|  |
| --- |
| [Nom de la société] |
| [Titre du document] |
| [Sous-titre du document] |

|  |
| --- |
| Zenidoc  [Date] |

Les informations contenues dans ce document pourront faire l’objet de modifications.

Aucune partie de ce manuel ne peut être reproduite ou transmise à quelque fin ou par quelque moyen que ce soit, électronique ou mécanique, sans la permission expresse de **Zenidoc** S.A.

*© 2022* ***Zenidoc*** *S.A. Tous droits réservés.*

*Zenidoc® est une marque déposée de Zenidoc S.A.*

**Remerciements**

Table des matières générale

Introduction 4

Partie A : Présentation de l’environnement du projet 4

I - Présentation générale de l’entreprise4

A – Présentation de Zenidoc4

II - Présentation des projets confiés par l’entreprise6

A – Suivi hebdomadaire et répartition du travail7

B – Technologies et outils utilisés 8

Partie B : Projet n°1 : Réalisation d’un formulaire de récupération de mots10

Table des matières10

I - Description du travail proposé11

II - Description du travail réalisé13

Partie C : Projet n°2 : Réalisation d’une interface pour la gestion de documents19

Table des matières19

I - Description du travail proposé20

II - Description du travail réalisé21

III - Commandes pour le lancement du projet26

Partie D : Conclusion, annexes et bibliographies des projets27

I - Conclusion27

II – Annexes 28

A – Projet n°128

B – Projet n°241

III - Bibliographies48

A – Projet n°148

B – Projet n°249

**Introduction**

Dans le cadre de la validation de la licence professionnelle Métiers de l’informatique : Conception, développement et tests de logiciels Parcours Agile : de l’Analyse au déploiement multi-supports que je prépare au cours de cette année universitaire 2021-2022, j’ai eu à réaliser un stage en entreprise d’une durée de douze semaines.

J’ai choisi d’effectuer mon stage au sein de Zenidoc, une PME spécialisée dans l’édition de solutions de reconnaissance vocale à destination du domaine médical, dont le siège est situé à Plan-de-Cuques. Le PDG, Francis Tibermont, m’a confié plusieurs projets web à réaliser lors de mon stage qui auront pour finalité d’être intégrés dans diverses solutions éditées par Zenidoc ou bien d’être inclus dans le processus d’amélioration de l’intelligence artificielle s’occupant de la reconnaissance vocale. La reconnaissance vocale et la transcription linguistique en général, sont des domaines qui m’attirent énormément et vers lesquels je pourrais éventuellement m’orienter en cas de poursuite d’études, de carrière dans le secteur du développement informatique.

Mon tuteur de stage, Julien Hirmke, chef de projet Web, ainsi que Nicolas Guelin, développeur, ont su être présents tout au long de mon stage afin de m’aider pour la réalisation de ces projets.

**Partie A :**

**pRésentation de l’environement du projet**

I - Présentation générale de l’entreprise

A – Présentation de Zenidoc

Fondée en 2002, Zenidoc est une société spécialisée dans la reconnaissance vocale dont le siège est situé à Plan-de-Cuques. Plusieurs de ses collaborateurs sont éparpillés aux quatre coins de la France.

Elle se spécialise dans la création de solutions logicielles, dont :

J’ai eu la chance de pouvoir être accueillie au sein de Zenidoc au sein de son siège pour une durée de douze semaines.

II - Présentation des projets confiés par l’entreprise

A – Organisation de la charge de travail

Je n’ai pas défini de planning prédéfini de mon travail car les éléments me permettant de réaliser les divers projets qui m’ont été confiées me parvenaient au et fur et à mesure. On peut ainsi dire que j’ai réalisé mes divers projets avec la méthode Agile, car je me fixais des objectifs à court terme à réaliser avant la fin de la semaine avec l’aide et la coordination de certains membres de l’équipe. Cela m’a ainsi permis de bénéficier d’une très grande agilité sur la réalisation de mon travail et de ne pas être totalement bloqué lorsque la réalisation de certaines tâches me faisait défaut.

J’ai utilisé Jira pour garder bien définir les tâches et l’objectif du sprint de la semaine en cours. Pour chaque projet sur lesquels j’ai pu travailler, le déroulement était à peu près semblable : il y avait au tout début une plus ou moins importante phase d’apprentissage des technologies que j’aillais utiliser, une phase de réalisation où je développais la solution et enfin une phase de bilan et de réflexion durant laquelle je réalisais les diverses documentations techniques des projets.

Le rapport hebdomadaire présent sur la page suivante permet de fournir le détail des activités que j’ai pu réaliser hebdomadairement.

**1 – Rapport hebdomadaire**

**Projet 1** Réalisation d’un Formulaire de récupération des Mots en JavaScript \_\_\_\_

**1ere semaine** (19/04 au 22/04)

Révisions JavaScript et apprentissage de diverses méthodes permettant la récupération des données d’une base MySQL avec uniquement du JavaScript

**2eme semaine** (25/04 au 29/04)

Appel de Mathilde (personne qui est responsable du projet) pour m’exprimer ses besoins

Visualisation du Google Sheet existant

Prototypage d’une première version du formulaire

**3eme semaine** (02/05 au 06/05)

Nouvelle version du formulaire sous forme de tableau, pour qu’il ressemble le plus possible au Google Sheet existant.

Génération automatique d’un tableau de données

Ajout fonction de vérification des données saisies par l’utilisateur

**4eme semaine** (09/05 au 13/05)

Accès à la BDD via des requêtes fetch et affichage des données dans le tableau

Ajout des contextes dans le tableau et modification de la structure de récupération des données

Envoi des données vers la BDD

**5eme semaine** (16/05 au 20/05)

Finalisation affichage des contextes

Correction fonction de vérification pour intégrer la vérification des tags multiples

Ajout fonction permettant le tri des tags par ordre alphabétique

Finalisation de certains éléments permettant d’améliorer l’intuitivité du formulaire pour l’utilisateur.

**6eme semaine** (23/05 au 27/05)

Tests pour déploiement du formulaire

Réalisation de la documentation technique du formulaire

**Projet 2** Réalisation d’une interface utilisateur en Vue.js pour la gestion de documents \_\_\_\_

**7eme semaine – 9eme semaine**(30/05 au 17/06)

Finalisation de la documentation technique du précédent projet

Apprentissage de Vue.js via Vue Mastery et mise en pratique

Recherche de solution pour la construction de l’architecture du projet

**10eme semaine**(20/06 au 24/05)

Réalisation du TreeView avec Vue.js, Vuetify et Axios

Rédaction de la documentation associée

Réalisation de l’interface d’affichage des documents

**11eme semaine**(27/06 au 01/07)

Rédaction de la documentation technique associée

Réalisation de l’interface de gestion des documents pour d’autres types de fichiers

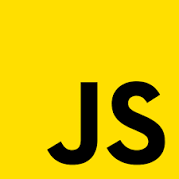
**12eme semaine** (04/07 au 08/07)

Finalisation de la documentation technique associée

B – Technologies et outils utilisés

Durant ces semaines de stage, j’ai eu l’occasion de pouvoir découvrir, utiliser et également renforcer et mettre en pratique diverses technologies dans le domaine du web. Suivant le projet, je n’ai pas utilisé les mêmes technologies, même si les deux partageaient des technologies communes comme :

* Outils de développement

** HTML, CSS, JavaScript comme langages de programmation**

*« Le HyperText Markup Language, généralement abrégé HTML ou, dans sa dernière version, HTML5, est le langage de balisage conçu pour représenter les pages web. »*

*« Les feuilles de style en cascade, généralement appelées CSS de l'anglais Cascading Style Sheets, forment un langage informatique qui décrit la présentation des documents HTML et XML. »*

*« JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives et à ce titre est une partie essentielle des applications web. »*

Le projet n°1 a été réalisé en JavaScript simple, malgré le fait que la mise en place de la relation entre la base de données MySQL et la solution développée aurait pu être faite plus rapidement, car personne n’aurait pu de m’aider en cas de soucis.

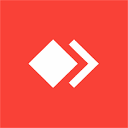
**L’IDE Visual Studio Code comme éditeur de code**

*« Visual Studio Code est un éditeur de code extensible développé par Microsoft pour Windows, Linux et macOS. Les fonctionnalités incluent la prise en charge du débogage, la mise en évidence de la syntaxe, la complétion intelligente du code, les snippets, la refactorisation du code et Git intégré. »*

**GitLab pour le versioning du code**

*« GitLab est un logiciel libre de forge basé sur git proposant les fonctionnalités de wiki, un système de suivi des bugs, l’intégration continue et la livraison continue. »*

* Outils de communication et planification

**AnyDesk pour pouvoir accéder à des bureaux distants**

Il me permettait d’avoir accès aux serveurs à distance, de pouvoir faire des démonstrations sur mon PC et pouvoir me faire dépanner à distance en cas de besoin.

**Skype Entreprise pour la communication**

Grâce à Skype, je pouvais communiquer avec les autres membres de l’équipe qui n’étaient pas présents au siège à Plan-de-Cuques, et également prendre part à des réunions de projet.

**Jira pour la planification et le suivi du projet**

Il permettait également à mes responsables de pouvoir suivre l’évolution des divers projets au fur et à mesure et me permettait d’avoir un support pour présenter mon travail de la semaine.

Par la suite, j’ai pu utiliser des technologies spécifiques à chaque projet. Une description plus spécifique de chaque technologie est disponible dans les documentations techniques des projets.

**Projet 1** Réalisation d’un Formulaire de récupération des Mots en JavaScript \_\_\_\_

* L’**API Fetch** pour la récupération et l’envoi des données du formulaire.

**Projet 2** Réalisation d’un composant Vue.js pour une arborescence de fichiers \_\_\_\_

* **Vue.js** comme framework de développement
* **Node.js** comme serveur web
* **VueCLI** pour accélérer le développement du projet en Vue.js
* **Vuetify** pour la réutilisation de certains composants
* **Vue-Router** pour la partie routage du projet
* **Axios** pour faire des appels API vers un serveur
* **Json-server** pour créer une fausse API qui renvoi des données au format JSON

**Partie B :**

**Projet n°1 : Réalisation d’un Formulaire de récupération des Mots en JavaScript**

Table des matières Projet 1

I - Description du travail proposé3

A – Le besoin3

B – La solution proposée3

C – Diagramme des Use Cases4

II - Description du travail réalisé5

A – API Fetch 5

B – Description intégration des commands5

1. Fonctionnement et intégration des commands 7

1.1 – Diagramme de séquence8

1. La base de données9

C – Diagramme d’activité10

III - Annexes11

IV - Bibliographie16

**I – Description du travail proposé**

**A – Le besoin**

La société Zenidoc édite un service de reconnaissance vocale. De ce fait, elle possède un dictionnaire de données permettant de reconnaitre l’ensemble des mots pouvant être retranscrits correctement. Cependant, du fait qu’elle cible une clientèle assez spécifique dans le domaine médical, tous les mots ne peuvent pas être tous immédiatement reconnus par le système car n’étant pas présents dans le dictionnaire de données. La reconnaissance de ces mots est ainsi faite manuellement par des auto-entrepreneuses via un fichier Google Sheet (**Annexe 6**) partagé en ligne par la personne encadrante du projet. Ce fichier partagé permet la catégorisation, l’identification, le traitement, la correction de ces mots avant leur validation par la personne encadrante du projet, permettant ainsi l’ajout de ces mots dans le dictionnaire de données.

Cependant, l’usage de ce fichier partagé n’est pas la meilleure solution car :

* Il est actuellement hébergé par Google, qui est un service commercial
* Il ne permet pas la vérification directe de la validité des champs saisis
* Il ne permet pas de garantir l’intégrité des données saisies

Il faudrait ainsi pouvoir fournir une solution permettant la saisie, la récupération et la vérification de ces données sans passer par ce fichier partagé.

**B – La solution proposée**

Réalisation d’un formulaire (**Annexe 2**) permettant de récupérer et envoyer à partir et vers une base de données MySQL, les informations saisies par les auto-entrepreneuses. Pour des raisons de praticité pour ses utilisateurs, l’interface devra être la plus fidèle à la solution actuelle (**Annexe 6**).

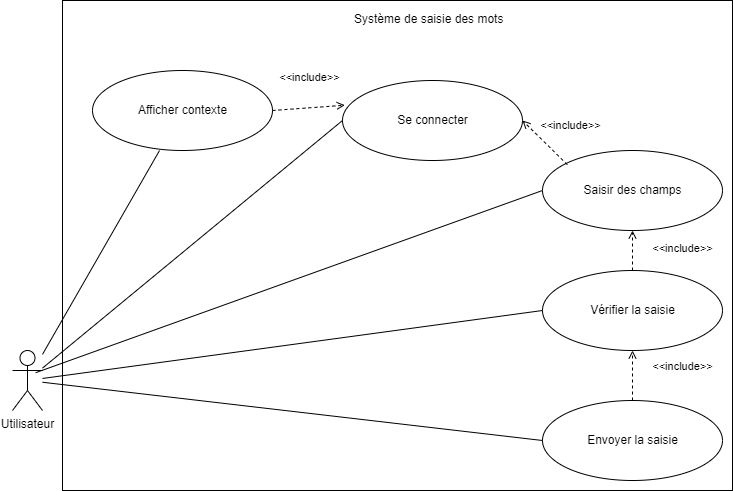
Il intègre un système d’authentification (**Annexe 1**), empêchant l’accès à toutes les personnes qui n’y seraient pas autorisées et permettant également d’afficher une liste de mots attribuées à l’utilisateur connecté. Ce formulaire pourrait ainsi inclure de contraintes sur la saisie (**Annexe 4**)qui n’étaient pas possible jusqu’à maintenant et assurer la sécurité et l’intégrité des données qui transitent par le site comme sur les TAGS. On peut alors bloquer l’envoi de la saisie si les TAGS saisis sont incorrects ainsi que pouvoir les trier par ordre alphabétique. La liste de l’ensemble des TAGS est disponible en **Annexe 8**.

Par la suite, il pourra aussi fournir des aides à la saisie pour ses utilisateurs, notamment en proposant une fonctionnalité permettant d’afficher le contexte des mots dans une phrase (**Annexe 3**).

De nombreuses fonctionnalités doivent encore être incluses dans ce formulaire, comme un système d’authentification des utilisateurs qui tentent de se connecter, pour savoir s’ils existent ou pas sur la base de données.

**C – Diagramme des Uses Cases**

Ce diagramme permet de représenter l’ensemble des cas d’utilisations qui pourront être possible à la suite de la réalisation du projet. Un cas d’utilisation représente le comportement d’un acteur (ici l’utilisateur) au sein d’un système, soit les actions qu’il peut réaliser.



