide qu'une dans l'arbre binaire. C'est donc cette structure que nous avons utilisée pour stocker nos dictionnaires.

3.1.2.3. 2ème Algorithme

Notre deuxième algorithme est très similaire au premier, en effet celui-ci a la même logique mais il s'applique à toutes les lettres de la phrase entrée par l'utilisateur. Nous avons utilisé un dictionnaire de mots français afin de le tester.

```
Initialisation(dictionnaire, phrase, output)
Pour chaque mot dans la phrase faire
    Pour chaque lettre du mot faire
        Échange la lettre avec les lettres des autres mots
            Si les nouveaux mots existent dans le dictionnaire
                Ajouter la nouvelle phrase dans output
Fin pour
Retourner output
```

Les tests effectués avec cet algorithme sont concluants et nous arrivons à retrouver certaines contrepèteries avec celui-ci. Les tests sont également rapides. Cependant cet algorithme montre ses limites, il y a peu de contrepèteries où l'échange se limite à une lettre, nous devons donc améliorer notre algorithme et désormais échanger des sons plutôt que des lettres.

3.1.2.4. 3ème Algorithme

Le troisième algorithme ne fonctionne plus avec des lettres mais avec des sons, c'est pourquoi nous avons utilisé le dictionnaire phonétique que nous avons créé. La logique de cet algorithme est très similaire au deuxième sauf que nous ne testons pas l'échange sur des lettres mais sur les sons. Nous avons également créé une fonction retournant la liste des sons de la phrase que nous appelons au début de l'algorithme.

```
Initialisation(phrase, listeSons)
Pour chaque mot dans la phrase
    Récupérer sa prononciation dans notre dictionnaire
        Ajouter la prononciation à la liste
Fin pour
```

```
Initialisation(phrase, dictionnaire, output)
Créaion liste de son
Pour chaque son dans la liste
    Échanger le son avec les autres sons
        Si la liste de sons forme une liste de mots valide
            Ajouter la liste de sons dans output
Fin pour
```

```
Phrase: la cuvette pleine de bouillons
[['l', 'ə'], ['k', 'y', 'v', 'ɛ', 't'], ['p', 'l', 'ɛ', 'n'], ['d', 'e'], ['b', 'u', 'j', 'ɔ̃']]
normal:  [{'te'}, {'cuvellent', 'cuvelle', 'cuvelles'}, 'pleine', 'de', 'bouillons']
normal:  ['la', {'duvette', 'duvettent', 'duvettes'}, 'pleine', {'quais'}, 'bouillons']
normal:  ['la', {'buvette', 'buvettes'}, 'pleine', 'de', {'couillonnes', 'couillonne', 'couillon', 'couillons'}]
normal:  ['la', {'dévête', 'dévêtent', 'dévêtes'}, 'pleine', {'cul', 'culs'}, 'bouillons']
avec voisin gauche du mot2:  ['la', {'cuvions'}, 'pleine', {'débous', 'débout'}, {'êtes'}]
normal:  ['la', 'cuvette', {'plaident', 'plaids', 'plaide', 'plaides'}, {'née', 'nées', 'nés', 'né', 'nez'}, 'bouillons']
normal:  ['la', 'cuvette', {'plèbe', 'plèbes'}, 'de', {'nouions'}]
normal:  ['la', 'cuvette', 'pleine', {'bée', 'béent', 'bées'}, {'douions'}]
avec voisin gauche du mot1:  ['la', 'cuvette', {'plaid', 'plais', 'plaies', 'plaît', 'plaie'}, {'niés', 'niées', 'nier', '
niai', 'niez', 'niée', 'nié'}, {'boudons'}]
normal:  ['la', 'cuvette', 'pleine', {'doue', 'doux', 'douces'}, {'béions'}]
normal:  ['la', 'cuvette', 'pleine', {'dont', 'don', 'dom', 'dons'}, {'bouillez', 'bouilliez'}]
```

Aperçu du résultat de l'algorithme

Optimisation :

Certaines contrepèteries nécessitent de modifier le voisinage des 2 nouveaux mots ou de créer un nouveau mot ou de rejoindre deux mots pour exister. Exemple :

Elle revient de ferme pleine d'espoir -> Elle revient de foire pleine de sperme

Ici, le 'e' de espoir forme le "de" de la contrepèterie. Pour cela, notre algorithme va procéder comme tel :

Notre algorithme va tester les échanges de sons dans la phrase jusqu'à arriver à "Elle revient de la foire pleine d esperme", les voisins du mot vont se rejoindre pour former desperme. Finalement, chaque possibilité de découpages sera testée pour obtenir le résultat attendu.

