LYCÉE ARISTIDE BERGÈS
SIO

Utiliser...

Visual Studio Code

Rédigé par

David ROUMANET
Professeur BTS SIO

Changement

Date	Révision
20 juillet 2021	Première publication du document

Sommaire

A Présentation

A.1 Origine

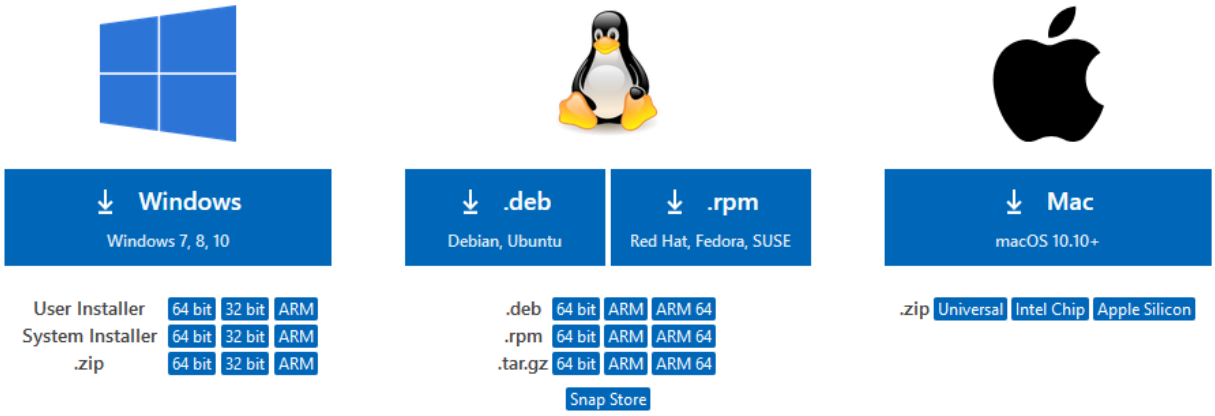
Visual Studio Code est un éditeur de texte, au même titre que Notepad++, Atom, Bracket, Sublim Text…

Il est développé par Microsoft, à l'aide du framework "Electron" (HTML, CSS, JavaScript) mais propose des performances très satisfaisantes.

C'est l'un des meilleurs outils pour développer en TypeScript (Angular2), il propose de nombreuses extensions et une ergonomie très correcte.

A.2 Téléchargement et installation

Visual Studio Code est disponible sur plusieurs environnements : Windows mais aussi Linux et MacOS. Il est disponible sur le site https://code.visualstudio.com/



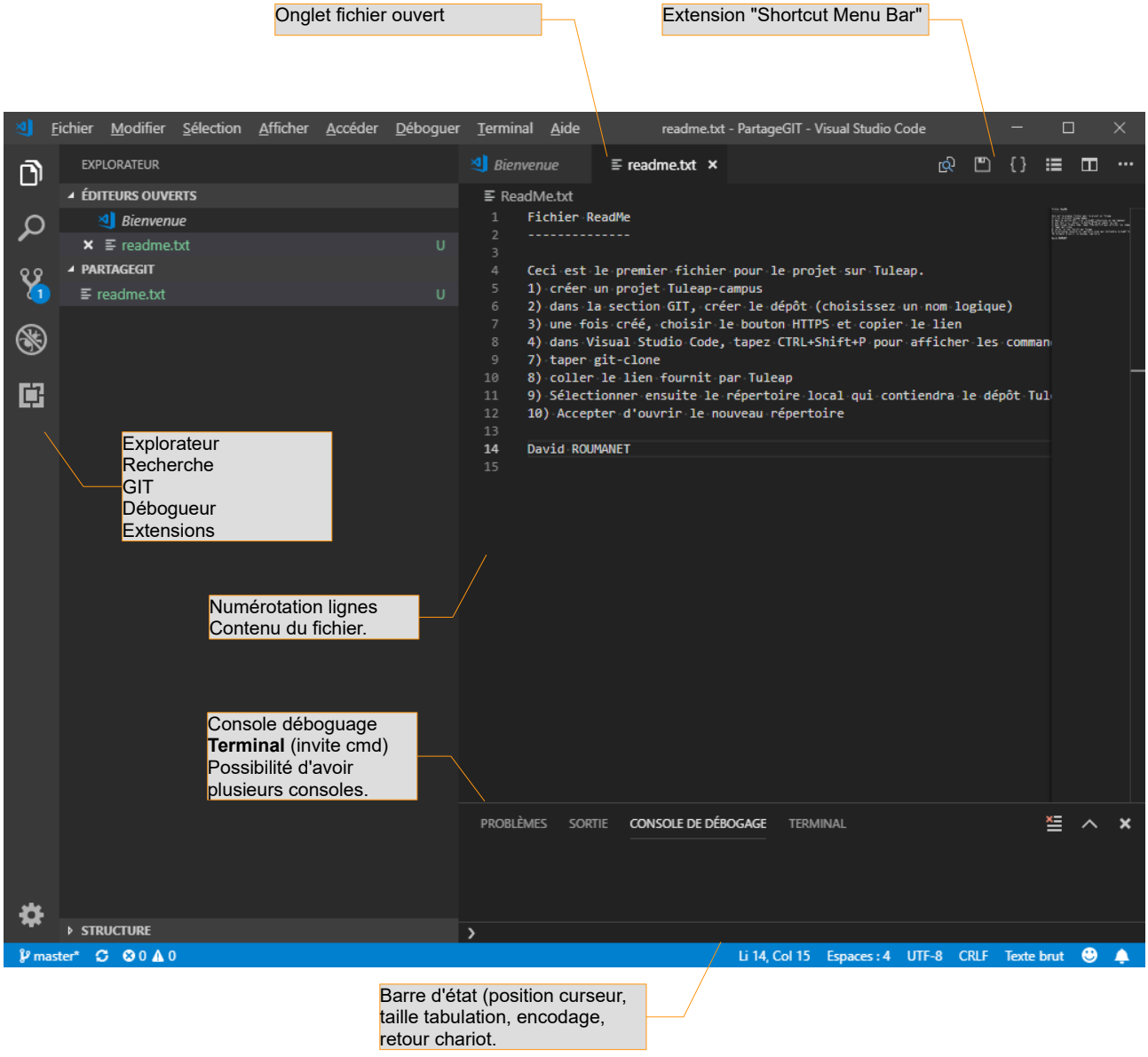
La version "User Setup", l'installation par défaut place les fichiers dans **C:\users\{username}\AppData\Local\Programs\Microsoft VS Code** et s'exécute sans aucun droit d'administration.