**II – Description du travail réalisé**

**A – API Fetch**

L’**API Fetch** permet la récupération et l’envoi de données à travers un réseau grâce à une requête qui correspond à une ressource souhaitée.

Dans notre cas, elle contient divers paramètres d’url dont :

* Le **logon** qui sert d’identifiant, de token pour se connecter en général sur le serveur
* La **command** qui permet d’identifier l’action que l’on souhaite réaliser, par exemple récupérer la liste des mots. Elles correspondent à des requêtes SQL prédéfinies.
* Les **paramètres P1/P2/P3** dont le contenu varie en fonction de la command saisie. Seul le contenu de P1 est variable et n’est pas à saisir.



*Format de la requête*

Il existe actuellement cinq command permettant de réaliser une action prédéfinie dont :

* Récupérer le nombre de mots alloués à un utilisateur : **customVERIFLISTE**
* Attribuer un nombre de mots et des mots si ce nombre est égal à 0 : **customAttributionListe**
* Afficher la liste des mots attribués à un utilisateur : **customListeMots**
* Récupérer un contexte par rapport à l’identifiant d’un mot : **customgetcontexte**
* Envoyer la saisie de l’utilisateur vers la base de données : **customvalidationcorrection**

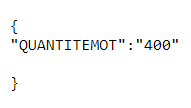
Une méthode similaire permettant la récupération d’informations depuis une base de données en utilisant l’**API Fetch** a déjà été réalisé par un autre membre de l’équipe (**Annexe 7**). Elle est utilisée sur un autre projet en permettant la récupération et l’affichage d’informations sur des patients. Il fallait donc adapter cette requête en fonction des besoins du formulaire.

**B – Intégration des commands et des données**

1. **Fonctionnement et intégration des commands**

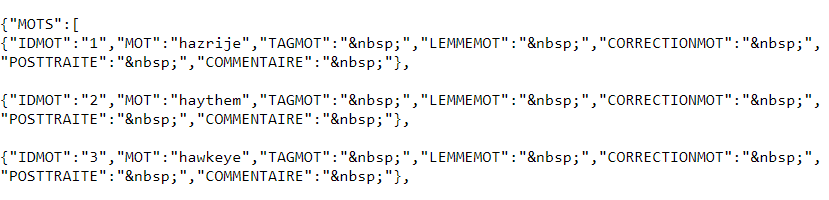
Les diverses commands énoncées dans la partie précédente sont exécutées de manière **asynchrone** les unes par rapport aux autres pour assurer une cohérence entre tous les points de vérification (comme vérifier qu’un utilisateur a des mots attribués ou pas). Les retours de ces fonctions étaient au **format JSON**, qu’il a fallu convertir pour qu’ils soient exploitables par le site. Si ces command renvoient une valeur à afficher sur le site, les fonctions JavaScript relatives à ces affichages sont appelées directement dans la méthode **fetch** JavaScript permettant la récupération des données globale. Ils ont pu être intégrées dans la réalisation du formulaire de la manière suivante :

* **customVERIFLISTE :** Suite à la connexion avec succès de l’utilisateur (en cas d’erreur ou de tentative d’accès direct à la page d’affichage des mots, on redirige l’utilisateur vers la page de connexion), l’utilisateur est automatiquement redirigé vers la page d’affichage des mots, avec en paramètre d’url l’identifiant qu’il a renseigné sur la page de connexion. Au chargement de la page, la fonction permettant l’envoi de la command **customVERIFLISTE**est lancé automatiquement et permet d’envoyer une requête pour récupérer le nombre de mots d’un utilisateur.



*Retour de la requête correspondant à l’usage de customVERIFLISTE*

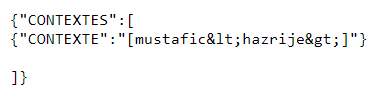
* **customAttributionListe :** Après l’exécution de **customVERIFLISTE**, sa valeur de retour (contenant le nombre de mots alloués à l’utilisateur connecté) est directement intégrée en tant que paramètre de la fonction faisant appel à la command **customAttributionListe.** Si le nombre de mots est égal à 0, alors on envoie une requête permettant l’exécution de la command **customAttributionListe** qui permet l’attribution d’un nombre de mots à l’utilisateur. Dans les autres cas, il n’y a pas besoin d’exécuter cette command et on passe directement à la récupération de la liste des mots.
* **customListeMots :** Cette command permet la récupération de la liste des mots d’un utilisateur au format JSON. La fonction JavaScript qui lui est associée permet l’affichage de ces données dans un tableau possédant des champs de saisies.



*Extrait du retour de la requête correspondant à l’usage de customListeMots*

* **customgetcontexte :** Elle permet de récupérer le contexte du mot de la ligne concernée, grâce à un bouton qui fait appel à une fonction JavaScript avec pour paramètre l’id du mot, ce qui permet de faire usage de la command. Également, le contexte récupéré est au format JSON. La fonction qui lui est associée permet l’affichage de ces données dans une fenêtre popup, qui possède une valeur par défaut si le mot sélectionné ne possède aucun contexte.

Elle est exécutée à la demande de l’utilisateur lorsqu’il appuie sur un bouton pour pouvoir économiser les ressources.



*Retour de la requête correspondant à l’usage de customgetcontexte*

* **customvalidationcorrection :** Cette command permet l’envoi des données vers la base de données MySQL. Elle se fait individuellement pour chaque mot et envoi en paramètre p3, l’ensemble des champs qui ont été saisis sous le format suivant :

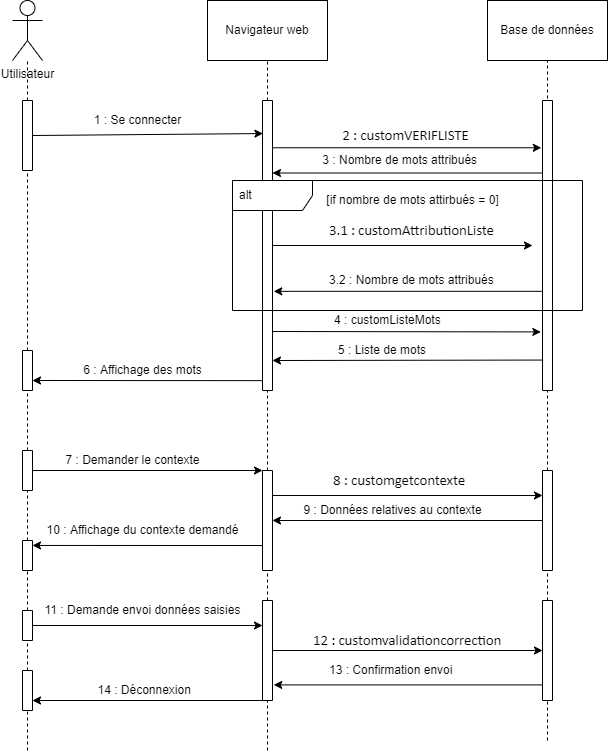
tag + ";" + lemme + ";" + correction + ";" + commentaire + ";" postTraitement

Les champs sont ainsi directement envoyés vers la base de données.

La manipulation du DOM ainsi que les appels API représentent l’essentiel du code.

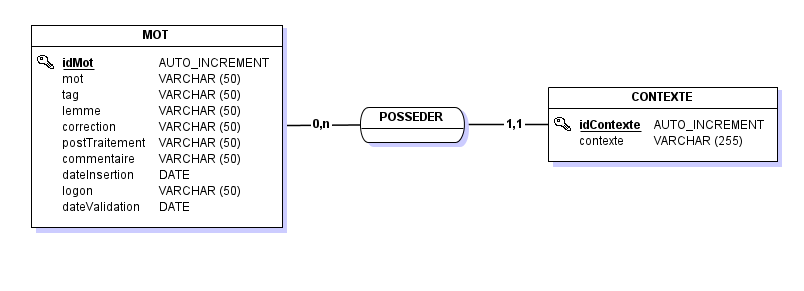
**1.1 – Diagramme de séquence**

Le diagramme de séquence suivant permet de modéliser **uniquement** les différentes interactions entre les commands et les éléments du formulaire, soit le site web contenant le formulaire ainsi que la base de données qui lui est associé.

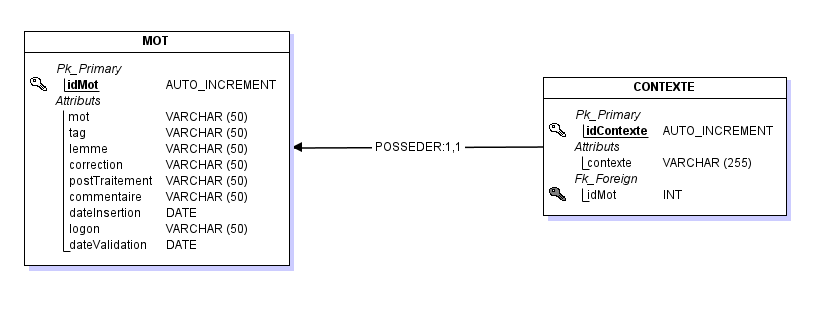


1. **La base de données**

La base de données a déjà été réalisée par mon tuteur de stage et permet de récupérer les données à afficher sur le site ou d’envoyer des données saisies lorsque l’utilisateur le demande.



*Modèle Conceptuel de la base de données*

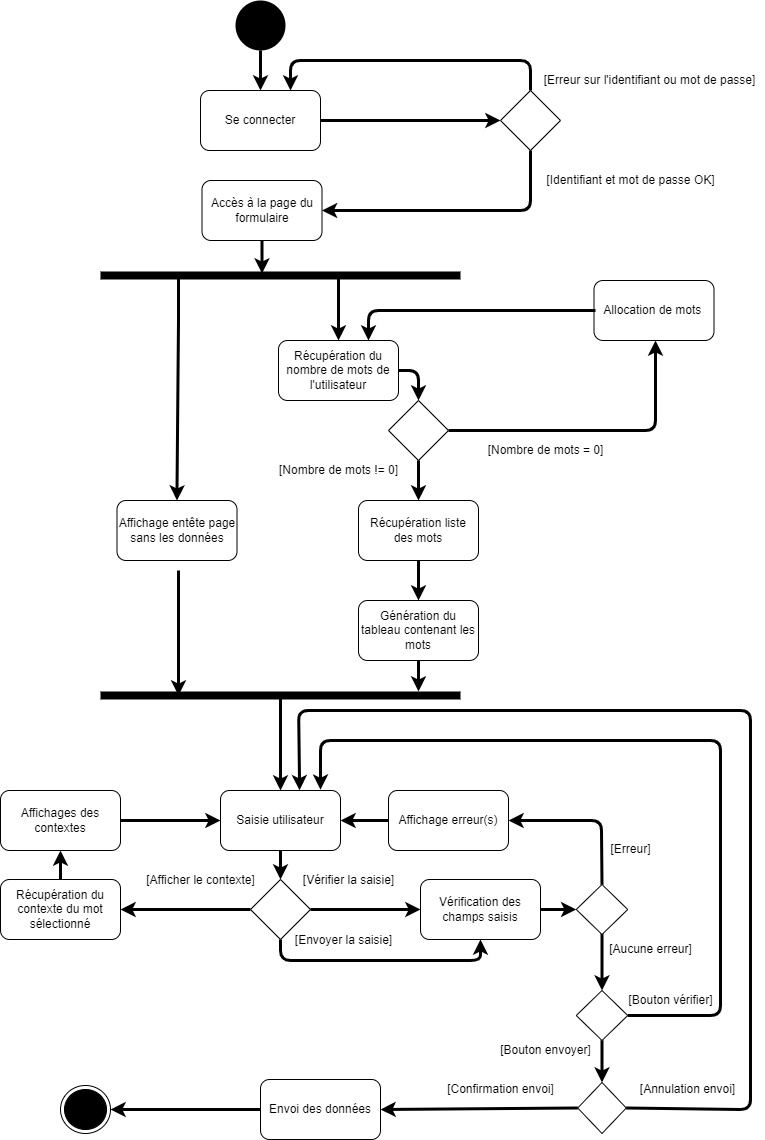


*Modèle Logique de la base de données*

Le système de gestion de base de données utilisé est **MariaDB**, une version **Open Source** de **MySQL** d’Oracle utilisant aussi le langage SQL (Structured Query Language).

**C – Diagramme d’activité**

Le **diagramme d’activité** suivant permet de décrire le fonctionnement global du site lorsqu’il est utilisé par un utilisateur. Les documents en **annexes 1 à 4** permettent de mieux visualiser chaque étape.



**Partie C :**

**Projet n°2 : Réalisation d’une interface utilisateur pour la gestion de documents en vue.JS**

Table des matières Projet 2

I - Description du travail proposé3

A – Le besoin3

B – La solution proposée3

II - Description du travail réalisé4

A – Vue.js, VueCLI et Vuetify 4

B – Axios 4

C – Vue-Router 5

D – Adaptations 5

1 – Json-server 5

2 – Versions utilisées 5

3 – Format de données à respecter5

III – Commandes pour le lancement du projet6

**I – Description du travail proposé**

**A – Le besoin**

Ce projet est la nouvelle version, la 8ème, de la solution web de Zenidoc. Elle fonctionne en lien avec le portail backend au travers d'une API dédiée, la PortalAPI. Ce frontend repose sur le framework Vue CLI qui permet d'aboutir après compilation à un ensemble de fichiers HTML/CSS/JS compatibles avec n'importe quel navigateur. Ce projet a été réalisé avec la version 2.6.11 de Vue.js. La structure des composants du projet existant est disponible en annexe 3 et 4, ce diagramme est à titre purement illustratif et ne respecte pas les normes UML.

Le but de ce projet est ainsi de reprendre un projet préexistant, en permettant l’affichage de documents sélectionnés grâce à un arborescence de fichiers intégrés, un TreeView. Les données contenues dans cette arborescence seront récupérées depuis le serveur et seront au format JSON.

**B – La solution proposée**

Réalisation d’un ensemble de composants pouvant être intégrés au projet principal. Ces composants devront pouvoir afficher une arborescence de fichiers et afficher également ces fichiers grâce à des données reçues au format JSON.

Puisqu’il est impossible de tester les composants en conditions réelles, c’est-à-dire en les intégrant directement dans le projet principal, il faudra développer une solution intégrant des outils permettant de recréer des conditions réelles (en générant le contenu du TreeView avec des données dynamiques au format JSON, et non pas statiques avec des données contenues dans le data par exemple).

L’actuel composant TreeView ainsi que le projet principal dans sa globalité possèdent une présentation syntaxique particulière. Si la présentation du code peut différer de celle que l’on peut croiser habituellement, c’est parce que l’on utilise la librairie **vue-property-decorator** (Vue Class Component) de Vue.js, permettant de créer des components ayant ainsi une syntaxe similaire à des classes. Elle permet de rendre le code plus compréhensible. La solution proposée utilise également cette librairie pour faciliter son intégration au projet principal.