Utilisation de la grammaire :

Comme montré sur la capture d'écran plus haute, il y a beaucoup de résultats et finalement, peu sont ceux qui sont pertinents. C'est pourquoi il paraît évident de trier les résultats. Pour ce faire, soumettre les phrases aux règles de grammaire semble être une bonne solution. Cependant nous avons trouvé aucune façon de les appliquer en l'état. En effet, il est premièrement nécessaire de connaître le type des mots pour pouvoir trier les résultats.

Ensuite, plusieurs pistes sont intéressantes en utilisant le types des mots.

- Se restreindre par exemple, aux seules phrases possédant un verbe conjugué.
- Respecter les conjugaisons, de façon, par exemple, à ne pas avoir un sujet à la 3ème personne du singulier avec un verbe conjugué au pluriel.
- Respecter les règles de grammaire simple. Par exemple qu'un verbe est précédé d'un sujet et suivi d'un complément, et non l'inverse.

3.1.2.5. Fonction utile

Il est nécessaire dans nos algorithmes, de split la prononciation des mots, c'est-à-dire de séparer un string caractère par caractère. Or, les caractères que nous utilisons pour définir les sons sont des caractères spéciaux avec des accents (ex : ɔ̃). Et en utilisant la fonction split(str) de base de python, celle-ci sépare la lettre de son accent et cela conduit à des résultats incongrus à cause de problème d'encodage. Nous avons donc dû coder notre propre fonction de split palliant ce problème.

3.2. La partie WEB

3.2.1. La transition du python à Javascript

Afin de réaliser un site WEB et pour que ce dernier soit utilisable côté client, nous avons dû transformer le code que nous avions en Python en Javascript. Nous avons eu de nombreux problèmes afin de transformer le Python en Javascript. Avec l'aide de notre tuteur, qui nous a donné un moyen d'héberger notre site, nous avons pu tester notre code.

3.2.2. Création du site vitrine

3.2.1.1. Partie HTML, CSS, Bootstrap

Nous avons rapidement créé un site vitrine, sans interaction avec un utilisateur, afin d'avoir un visuel sur la structure de notre site et qu'il soit validé rapidement car le projet ayant un but pédagogique, le site doit être ergonomique et facilement lisible. Nous avons tout d'abord créé différentes pages ayant chacune un but précis. Tout d'abord, une page définissant de façon globale ce qu'est une contrepèterie, a été créée. Ensuite, nous avons fait une liste de pages sur la classification de contrepèteries où nous avons détaillé les différents types de contrepèteries avec différents exemples tirés de *La Bible du contrepet.* Nous avons aussi réalisé une page spécialement dédiée à la génération de contrepèterie qui permet de charger un dictionnaire de mots de créer des contrepèteries à partir d'un mot donné. Enfin, nous avons fait une page pour y insérer le jeu de contrepèteries avec plusieurs modes de jeux.

