P8 Text 2 Cloud

Nuage de mots-clés

Documentation Technique

Application accessible en ligne (bonus) :

bademba.fr/text2cloud

Code source GitHub :

github.com/bademba fr/text2cloud

Bademba SANGARE (22009816)

Master 1 Informatique et Big Data - Groupe 2
Université Paris 8 à Saint-Denis

Professeur : Nasreddine BOUHAÏ
Cours : Introduction aux Technologies Hypermedia

18 décembre 2025

1 Présentation du projet

1.1 Contexte et objectifs

P8 Text 2 Cloud est une application web permettant de générer des nuages de mots-clés à partir de textes bruts. L'objectif principal est de créer une interface **simple, efficace et intuitive** permettant à tout utilisateur de visualiser rapidement les mots les plus fréquents d'un texte.

1.2 Objectifs pédagogiques

— Comprendre et expliquer le flux de traitement d'un texte brut

— Représenter l'architecture simple d'un système Web

— Manipuler les technologies de base du Web : HTML, CSS, JavaScript et PHP

— Produire une documentation claire et structurée

1.3 Fonctionnalités

1. **Saisie du texte :** Import de fichier .txt ou saisie directe
2. **Traitement automatique :** Nettoyage, tokenisation, comptage
3. **Visualisation :** Nuage de mots dynamique et coloré
4. **Statistiques :** Nombre de mots total, uniques, gain en pourcentage
5. **Export :** Téléchargement du nuage (PNG/JPEG) et des occurrences (CSV)

1.4 Version bonus

En complément du projet local demandé, l'application a été déployée en production sur un serveur Apache (hébergeur o2switch), accessible publiquement à l'adresse bademba.fr/text2cloud.

2 Flux de traitement (Pipeline)

Le traitement des données suit un pipeline séquentiel en 7 étapes :

1. Récupération du texte → 2. Envoi serveur PHP → 3. Comptage caractères → 4. Tokenisation → 5. Nettoyage & Filtrage → 6. Comptage & Tri → 7. Génération nuage

2.1 Détail des étapes

Étape 1 - Récupération : L'utilisateur importe un fichier .txt ou colle directement le texte. JavaScript (FileReader API) charge le contenu dans le `textarea`.

Étape 2 - Transmission : Le texte est envoyé au serveur PHP via une requête POST asynchrone (Fetch API).

Étape 3 - Statistiques : PHP compte le nombre de caractères avec `mb_strlen()`.

Étape 4 - Tokenisation : La fonction `fragmenter()` découpe le texte en mots selon des séparateurs (espaces, ponctuation).

Étape 5 - Nettoyage : Conversion en minuscules, suppression des points d'interrogation aux extrémités, filtrage des mots courts et des stop words (˜1500 mots vides en français et anglais : articles, pronoms, etc.).

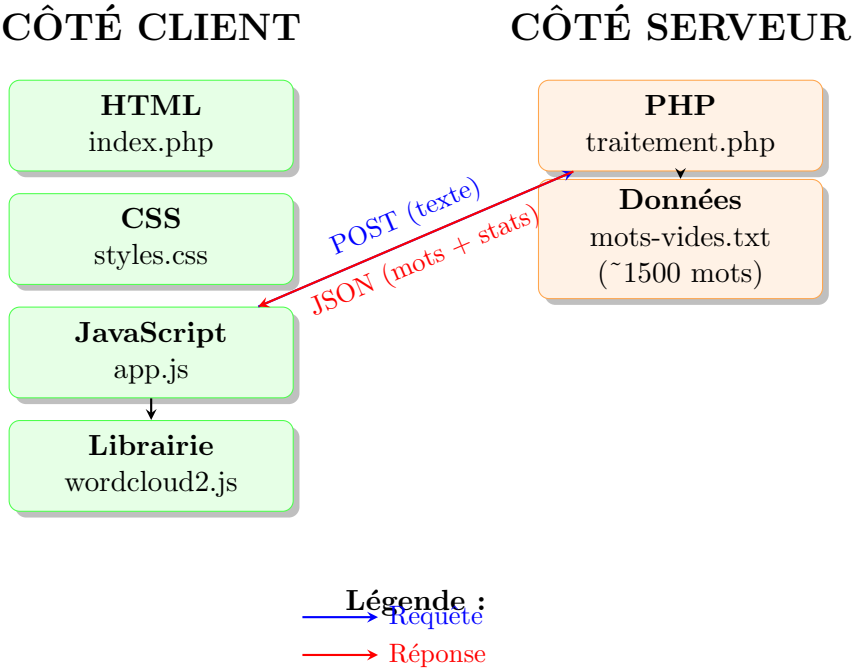
Étape 6 - Comptage : Les occurrences sont comptées dans un tableau associatif puis triées par fréquence décroissante avec `arsort()`.

Étape 7 - Visualisation : JavaScript reçoit le JSON et génère le nuage avec WordCloud2.js.

3 Diagramme d'architecture

3.1 Architecture Client-Serveur choisie

Le projet utilise une architecture **client-serveur** avec traitement PHP côté serveur. Ce choix permet de séparer clairement la logique métier du traitement textuel (serveur) et l'interface utilisateur (client).



3.2 Description des composants

Côté Client :

— **index.php** : Structure HTML de l'interface (formulaire, canvas, tableaux)

— **styles.css** : Mise en forme responsive et moderne

— **app.js** : Gestion des interactions, requêtes AJAX, génération du nuage

— **wordcloud2.js** : Bibliothèque de visualisation (via CDN)

Côté Serveur :

— **traitement.php** : Traitement du texte (tokenisation, nettoyage, comptage)

— **mots-vides.txt** : Liste de ˜1500 stop words

4 Technologies utilisées

J'ai choisi d'utiliser **PHP** pour le traitement serveur, en complément de **HTML, CSS et JavaScript** pour le front-end.

4.1 Front-end

— **HTML** : Structure de la page

— **CSS** : Mise en forme responsive

— **JavaScript** : Logique applicative, manipulation DOM

— **WordCloud2.js** : Génération du nuage de mots

4.2 Back-end

— **PHP** : Traitement du texte côté serveur

— **JSON** : Format d'échange de données

5 Conclusion

En tant qu'étudiant, j'ai particulièrement apprécié réaliser ce projet qui m'a permis de progresser en **PHP** pour le traitement et la génération du nuage de mots-clés. L'architecture client-serveur optimise la séparation des responsabilités : le serveur PHP gère le traitement des données textuelles tandis que le client JavaScript assure une interface fluide et réactive.

L'interface a été pensée pour être **simple, efficace et intuitive**, permettant une prise en main immédiate par tout utilisateur.

En bonus, le déploiement réussi sur bademba.fr/text2cloud prouve la viabilité du projet en conditions réelles.