**II – Description du travail réalisé**

**A – Vue.js, VueCLI et Vuetify**

Vue est un **framework évolutif** pour construire des interfaces utilisateur. À la différence des autres frameworks monolithiques, Vue a été conçu et pensé pour pouvoir être adopté de manière incrémentale. Le cœur de la bibliothèque se concentre uniquement sur la partie vue, et il est vraiment simple de l’intégrer avec d’autres bibliothèques ou projets existants.

VueCLI est un système complet permettant d’accélérer le temps de développement en Vue.js.

La nouvelle version du composant TreeView a été construite grâce à la librairie **Vuetify**. Cette librairie, permet de créer rapidement et avec facilité des interfaces utilisateurs grâce à une documentation très complète. Cette librairie est construite au-dessus de Vue.js, ce qui permet de rendre l’ensemble des composants construits avec **Vuetify** compatibles avec des composants Vue.js.

La structure et les diverses fonctionnalités de notre composant sont ainsi directement gérées et créées par **Vuetify**. Le résultat obtenu est visualisable en Annexe 1.

**B – Axios**

**Axios** est une bibliothèque JavaScript fonctionnant comme un client HTTP. Elle permet de communiquer avec des API en utilisant des requêtes. Elle permet ainsi la récupération des données depuis le serveur via une requête effectuée dans le composant en lui-même.

On envoie une requête pour pouvoir construire l’arborescence de notre TreeView grâce aux données récupérées.

Dans le cycle de vie du composant, cet appel est effectué dès l’initialisation des injections et de la réactivité. Un schéma récapitulatif de ce cycle de vie est disponible en Annexe 2.

created() {

        axios

          .get('http://localhost:3000/items')

          .then(response => {

            console.log(response.data)

            this.items = response.data

          })

          .catch(error => {

            console.log('Erreur : ', error.response)

          })

      }

L’utilisation d’Axios étant faite dans le composant parent de TreeView (SideBar), elle permet la transmission des données au composant par son élément parent via des props, comme c’est le cas dans le projet principal.

**C – Vue Router**

**Vue Router** est le router officiel pour Vue.js. Il s'intègre aisément avec Vue.js pour faire des applications mono page avec Vue.js.

Utilisé dans le projet, il permet de définir des routes ainsi que le ou les composants à afficher en fonction de la route définie. La route /ged qui utilise comme composant de sidepanel le TreeView (qui est compris dans le SideBar.

path: '/ged',

      component: AppTemplateDefault,

      children: [

        {

          path: '',

          components: {

            header: () => import('./views/app.ged/Header.vue'),

            sidepanel: () => import('./views/app.ged/SideBar.vue'),

            content: () => import('./views/app.ged/Content.vue'),

          },

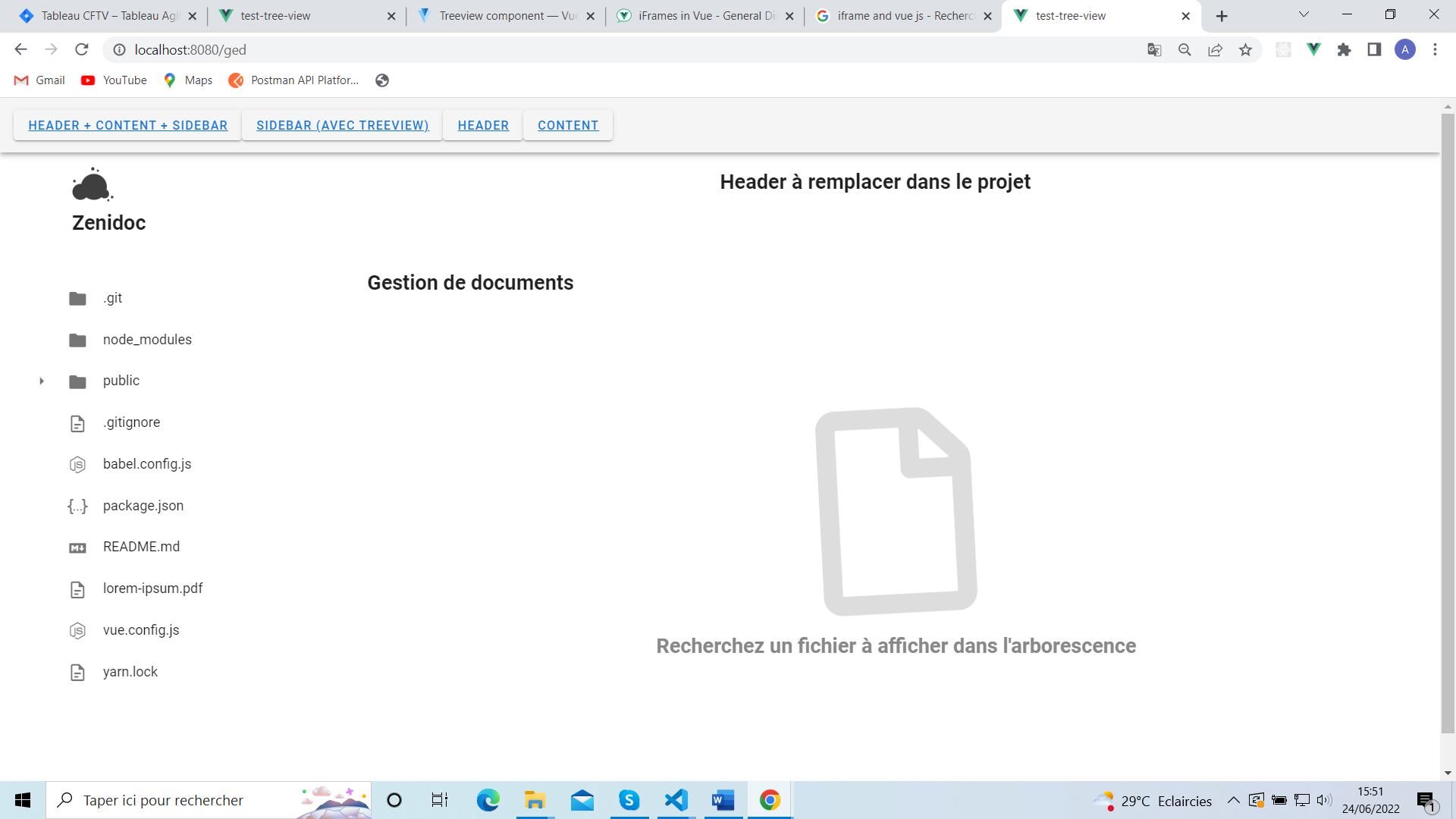
        },

      ],

    },

*Utilisation de Vue Router pour le chemin /ged dans le projet*

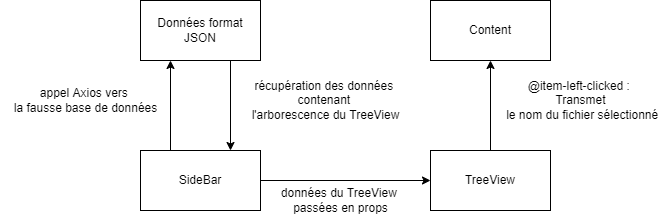
Nous retrouvons ainsi à la route /ged, l’interface suivante :



Cette interface combine l’ensemble des composants du projet, soit le SideBar (contenant le TreeView), le Content et le Header en une seule et unique page. Elle peut ainsi afficher divers types de fichiers dont la liste est disponible directement dans le composant TreeView. Un aperçu de l’ensemble de fichiers pouvant être affichés par le site est disponible en Annexe 6.

Pour résumer le fonctionnement de l’affichage du contenu très brièvement, lorsque l’on clique sur un élément du TreeView, le nom de cet élément est récupéré et est communiqué à l’ensemble des composants de l’application. Le composant Content récupère cette information et affiche l’élément correspondant.

Pour l’instant, dans le composant Content, cet élément renvoi à des liens statiques mais lorsque l’accès au serveur sera possible, on peut envisager que l’on pourra rechercher directement les liens menant à la ressource demandée grâce à un appel vers PortalAPI. Le schéma ci-dessous permet de mieux illustrer les relations qui existent entre chaque élément du projet.



**D – Adaptations**

Le composant n’étant pas, à l’heure actuelle, directement intégré au projet principal, l’utilisation de certains outils aura été nécessaire afin de permettre le bon fonctionnement de ce dernier. Ces outils seront potentiellement amenés à disparaître lorsque le composant aura été intégré au projet.

**1 – Fausse API REST : json-server**

Pour que le composant se comporte de la manière dont on le souhaite, c’est-à-dire qu’il puisse utiliser **Axios**, il était nécessaire de créer une fausse API REST renvoyant des données sur les dossiers/fichiers à afficher au format JSON. **json-server** permet ainsi de créer une fausse API sur laquelle on peut faire des appels pour récupérer des données au format JSON.

Ce serveur local ne sera alors plus nécessaire dès lors que le composant sera intégré au projet.

**2 – Détail des versions utilisées**

Pour assurer une pleine compatibilité entre le composant TreeView et le projet principal, les mêmes versions de dépendances ont été utilisées, soit pour les principales :

"vue": "^2.6.11",

"vue-json-tree-view": "^2.1.6",

"vue-loader": "^15.7.0",

"vue-property-decorator": "^8.4.0"

"axios": "^0.18.0",

"vue-router": "^3.1.6",

L’ensemble des versions utilisées est disponible dans les fichiers package-lock.json et package.json situés à la racine du projet.

**3 – Format de données à respecter**

Du fait que le TreeView soit complètement géré par une librairie externe, un format de données doit être respecté pour assurer le bon traitement des données par Vuetify.

Un exemple de données au format JSON pouvant être transmises avec succès au composant TreeView est disponible en Annexe 5.

Ces dernières doivent ainsi posséder un unique élément parent appelé ici « items », qui possèdera des enfants qui eux-mêmes pourront posséder des enfants à condition qu’ils soient tous compris dans un élément « children ». Ces derniers pourront également avoir des enfants et ainsi de suite. Un nœud doit être constitué du nom du fichier auquel il correspond suivi de son extension.

Également, concernant l’affichage de contenu, il faut que la source du iframe permettant d’afficher le contenu sélectionné, renvoi à une ressource externe et non pas une ressource stockée localement sur l’appareil. Car sinon, ce n’est pas la ressource demandée qui s’affichera mais la page actuelle.

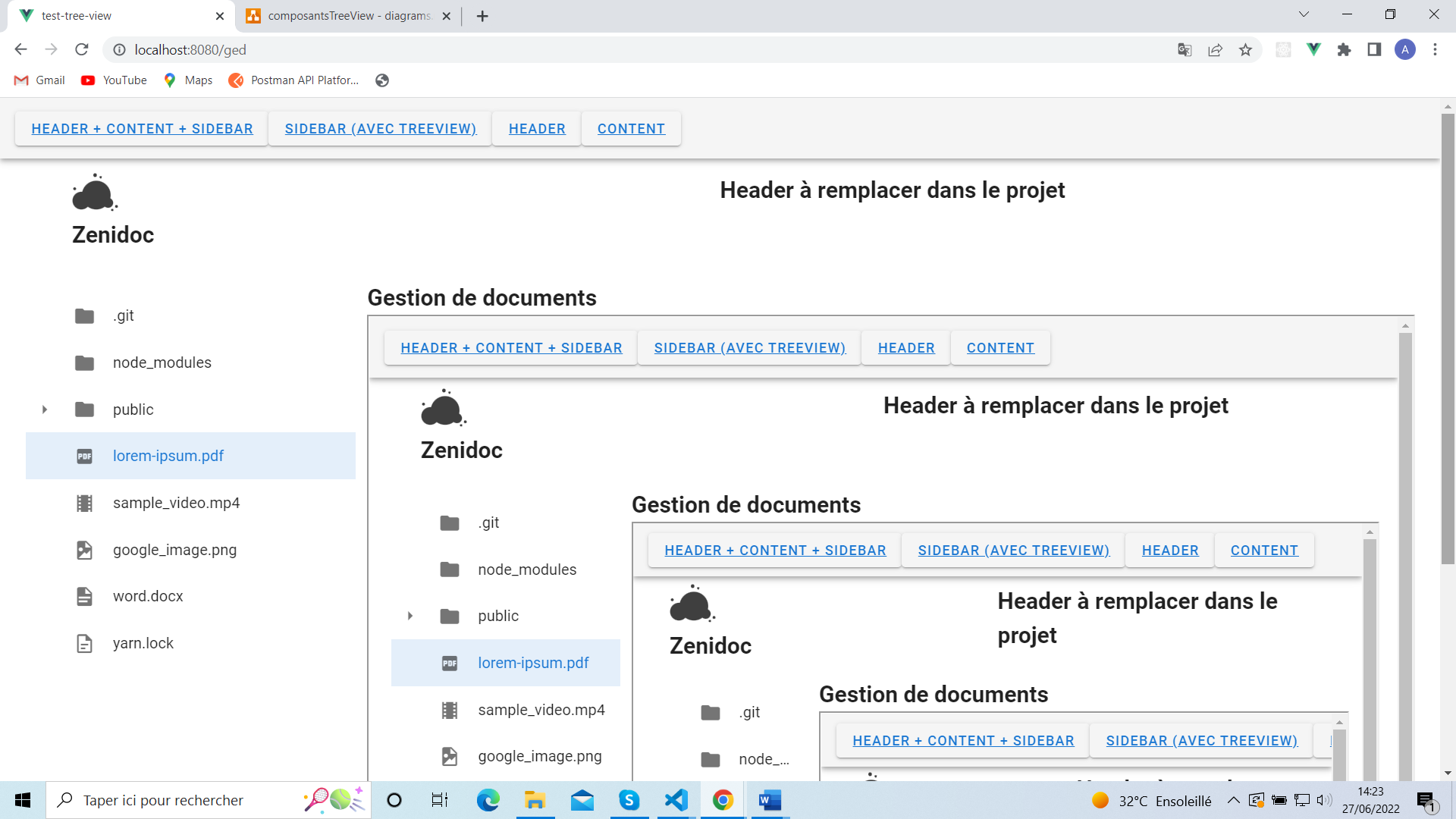
**Exemple** :

Fichier local en tant que source :

Code :

this.source = '@/fake\_files/lorem-ipsum.pdf';

Affichage Navigateur :



**III – Commandes pour le lancement du projet**

* Installer Axios

vue add axios@**0.18.0**

↑

(correspond à la version d’Axios installé sur le projet)

* Installer json-server

npm install -g json-server

* Lancer json-server

json-server --watch **db.json**

↑

(nom du fichier contenant les données au format JSON)