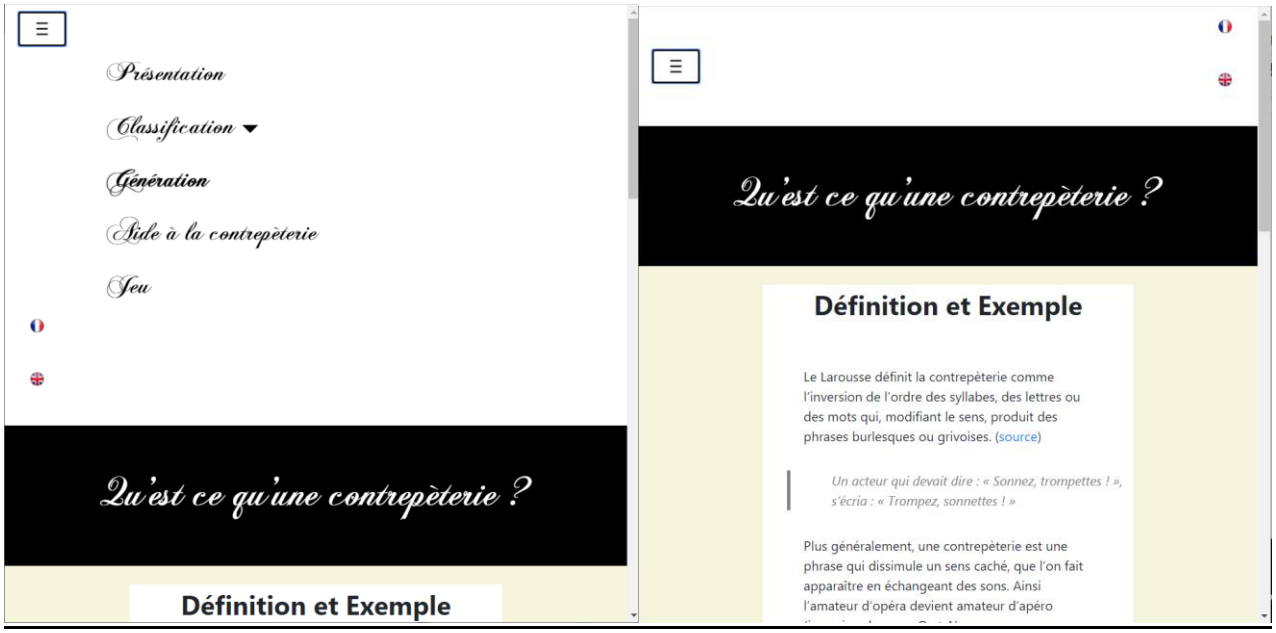
L'HTML a ainsi permis de créer la globalité de la structure du site en disposant les éléments du site de façon à ce que ce soit facile à comprendre et à utiliser. Nous avons ensuite stylisé les éléments HTML avec l'utilisation de CSS.

Le CSS a eu une place importante lors de la création du site. Le site ayant un but éducatif, il fallait que les couleurs choisies, que la police de caractère ainsi que la taille des éléments soient adaptées aux utilisateurs pour leur donner envie de venir sur le site et de le visiter. Ainsi nous avons élaboré le site en suivant ces critères. Nous avons ainsi importé notre propre police de caractère qui a un style calligraphique qui correspond bien au sujet du projet. Nous avons également utilisé des couleurs qui contrastent bien, avec le noir, le blanc et le beige, afin d'attirer l'œil des utilisateurs potentiels.

Le Bootstrap a été également très utile lors de la réalisation de ce projet car il a permis de rendre le site beaucoup plus ergonomique, esthétique, agréable et facile d'utilisation. En effet, la navigation entre les pages, grâce à la navbar, permet de rendre plus fluide cette action. L'utilisation a également permis de rendre le site responsif, c'est-à-dire d'adapter le site à la taille de l'écran. Si on reprend l'exemple de la navbar, on constate que lorsque l'écran est plus petit, une icône apparaît qui permet lorsqu'on clique dessus dérouler un menu avec les différentes pages présentes sur le site.

Une version anglaise du site est également disponible en plus de la version française, car nos algorithmes peuvent s'adapter à plusieurs langues et nous avons trouvé que cela ajoutait un plus au projet. Pour naviguer entre le site français et anglais, il suffit juste de cliquer sur la petite icône, ayant pour image le drapeau français ou anglais, en haut à droite du site.

Site vitrine en plein écran



Site vitrine "responsive"

3.2.1.2. Partie JavaScript

Le JavaScript a eu une place importante lors de la création du site. Après avoir conçu la structure du site, l'utilisation du JavaScript nous a permis de rendre le site ergonomique en le rendant plus facile d'utilisation et plus agréable pour les utilisateurs grâce notamment aux animations créées à l'aide de ce langage. Dans un premier temps, nous avons développé un script permettant, lors de l'ouverture de la page principale, d'afficher un écran titre qui se déroule automatiquement lors d'un mouvement de souris vers la définition de la contrepèterie.