* Lancer l’exécution de l’application en mode développement

npm run serve

**Partie D :**

**CONCLUSION, annexes et bibliographies des projets**

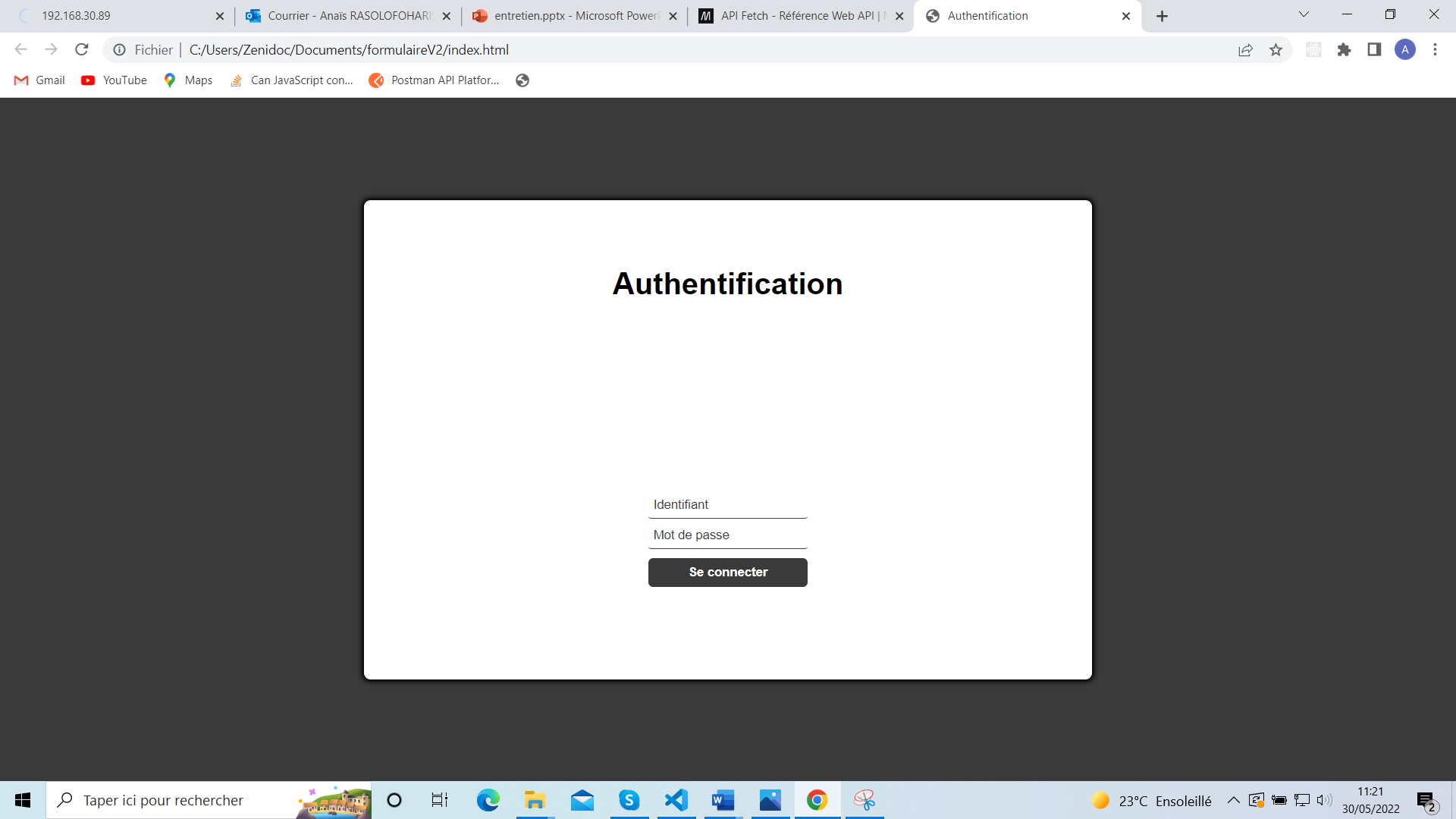
**I – Conclusion**

Au cou

**II – Annexes**

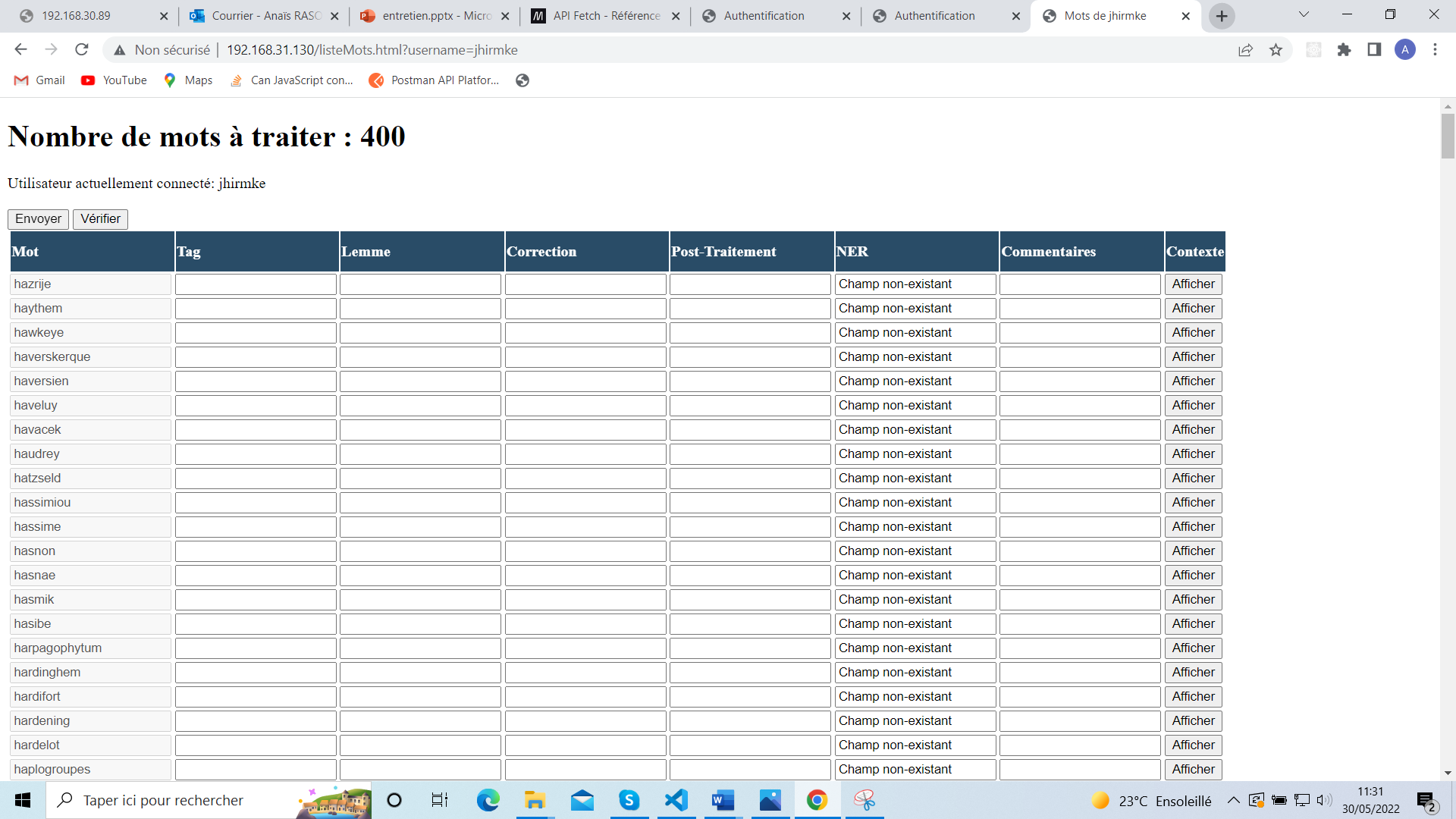
**A – Projet n°1**

**Annexe 1 :** Interface de connexion



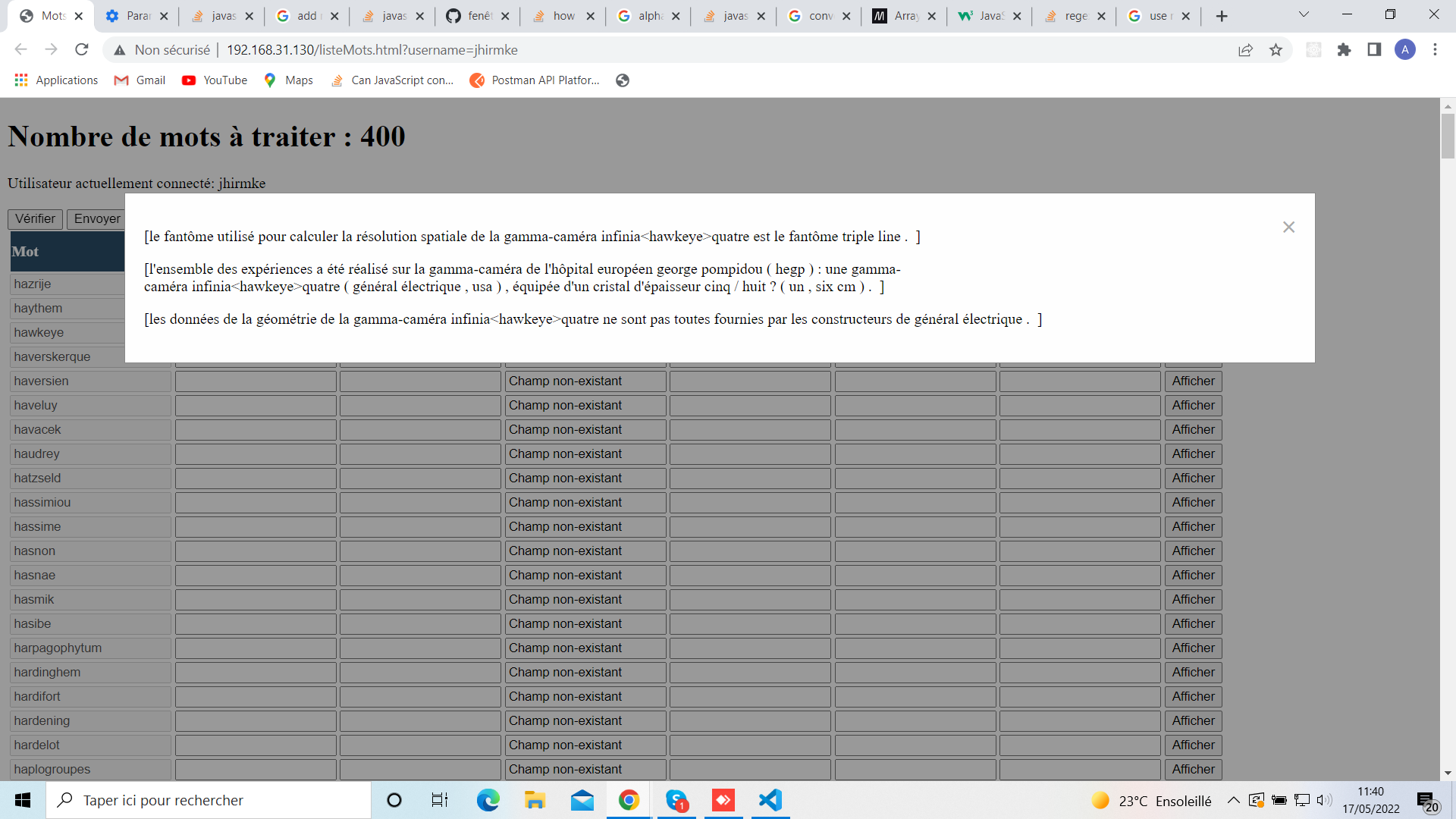
*Source : index.html*

**Annexe 2 :** Exemple du formulaire



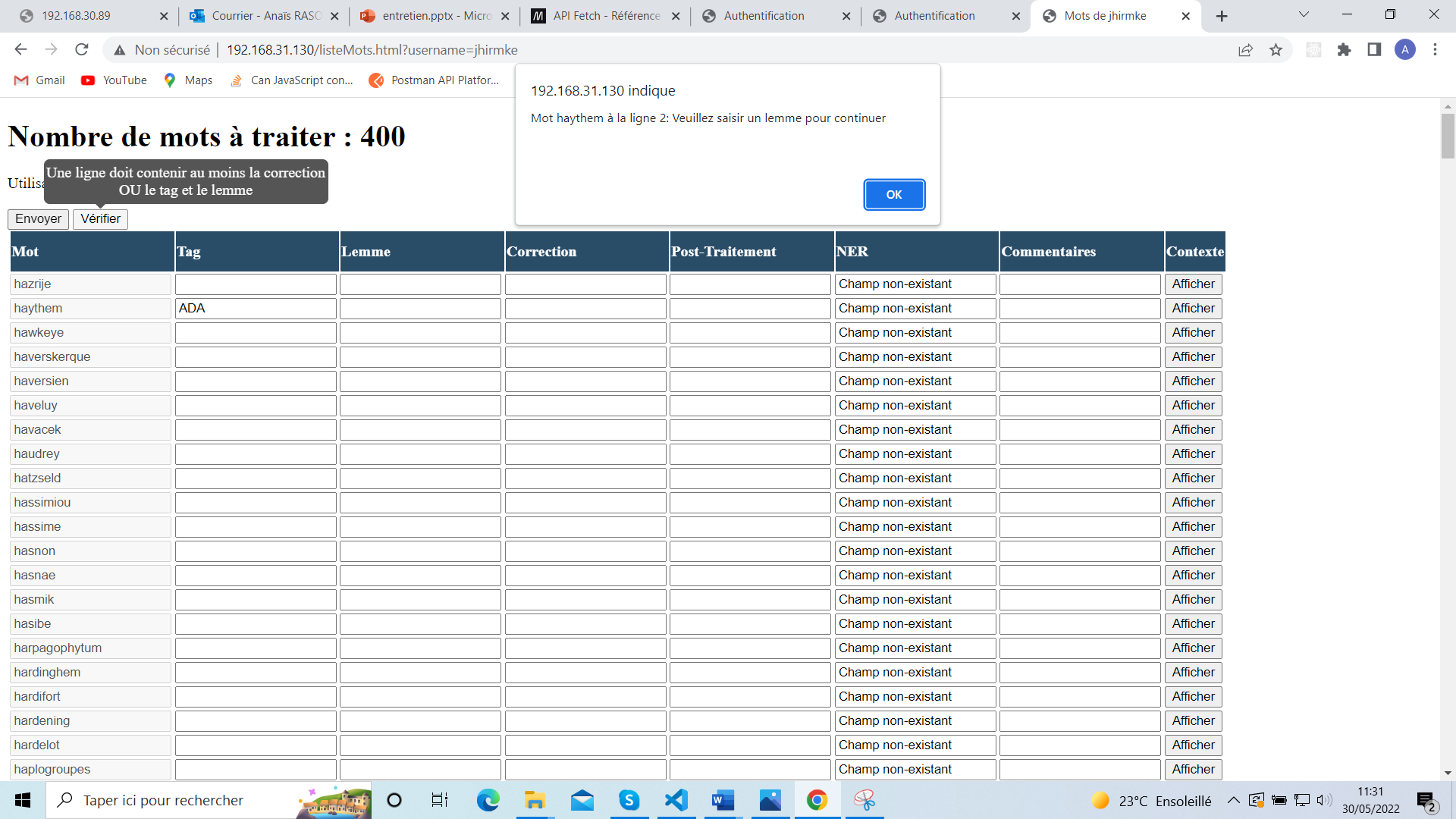
*Source : listeMots.html*

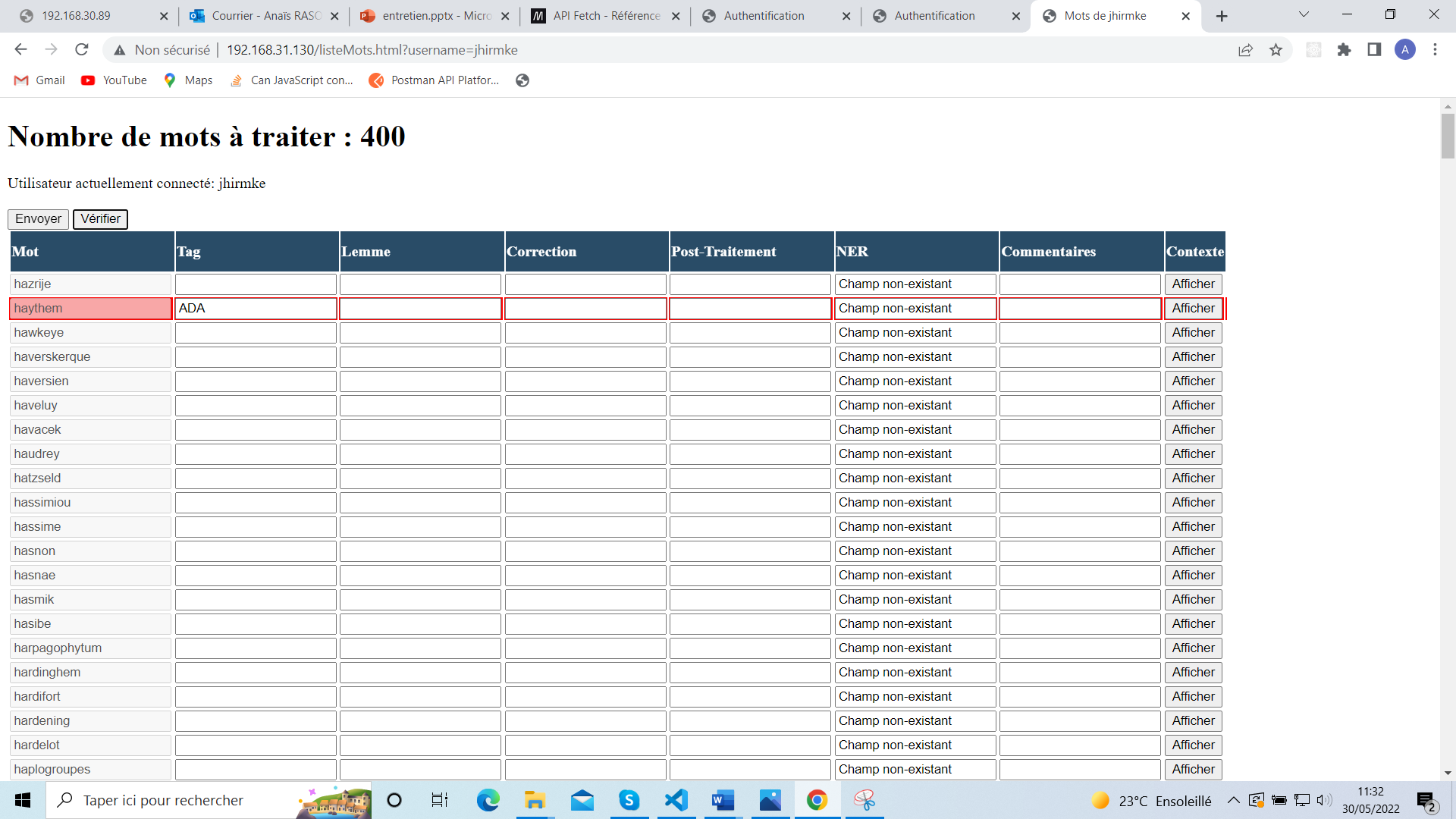
**Annexe 3 :** Exemple de contexte

****

*Source : listeMots.html*

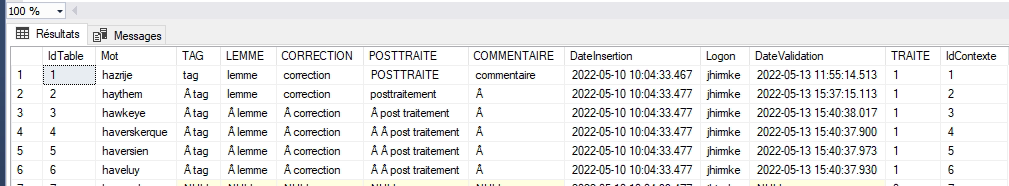
**Annexe 4 :** Exemple d’erreur



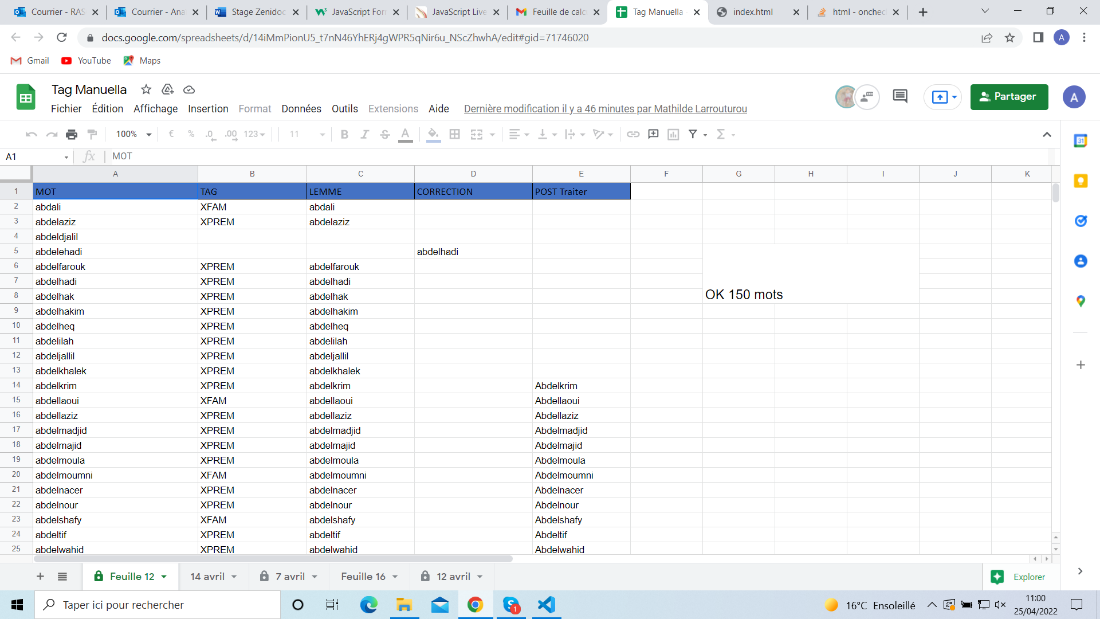


*Source : listeMots.html*

**Annexe 5 :** Champs dans la base de données

****

**Annexe 6 :** Extrait fichier Google Sheet

****

*Source : Fichier partagé sur Google*

**Annexe 7 :** Fetch

function getDatafromZenidoc(

    portal = "",

    command,

    p1 = "",

    p2,

    p3) {

    portal = "";//contient initialement l’adresse ip du portail

    const url = portal + "/portail.iz?DoCommand?logon= /\*contient la valeur du logon\*/ &command=" + command + "&p1=" + p1 + "&p2=" + p2 + "&p3=" + p3;

    fetch(url, {

        method: "GET",

    }).then(response => {

        return response.text();

    }).then(function (data) {

        var body = data

        if (body.toString().startsWith("{")) {

            result = body.replace('},\n\n]}', '}]}');

            result = JSON.parse(result)

            //identification du point d’entrée dans la fonction

            if (command == "customVERIFLISTE" || command == "customAttributionListe"){

                //on affiche le nb de mots à l'utilisateur

                document.getElementsByTagName("h1")[0].getElementsByTagName("span")[0].innerHTML = result.QUANTITEMOT;

                allocationDeMots(result.QUANTITEMOT);}

            else if (command == "customListeMots"){

                genererTableauDeMots(result);}

            else if (command == "customgetcontexte"){

                //on définit le contexte

                setContexte(result); }

        } else {

            var getFullURl = RegExp(

                '(?<=var target=").\*?(?=";)',

            );

            var getP1 = RegExp(

                '(?<=&p1=).\*?(?=&p2)',

            );

            p1 = body.match(getFullURl).toString();

            p1 = body.match(getP1).toString() ?? "";

            return getDatafromZenidoc(

                portal, command, p1, p2, p3);

        } }).catch(err => {

        console.log(err); })}

*Fichier source : js/scriptMots.js*

**Annexe 8** : Définition et liste de l’ensemble des TAGS

**TAGGS**

L’Excel présente un tableau qui contient 4 colonnes :

* MOT
* TAG
* LEMME
* NER (facultative)

L’idée c’est de pouvoir mettre des mots dans des catégories afin que le moteur puisse les reconnaitre et les analyser.

MOT :

Dans la colonne « MOT », vous aurez donc tous les mots inconnus pour le moment qui seront à catégoriser.

TAG POS :

C’est ici que nous mettons la forme grammaticale du mot. Pour cela, nous avons des « étiquettes morpho syntaxiques » (EMS) définies en annexe à l’aide de tableau.

LEMME :

Cette catégorie est là pour définir l’origine du mot. Par défaut nous la remplissons comme le mot d’origine mais IL FAUT la changer régulièrement.

Par exemple :

Le LEMME de hémorragies sera hémorragie.

Le LEMME d’analysées sera analyser.

Le LEMME le verticaux sera vertical.

ATTENTION :

1. Chaque étiquette est unique dans le sens où se sont les seules à pouvoir être utilisées pour être reconnue par le moteur. Si vous faites une faute de frappe ou que vous mettez une étiquette qui n’est pas inscrite dans les tableaux de l’annexe, votre tag ne sera pas pris en compte. Les plus communes sont celles des adjectifs ou des noms.
2. Beaucoup de mots ont plusieurs TAGS, à tous mettre, en les séparant par /. Par exemple, anti-inflammatoire est à la fois un adjectif singulier masculin ou féminin et un nom singulier masculin ou féminin. Il sera donc AFS/AMS/NFS/NMS
3. Les acronymes prennent le ou les tags des mots qu’ils remplacent. Par exemple ald signifie affection longue durée, le mot est donc NFS mais également NFP car il peut aussi être utilisé tel quel au pluriel.
4. Idem pour les abréviations, cardio peut signifier cardiologue, cardiologie, cardiologues donc il est NFS/NFP/NMS/NMP et également ADV (adverbe, dans cardio respiratoire)
5. Les préfixes sont des adverbes. Abdo, cardio, traumato prennent l’étiquette ADV en plus d’autres éventuelles.
6. Pour les médicaments, on regarde sur internet si on trouve un genre précis. Sinon on tague NMS ou NMS/NMP (si on pense qu’il peut être au pluriel).
7. Beaucoup de mots correspondent à une faute de frappe (un oubli d’espace, une permutation de lettre, un accent incorrect). Dans ce cas là, écrire HO en colonne deux ou cinq pour que nous systématisions la correction de l’erreur dans nos algorithmes. Le mot n’a pas besoin d’étiquette.

**ANNEXE :**

**TAGG :**

|  |  |  |
| --- | --- | --- |
| **CATEGORIE** | **SIGNIFICATION** | **EXEMPLE** |
| **ADVERBES** | | |
| ADV | Adverbe | Y |
| ADVNE | Adverbe ne | Ne |
| ADVPAS | Adverbe pas | Pas |
| **ADJECTIFS INDEFINIS** | | |
| AINDFP | Adjectif indéfini féminin pluriel | Toutes |
| AINDFS | Adjectif indéfini féminin singulier | Aucune |
| AINDMP | Adjectif indéfini masculin pluriel | Tous |
| AINDMS | Adjectif indéfini masculin singulier | Aucun |
| **ADJECTIFS** | | |
| AFP | Adjectif féminin pluriel | Antidémocratiques |
| AFS | Adjectif féminin singulier | Double |
| AMP | Adjectif masculin pluriel | Anciens |
| AMS | Adjectif masculin singulier | Fameux |
| **CHIFFRES** | | |
| CHIF | Chiffre | Dix |
| **CONJONCTION DE COORDINATION** | | |
| COCO | Conjonction de coordination | Et |
| **CONJONCTION SUBORDONNEE** | | |
| COSUB | Conjonction subordonnée | Si |
| **DETERMINANT** | | |
| DETFP | Déterminant féminin pluriel | Les |
| DETFS | Déterminant féminin singulier | Unes |
| DETMP | Déterminant masculin pluriel | Les |
| DETMS | Déterminant masculin singulier | L’ |
| **DETERMINANT INDEFINI** | | |
| DINTFP | Déterminant indéfini féminin pluriel | Quelles |
| DINTFS | Déterminant indéfini féminin singulier | Quelle |
| DINTMP | Déterminant indéfini masculin pluriel | Quels |
| DINTMS | Déterminant indéfini masculin singulier | Quel |