Ensuite, nous avons réalisé l'entière partie du jeu en JavaScript. Pour cela, nous avons conçu plusieurs scripts, que ce soit pour le timer qui s'incrémente lorsqu'on appuie sur le bouton "Start", pour rajouter 10 sec au timer lorsqu'on trouve la bonne réponse à la contrepèterie ou encore pour passer d'une contrepèterie à une autre. Les données des contrepèteries du jeu sont stockées dans un fichier JSON qui est un gestionnaire de données très utilisé en JavaScript.

Néanmoins, la partie la plus importante est le script qui permet de détecter lorsqu'on appuie sur deux bouts de mots ou lettres si la contrepèterie est juste ou fausse. Ainsi, lorsque l'utilisateur clique, le bout de mot cliqué est mis en bleu afin d'indiquer à l'utilisateur ce qu'il a sélectionné. L'utilisateur peut ensuite cliquer à nouveau sur un autre mot qui apparaîtra également en bleu. Si l'utilisateur a trouvé les deux bons mots à permuter afin de créer une contrepèterie alors la réponse s'affiche et l'utilisateur peut passer à la contrepèterie suivante. Néanmoins, si au moins l'un des deux choix est mauvais, il peut cliquer à nouveau sur un des éléments sélectionnés en bleu afin de le désélectionner ce qui lui permettra de changer son choix et ce jusqu'à ce qu'il trouve la combinaison adéquate.

```
function verif(index){
    if(selection1==document.getElementById('span'+tabContrepeterie[index][1][0])
        &&selection2==document.getElementById('span'+tabContrepeterie[index][1][1])
        ||selection2==document.getElementById('span'+tabContrepeterie[index][1][0])
        &&selection1==document.getElementById('span'+tabContrepeterie[index][1][1])
            $('span').unbind("click");
            secondes+=10;
            let soluce= document.createElement('p');
            soluce.innerText=tabContrepeterie[index][2];
            soluce.style.color='green';
            soluce.setAttribute('id','soluce');
            document.getElementById('divSpan').append(soluce);
            let bouton= document.createElement('button');
            bouton.innerText="Contrepèterie suivante";
            bouton.id='btnNext';
            document.getElementById('divSpan').append(bouton);
            $('#btnNext').click(function(){
                console.log('erhh');
                jouer(index+1);
                selection1=null;
                selection2=null;
            });

    }
```

Code du script permettant de sélectionner une syllabe

```
function selection(j){
        temp=document.getElementById('span'+j);
        if(temp!=null){
            if(selection1==null|| selection1==''){
                selection1=temp;
                selection1.style.color='blue';
                temp=null;
            }
            else
                if(selection2==null && selection1!=temp || selection2=='' && selection1!=temp){
                    selection2=temp;
                    selection2.style.color='blue';
                    temp=null;
                }
        }
        if(selection1==temp && selection1!=null){
            selection1.style.color='black';
            selection1=null;
        }
        else
            if(selection2==temp && selection2!=null){
                selection2.style.color='black';
                selection2=null;
            }
}
```

Code du script permettant de vérifier si une contrepèterie est correcte dans le jeu JS

Nous avons également pensé que rendre le jeu ergonomique et animé inciterait les utilisateurs à y jouer. Pour cela, nous avons codé un autre script permettant, lorsqu'on appuie sur le bouton "Commencez la partie", de créer une animation qui agrandit la fenêtre du jeu, qui change le style et qui scroll sur la vue du jeu.

Tout cela permet de rendre le jeu plus agréable d'utilisation et, le site ayant un but éducatif, d'attirer les gens pour qu'ils participent au jeu afin de développer leurs connaissances dans le domaine des contrepèteries. Le jeu est également disponible en anglais avec bien évidemment des contrepèteries anglaises.