|  |  |  |  |
| --- | --- | --- | --- |
| **CATEGORIE** | **SIGNIFICATION** | | **EXEMPLE** |
| **MOT INCONNU** | | | |
| MOTINC | Mot inconnu | | Constitutionnalistes |
| **NOMS** | | | |
| NFP | Nom féminin pluriel | | Chapelles |
| NFS | Nom féminin singulier | Actualité | |
| NMP | Nom masculin pluriel | | Membres |
| NMS | Nom masculin singulier | | Article |
| **PRONOM DEMONSTRATIF** | | | |
| PDEMFP | Pronom démonstratif féminin pluriel | | Celles |
| PDEMFS | Pronom démonstratif féminin singulier | Celle | |
| PDEMMP | Pronom démonstratif masculin pluriel | | Ceux |
| PDEMMS | Pronom démonstratif masculin singulier | | Ceci |
| **PRONOM INDEFINI** | | | |
| PINDFP | Pronom indéfini féminin pluriel | | Certaines |
| PINDFS | Pronom indéfini féminin singulier | | L’une |
| PINDMP | Pronom indéfini masculin pluriel | | Plusieurs |
| PINDMS | Pronom indéfini masculin singulier | | L’un |
| **PRONOM INTERROGATIF** | | | |
| PINTFP | Pronom interrogatif féminin pluriel | | Où |
| PINTFS | Pronom interrogatif féminin singulier | | Où |
| PINTMP | Pronom interrogatif masculin pluriel | | Qui |
| **PRONOM PERSONNEL** | | | |
| PPER1P | 1er pers du pluriel | | Nous |
| PPER1S | 1er pers du singulier | | Je |
| PPER2P | 2ème pers du pluriel | | Vous |
| PPER2S | 2ème pers du singulier | | Tu |
| PPER3FP | 3ème pers du pluriel féminin | | Elles |
| PPER3FS | 3ème pers du singulier féminin | | Elle |
| PPER3MP | 3ème pers du singulier masculin | | Ils |
| PPER3MS | 3ème pers du singulier masculin | | Il |
| **PRONOM OBJET** | | | |
| PPOBJFP | Pronom objet féminin pluriel | | En |
| PPOBJFS | Pronom objet féminin singulier | | Lui |
| PPOBJMP | Pronom objet masculin pluriel | | Vous |
| PPOBJMS | Pronom objet masculin singulier | | Lui |
| **PRONOM REFLECHI** | | | |
| PREFFS | Pronom réfléchi féminin singulier | | S’ |
| PREFMP | Pronom réfléchi masculin pluriel | | S’ |
| PREFMS | Pronom réfléchi masculin singulier | | Se |

|  |  |  |
| --- | --- | --- |
| **CATEGORIE** | **SIGNIFICATION** | **EXEMPLE** |
| **PRONOM RELATIF** | | |
| PRELFP | Pronom relatif féminin pluriel | Qui |
| PRELFS | Pronom relatif féminin singulier | Qui |
| PRELMP | Pronom relatif masculin pluriel | Qui |
| PRELMS | Pronom relatif masculin singulier | Qui |
| **PREPOSITION** | | |
| PREP | Préposition | Chez |
| PREPADE | Préposition de | De |
| PREPAU | Préposition au | Au |
| PREPAUX | Préposition aux | Aux |
| PREPDES | Préposition des | Des |
| PREPDU | Préposition du | Du |
| **VERBE** | | |
| V1P | Verbe 1er pers du pluriel | Perdrions |
| V1S | Verbe 1er pers du singulier | Pense |
| V2P | Verbe 2ème pers du pluriel | Ayez |
| V2S | Verbe 2ème pers du singulier | Verras |
| V3P | Verbe 3ème pers du pluriel | Opposent |
| V3S | Verbe 3ème pers du singulier | Pourrait |
| VA1P | Verbe avoir 1er pers du pluriel | Avons |
| VA1S | Verbe avoir 1er pers du singulier | Ai |
| VA2P | Verbe avoir 2ème pers du pluriel | Aurez |
| VA2S | Verbe avoir 2ème pers du singulier | As |
| VA3P | Verbe avoir 3ème pers du pluriel | Ont |
| VA3S | Verbe avoir 3ème pers du singulier | A |
| VAINF | Verbe avoir infinitif | Avoir |
| VE1P | Verbe être 1er pers du pluriel | Sommes |
| VE1S | Verbe être 1er pers du singulier | Suis |
| VE2P | Verbe être 2ème pers du pluriel | Etes |
| VE2S | Verbe être 2ème pers du singulier | Es |
| VE3P | Verbe être 3ème pers du pluriel | Sont |
| VE3S | Verbe être 3ème pers du singulier | Est |
| VEINF | Verbe être infinitif | Être |
| VINF | Verbe infinitif | Devenir |
| **PARTICIPE PASSE** | | |
| VPPFP | Verbe participe passé féminin pluriel | Interrogées |
| VPPFS | Verbe participe passé féminin singulier | Préconisée |
| VPPMP | Verbe participe passé masculin pluriel | Publiés |
| VPPMS | Verbe participe passé masculin singulier | Permis |
| **PARTICIPE PRESENT** | | |
| VPPRE | Verbe participe présent | Etant |

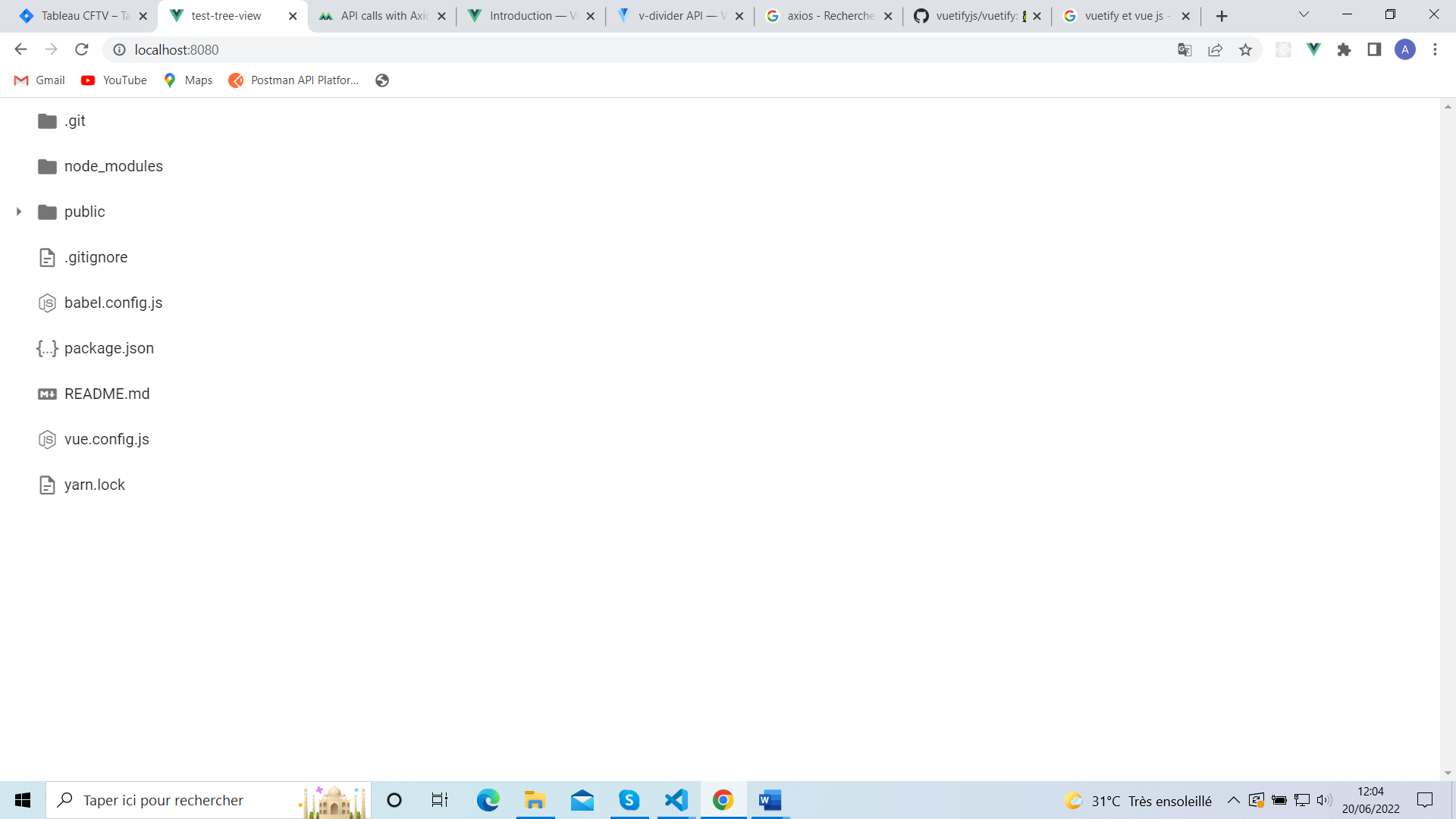
|  |  |  |
| --- | --- | --- |
| **CATEGORIE** | **SIGNIFICATION** | **EXEMPLE** |
| **NOM DE FAMILLE** | | |
| XFAM | Nom de famille | Mitterrand |
| **PAYS** | | |
| XPAYFP | Pays féminin pluriel | Landes |
| XPAYFS | Pays féminin singulier | Syrie |
| XPAYMP | Pays masculin pluriel | Elysées |
| XPAYMS | Pays masculin singulier | Dauphiné |
| **PRENOM** | | |
| XPREF | Prénom féminin | Ruth |
| XPREM | Prénom Masculin | François |
| **SIGLE** | | |
| XSOC | Sigle | PS |
| **VILLE** | | |
| XVILLE | Ville | Paris |

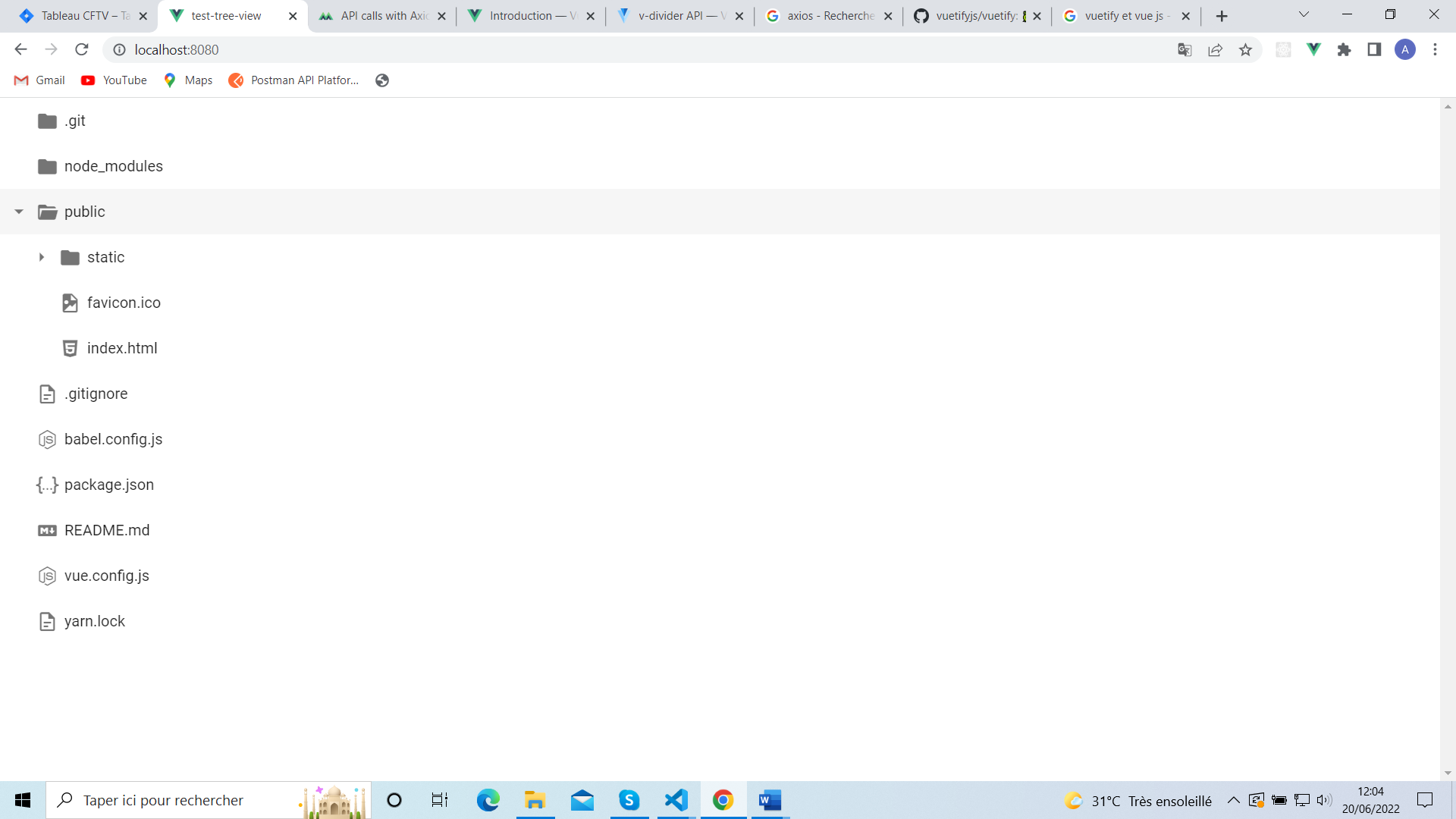
**NER :**

|  |  |  |
| --- | --- | --- |
| **CATEGORIE** | **DEFINITION** | **EXEMPLES** |
| ANAT | Anatomy/ Anatomie | *Système nerveux central, leucocytes, tissu sériques, hépatique rénale, globules blancs* |
| DISO-SYMP | Disorders-Smptoms/ Symptômes | *Aggravation de la maladie, maux de tête aggravée, complications, crampes, douleur, endormissement* |
| DISO-DISE | Disorders-Disease / Maladies | *Cancer, spondilartrite enkylosante, embolie pulmonaire, cancer de la prostate stade 4* |
| PROC | Procedure / Procédure | *Bronchotomie, injection intramusculaire, imagerie par résonance magnétique, IRM, perfusion, traitement, prise en charge, administration de médicaments par voie intrarachidienne, examens, administration* |
| CHEM-DRUG | Chemicals Drugs/ Médicaments | *Doliprane, Lévothyrox* |
| CHEM-MOLE | Chemicals Molecule/ Molécule chimique | *Paracétamol, Lévothyroxine* |
| DOSAGE | Dosage | *1000, 1,00 mg, 10 000 unités, 600 UI* |
| DURATION | Durée | *1 jour, 1 semaine* |
| FORM | Formes médicamenteuse | *Solution buvable, sol buv, comprimé, cpr* |
| POSOLOGY | Posologie | *x1/j, 4 cps/j matin, midi et soir* |
| AGE | Age | *63 ans* |
| WEIGHT | Poids | *43 Kg* |
| OUTCOME | Treatment outcome/ Résultat du traitement | *Guéri, mort, abandonné, échoué* |
| DISEASE | Maladie | *Embolie pulmonaire, cancer de la prostate stade 4* |
| NAME | Nom | *Labrak* |
| FIRSTNAME | Prénom | *Yanis* |
| ADDRESS | Adresse | *1 Avenue Marius Pinatel* |
| CITY | Ville | *Plan-de-Cuques* |
| POSTAL | Code Postal | *13380* |
| HOSPITAL | Nom de l’hôpital | *Centre Hospitalier Edouard Toulouse, Clinique Saint-Barnabé, Ramsay Santé* |
| PHONE | Téléphone | *0607080910* |
| BIRTHDAY | Date de naissance | *01/07/1999* |

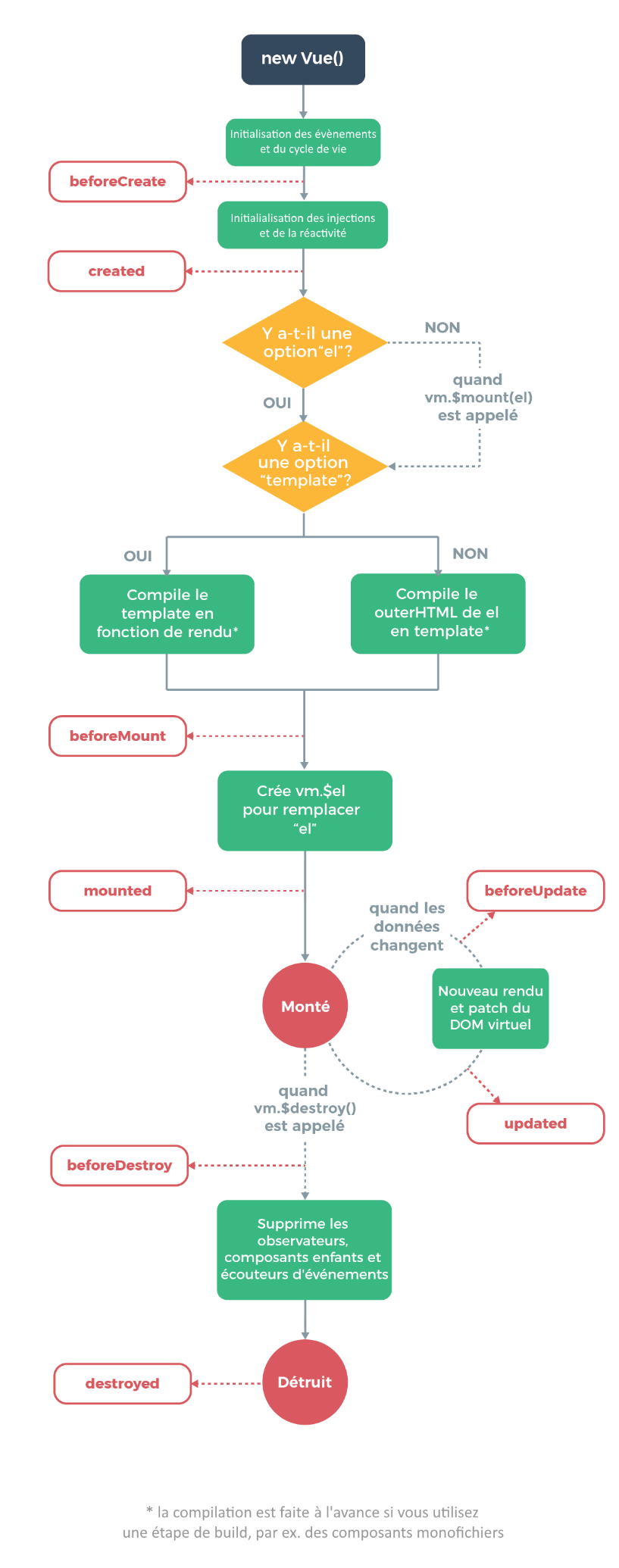
**B – Projet n°2**

**Annexe 1 :** Représentation visuelle du TreeView seul

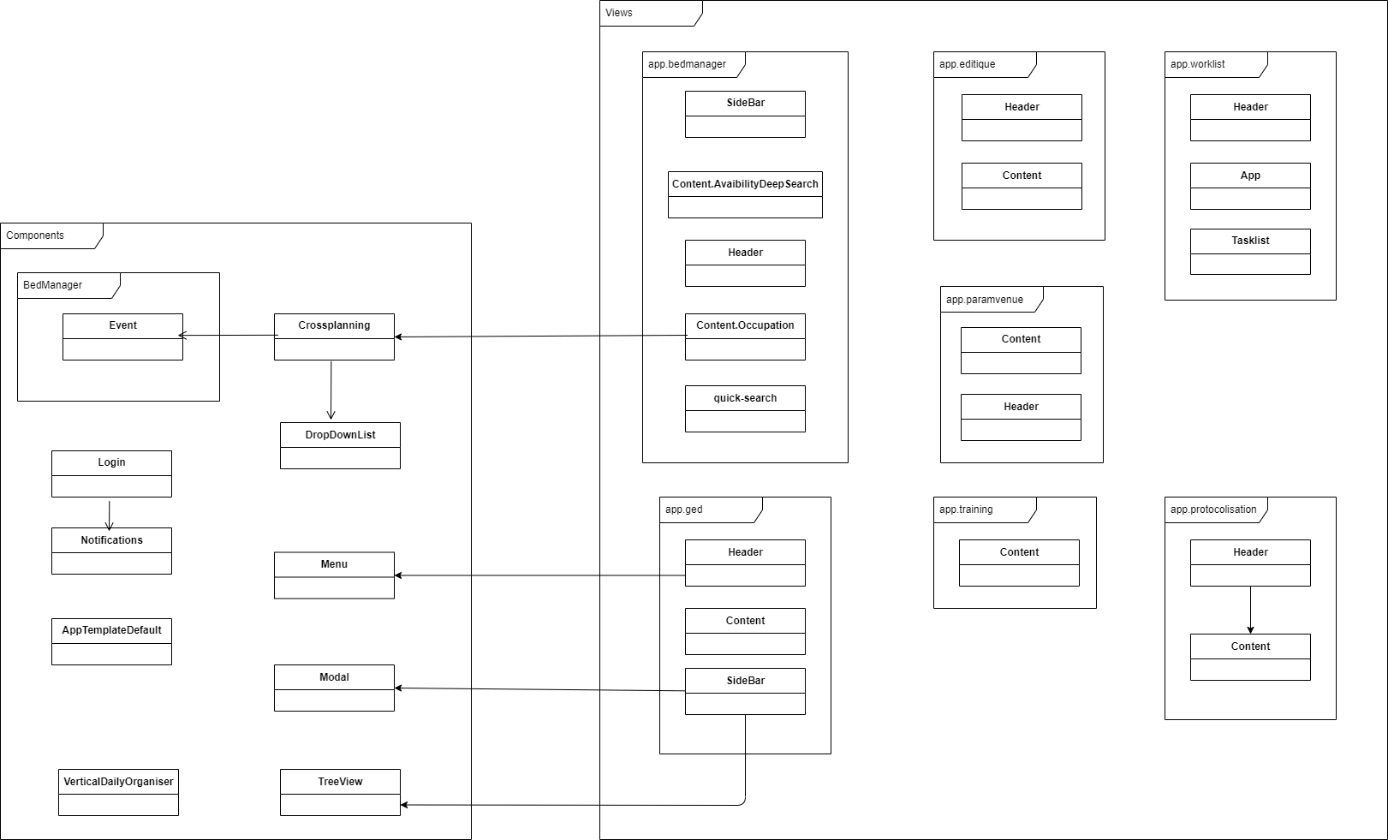




**Annexe 2 :** Schéma du cycle de vie d’un composant

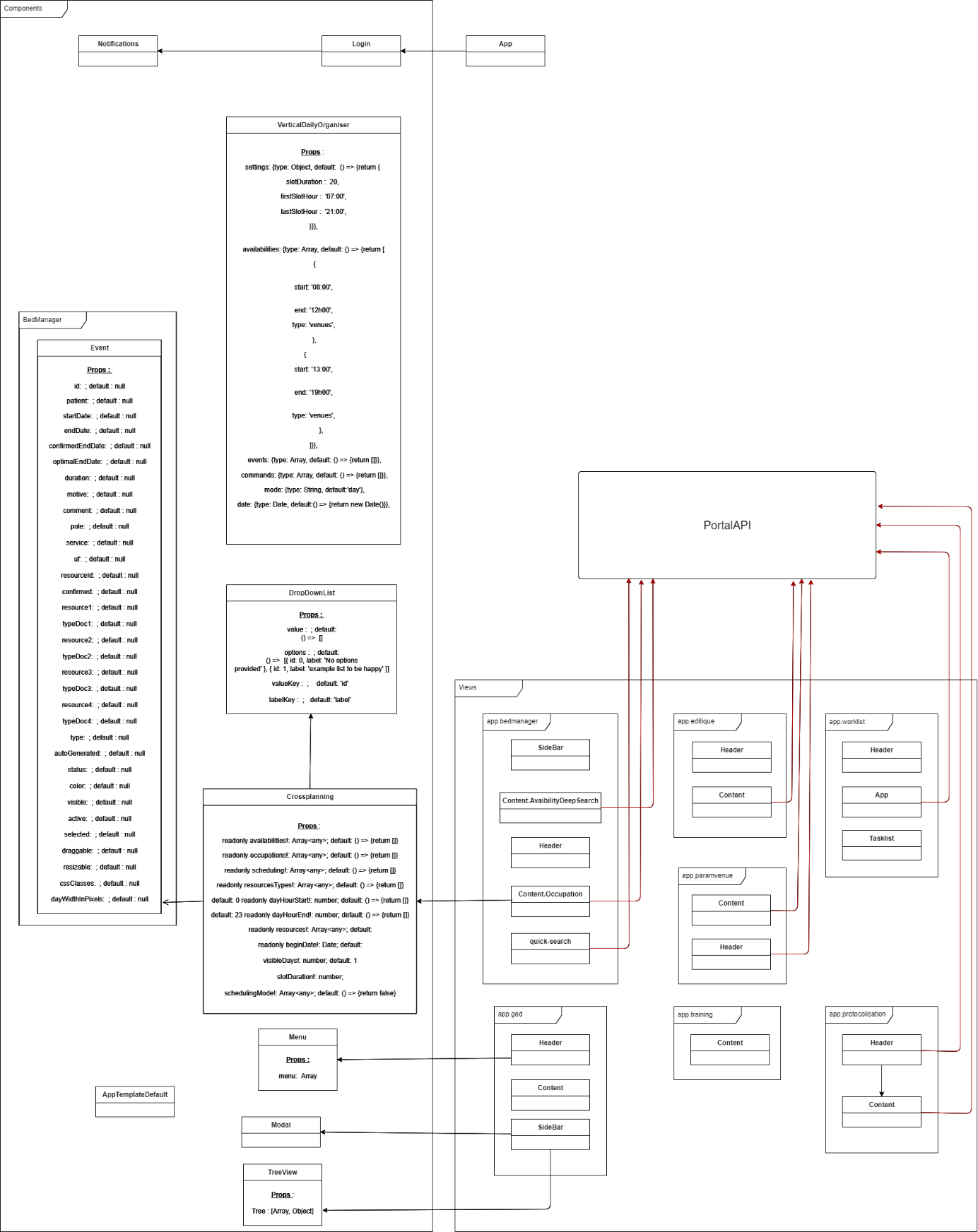


**Annexe 3 :** Structure des composants sans les props



*Source : /docs/composantsSansProps.png*

**Annexe 4 :** Structure des composants avec les props et les appels PortalAPI



*Source : /docs/composants.drawio*

*/docs/composants.png*

**Annexe 5 :** Exemple de format de données utilisées pour l’affichage du TreeView

{

"items": [

  {

    "name": ".git"

  },

  {

    "name": "node\_modules"

  },

  {

    "name": "public",

    "children": [

      {

        "name": "static",

        "children": [

          {

            "name": "logo.png",

            "file": "png"

          }

        ]

      },

      {

        "name": "index.html",

        "file": "html"

      }

    ]

  },

  {

    "name": ".gitignore",

    "file": "txt"

  },

  {

    "name": "babel.config.js",

    "file": "js"

  },

  {

    "name": "package.json",

    "file": "json"

  },

  {

    "name": "README.md",

    "file": "md"

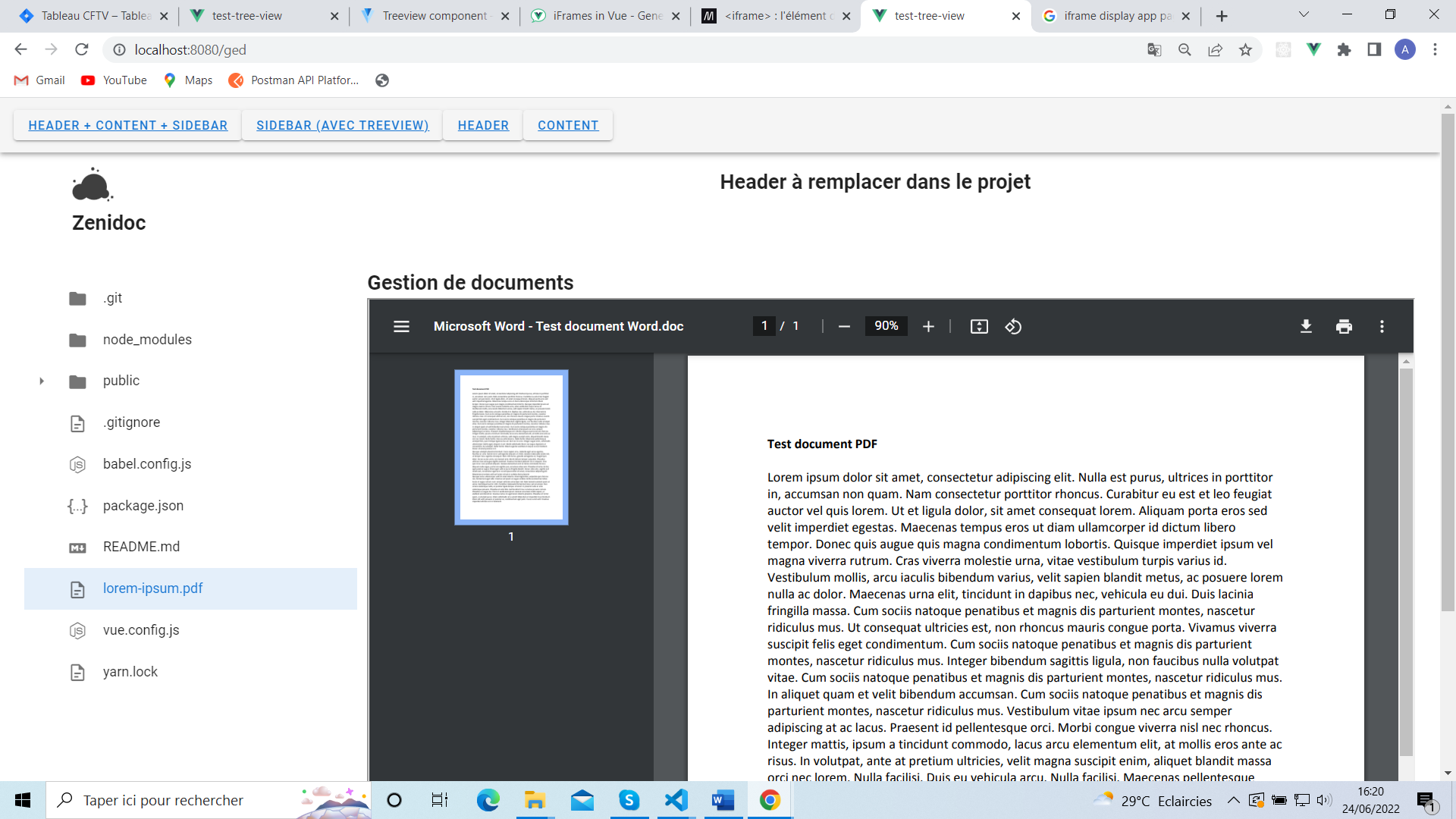
  },

]