3.2.3. Chargement dictionnaire côté client

Afin que l'utilisateur puisse utiliser le vérificateur de contrepèterie, il doit avoir accès à notre dictionnaire sans quoi ses tests n'auraient aucun sens et aucun résultat. Nous devons donc charger notre dictionnaire chez le client, pour cela nous avons utilisé une bibliothèque JavaScript : PapaParse. Comme son nom l'indique, PapaParse permet de parser des fichiers CSV, qui est le format de notre dictionnaire. Ainsi, nous récupérons notre dictionnaire via un bouton sur lequel l'utilisateur devra cliquer. Ce bouton appelle une fonction qui va charger le dictionnaire qui est hébergé sur le site de M.LAFOURCADE. PapaParse traite le fichier ligne par ligne. Nous ajoutons donc ces lignes dans une variable. Cette variable contient au final les mots et leur phonétique. Nous séparons celle-ci en deux variables, une pour les mots et une pour les sons, ce qui nous permet de les manipuler plus facilement. La problématique du temps de chargement du dictionnaire était une crainte car celui-ci devait se charger vite si nous voulions que notre site soit efficace. Grâce à PapaParse, le dictionnaire se charge de manière quasiment instantanée avec une connexion décente.

4. CONCLUSION

Ce projet nous a permis de nous rendre compte que nous devons être capables de répondre aux attentes tout en devant faire face à des imprévus. Nous avons dû développer une méthodologie de travail spécifique où nous nous sommes divisé le travail en fonction de nos points forts respectifs. Afin de s'organiser dans notre projet nous avons également dû scinder le projet en étapes nous créer des "deadlines" avec les tâches respectives. Au cours de ce projet nous avons également renforcé notre sens du travail en équipe, notamment par le fait de séparer les tâches entre les différents individus et lors des sessions de travail collectives dans l'IUT et hors de l'IUT. Nous avons également vu que l'informatique être utile dans tous les domaines. Travailler sur ce projet nous a permis d'accroître notre attention lors de lectures ou de discours en cherchant des contrepèteries.

Concernant le projet, nous pouvons dire que celui-ci est un succès, nous sommes capables grâce à nos algorithmes de vérifier si une phrase contient des contrepèteries, et si elle n'en contient, de générer de possibles contrepèteries. Le système d'aide fonctionne et permet à un utilisateur de trouver des mots complémentaires à son mot entré. Une version du jeu est fonctionnelle et permet à des utilisateurs d'entraîner leur réflexion en cherchant les contrepèteries cachées. Cependant, des améliorations à ce projet sont toujours possibles. En effet, nous n'avons pas encore implémenté de vérificateur grammatical, c'est pourquoi nos algorithmes donnent de nombreuses phrases dénuées de sens. De plus, d'autres versions du jeu peuvent être ajoutées afin d'avoir un ensemble de jeu pouvant plaire à un maximum de personnes.

Et puis, ça change des maths.

5. RÉSUMÉ EN ANGLAIS

This project was made as part of our two-year university degree in computer-science (BTEC Higher National Diploma) at the Technological department of the UCA of Clermont-Ferrand In France.

The objective of our project was to create and generate spoonerisms and also to make a game on spoonerism. We had to create a website. Our tutor M. LAFOURCADE helped us during this project.

To realise this project we have decided to use the agile methodology. We made a Gantt chart to illustrate our project schedule and keep a track of what we did and how we did it.

First, we have chosen to create the algorithms with Python because it was a programming language that most of the group knew. We also knew that it would not be the final language as we wanted the final version to be coded in JavaScript. The website is programmed in HTML5 and CSS3 and we also used the Bootstrap Framework. We split the group in two, one group, composed of Pierre, Thomas and Jean-Baptiste, was working on the algorithms in Python and the other one was working on the showcase site. The algorithm team quickly came up with a first prototype to verify spoonerisms. After this, they created algorithms which were more and more advanced. There were also some performance tests on them. There are three different algorithms, the first one switch the first letter of two words. The second one does the same thing as the first one but with all the letters of the words. The last one is totally different, it tries to switch two sounds within the sentence. The website team worked on the presentation of the spoonerism and the creation of pages that would be used later. Once this was finished, the team worked on the game and on the translation of the algorithms from Python into JavaScript to allow us to implement them in our website. The game is only programmed in JavaScript, the aim is to have a spoonerism which is displayed on the screen. The player must click on the two correct syllables to go to the next spoonerism. To make it more pleasant and user-friendly we used JavaScript to create some animations.

6. WEBOGRAPHIE

- https://fr.wikipedia.org/wiki/Contrep%C3%A8terie
- https://lapoulequimue.fr/
- http://www.contrepetries.com/listes/
- https://lemomo2.pagesperso-orange.fr/contresol.htm
- https://www.planzone.fr/blog/quest-ce-que-la-methodologie-agile
- http://www.magmalemag.com/les-contrepeteries-a-placer-au-repas-de-noel-a977.html

Documentations :

- https://docs.python.org/fr/3/
- https://developer.mozilla.org/fr/docs/Web/JavaScript

7. LEXIQUE

Threads : En informatique, un fil d'exécution est la plus petite séquence d'instructions programmées pouvant être gérée indépendamment par un ordonnanceur.

Module python : Un module est un fichier composé de code Python. Il permet d'organiser logiquement du code. Le regroupement du code associé dans un module facilite la compréhension et l'utilisation du code Un module peut définir des fonctions, des classes et des variables. Un module peut également inclure du code exécutable.

Framework : En programmation informatique, un framework désigne un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel .

Parser : Exploiter un fichier sous une forme brute afin d'en tirer les informations utiles.

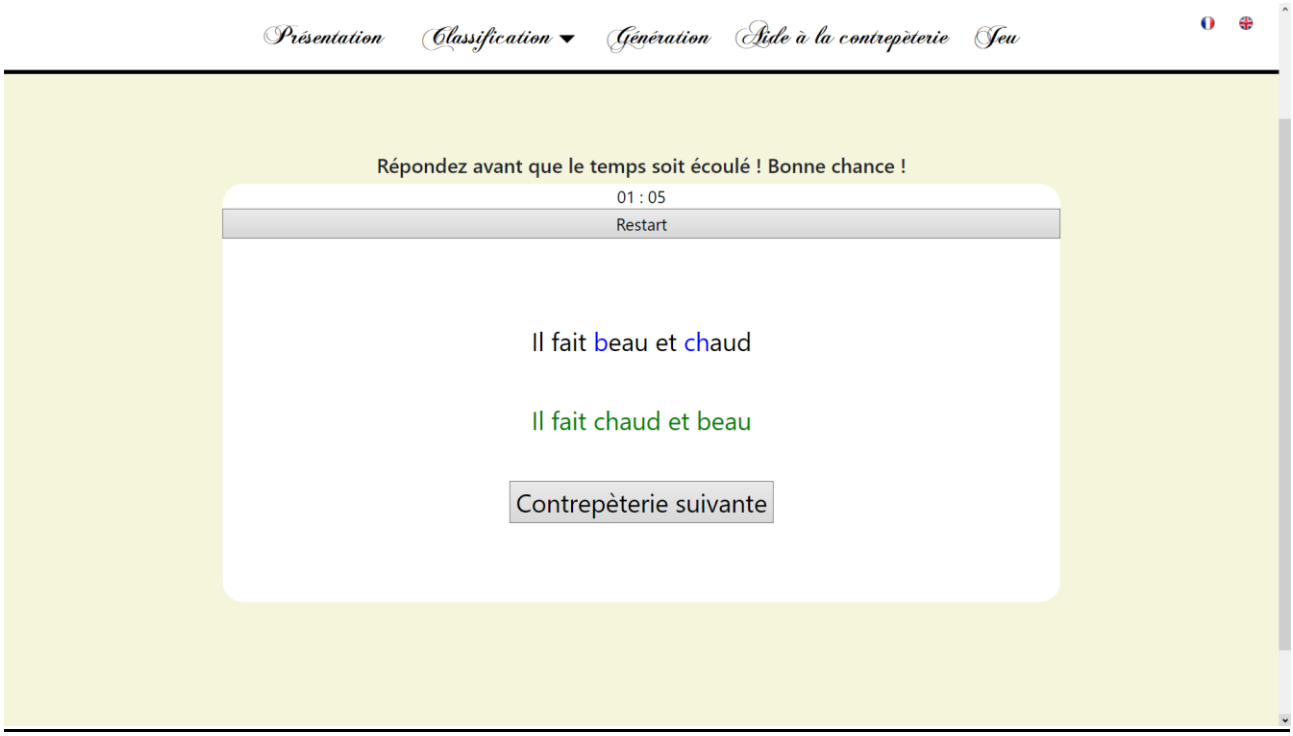
PapaParse : bibliothèque JavaScript permettant de parser des fichiers de type CSV.