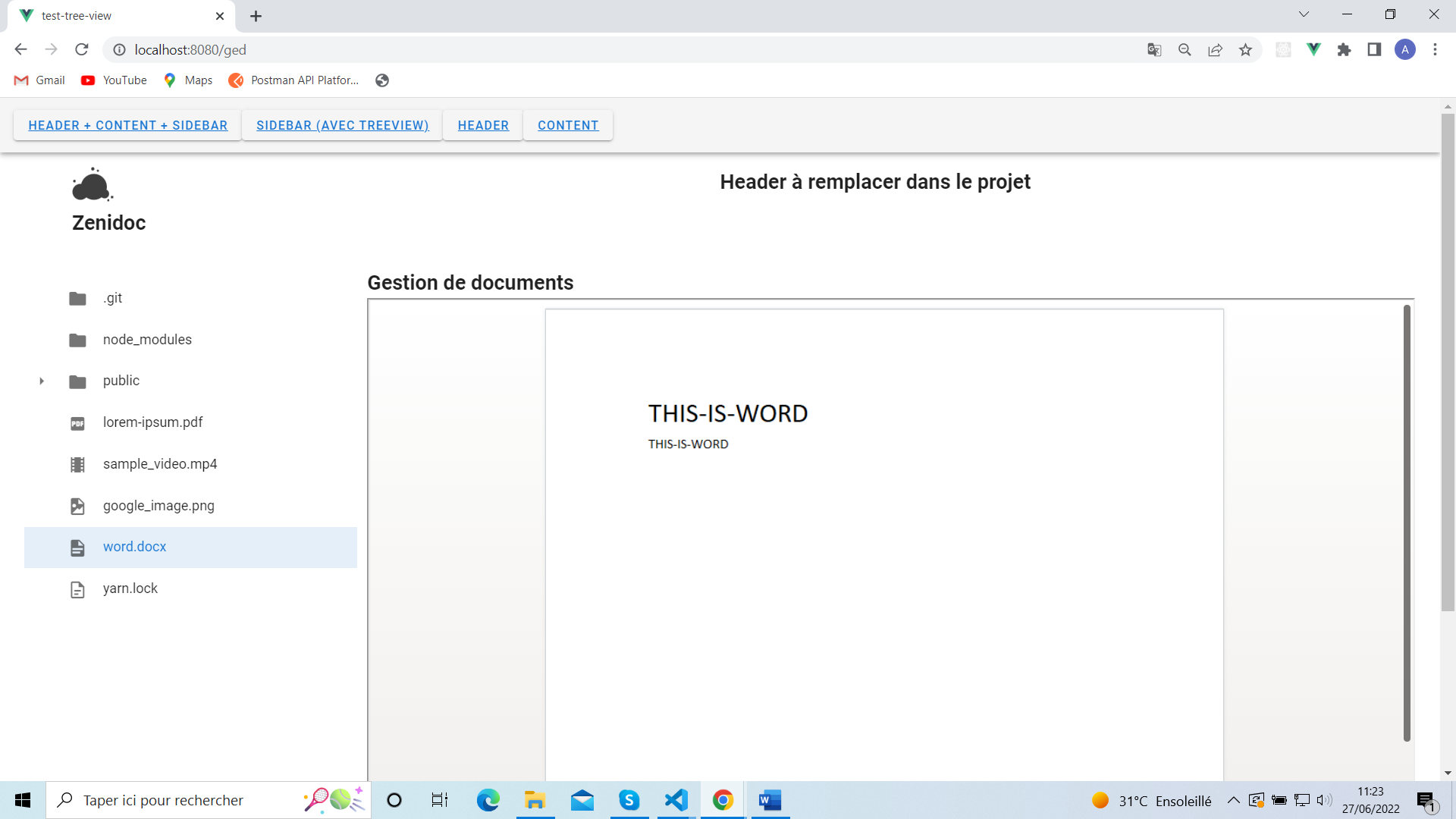
}

*Source : /db.js*

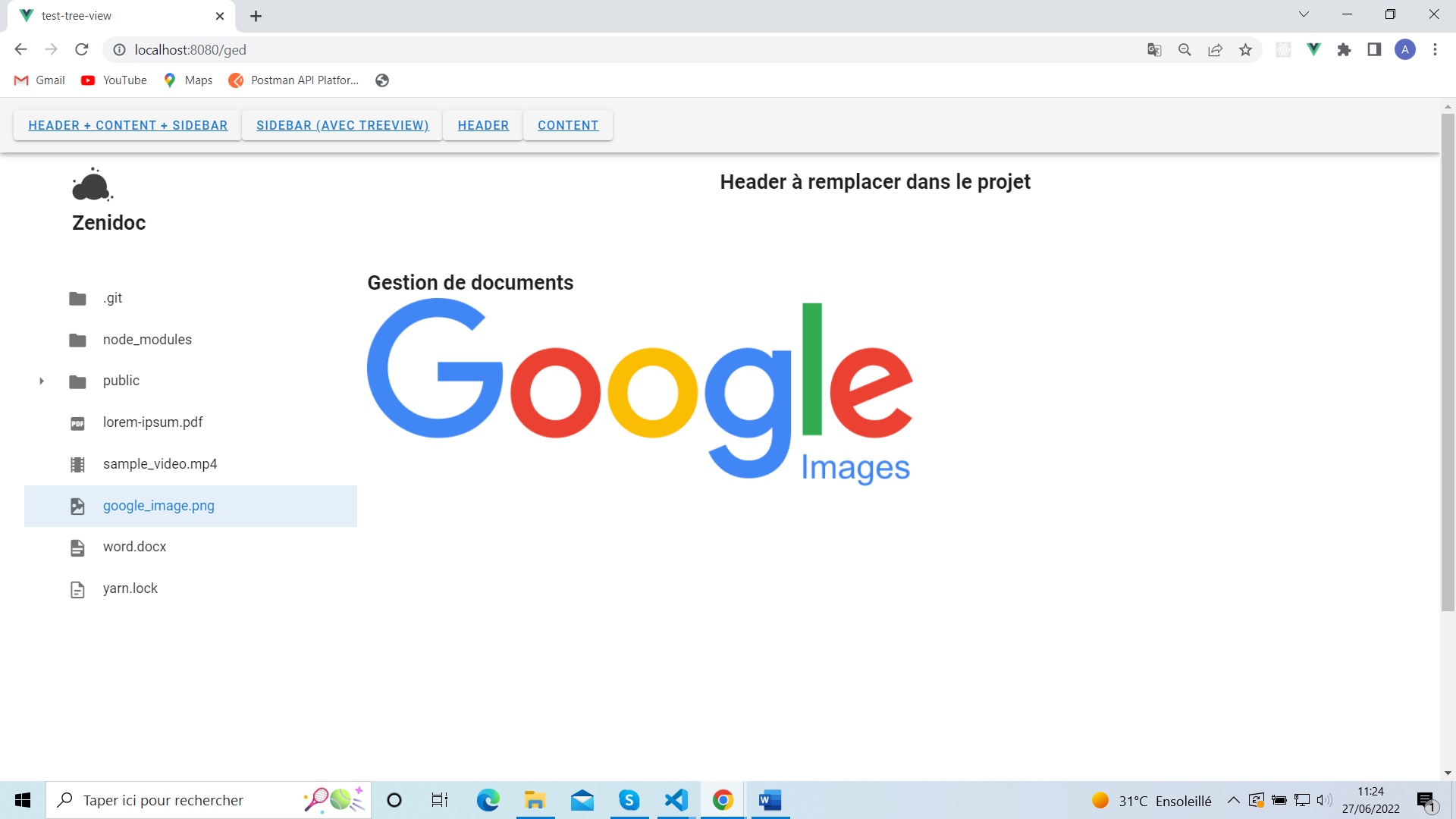
**Annexe 6 :** Exemples de divers types de fichiers affichés



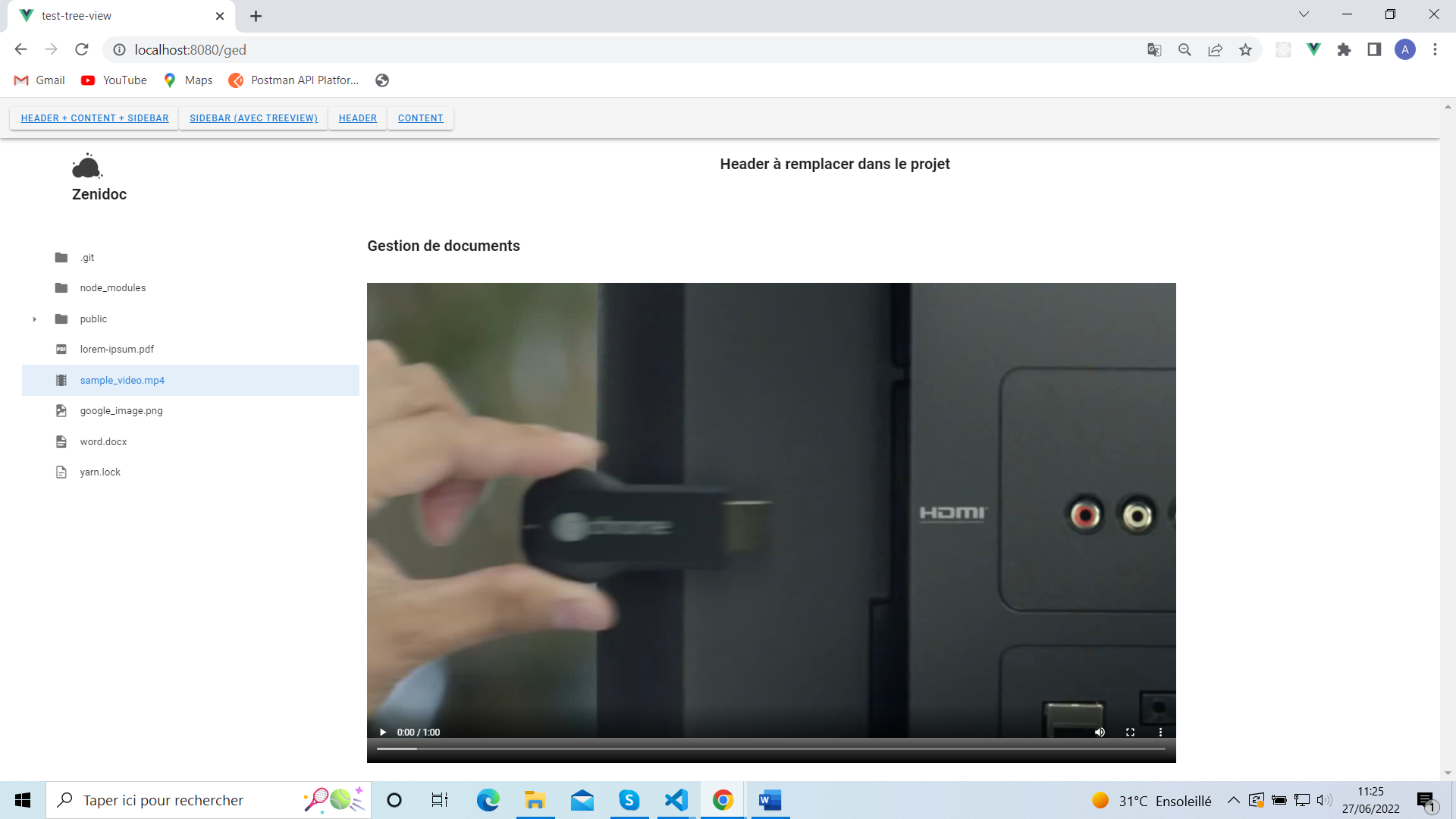
*Fichier .pdf*



*Fichier .docx*



*Fichier .png*



*Fichier .mp4*

**III – Bibliographie**

**A – Projet n°1**

**API Fetch :**

* <https://developer.mozilla.org/fr/docs/Web/API/Fetch_API>

**JavaScript :**

* <https://developer.mozilla.org/fr/docs/Web/JavaScript>
* <https://www.w3schools.com/js/default.asp>
* <https://www.toutjavascript.com/main/index.php3>

**B – Projet n°2**

**Documentation Vue.js 2 :**

* <https://v2.vuejs.org/v2/guide/>
* <https://openclassrooms.com/fr/courses/6390311-creez-une-application-web-avec-vue-js>
* <https://www.vuemastery.com/>

**Documentation sur l’utilisation du composant TreeView et de Vuetify :**

* <https://vuetifyjs.com/en/components/treeview/>
* <https://vuetifyjs.com/en/api/v-treeview/>
* <https://www.vuemastery.com/courses/beautify-with-vuetify/getting-started-with-vuetify>

**Documentation sur l’utilisation d’Axios et de json-server :**

* <https://axios-http.com/docs/intro>
* <https://github.com/typicode/json-server>
* <https://www.vuemastery.com/courses/real-world-vue-js/API-calls-with-Axios>

**Différentes icônes pour la représentation des fichiers :**

* <https://materialdesignicons.com/>

**Documentation sur Vue Class Component :**

* <https://class-component.vuejs.org/>

**Documentation sur Vue-Router :**

* <https://router.vuejs.org/guide/>
* <https://www.vuemastery.com/courses/real-world-vue-js/vue-router>

**Afficher des documents de la suite Microsoft Office dans un iframe :**

* <https://www.microsoft.com/en-us/microsoft-365/blog/2013/04/10/office-web-viewer-view-office-documents-in-a-browser/>