Blacklister : empêcher quelqu'un d'avoir accès à une ressource.

8. ANNEXE

Page d'accueil du site en français

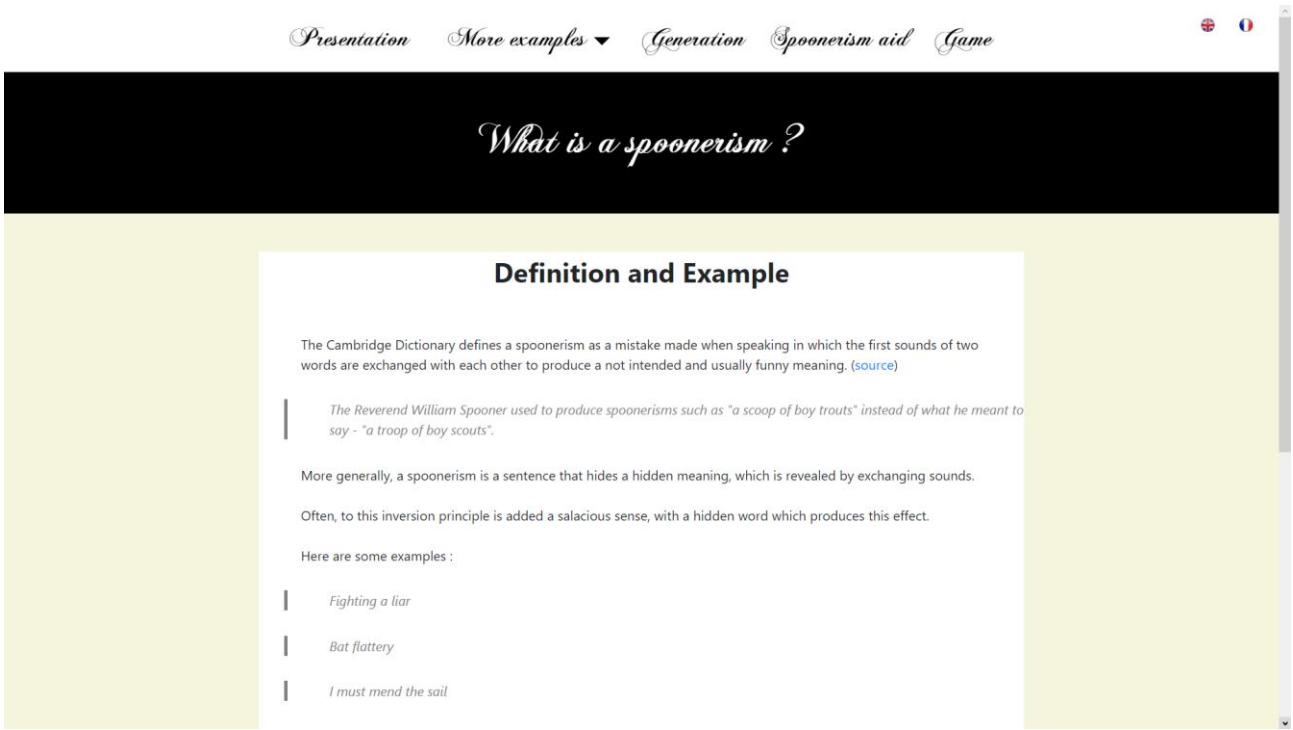
Page du lancement du jeu



Page du jeu en cours d'utilisation

Page de génération de contrepèterie

Page d'aide à la contrepèterie



Page d'accueil du site en anglais