La version "System Setup" demande l'accès aux privilèges administrateurs.

Sur un poste partagé, il est préférable d'utiliser cette version, pour n'installer qu'une fois l'application (dans la version "User Setup" chaque utilisateur installe sa propre version).

A.3 Interface

L'interface de Visual Studio Code est plus légère que Notepad++ pour laisser plus de place aux codes affichés : ainsi, les icônes sont considérées superflues et le découpage de la fenêtre principale est très sobre.



A.4 Avantages de VSCode

Visual Studio Code est relativement léger à l'installation (environ 80Mo téléchargés) mais permet de télécharger des extensions pour chaque langage. Par défaut, VSCode permet la complétion de code (fonctionnalité "IntelliSense") : cela va au-delà de compléter les instructions, car VSCode propose des aides sur les paramètres de méthodes, sur la terminaison des blocs, etc.

B Raccourcis claviers

VSCode s'utilise principalement avec des raccourcis clavier et des commandes à écrire. Il y a très peu d'icônes.

B.1 Fichiers

Comme dans la plupart des logiciels, les commandes suivantes fonctionnent :

- CTRL+O pour ouvrir un fichier
- CTRL+S pour sauvegarder l'onglet courant (le fichier en cours)
- CTRL+N pour créer un nouveau fichier
- CTRL+K puis S permet de sauvegarder tous les fichiers

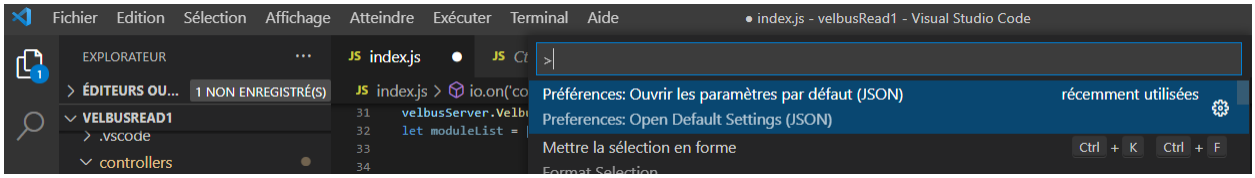
B.2 Édition

On trouve ici les commandes pour rechercher, remplacer mais aussi formater un code :

- CTRL+F pour chercher
- CTRL+H pour remplacer
- SHIFT+ALT+F pour formater le fichier
- CTRL+A pour sélectionner tout le document
- CTRL+SHIT+L pour sélectionner toutes les occurrences d'un mot sélectionné et les éditer simultanément
- F2 renommage d'un symbole (d'un attribut, d'une variable) dans tout le document

B.3 Autres

Il s'agit d'un raccourci très utilisé dans Visual Studio Code, pour accéder aux fonctionnalités évoluées par ligne de commande : CTRL+SHIFT+P qui affiche la palette de commandes.



L'autre raccourci utile est CTRL+K puis CTRL+S qui affiche tous les raccourcis ('S' comme Shortcut).

C Extensions intéressantes

Comme tout développeur, il existe quelques extensions à ajouter et d'autres fonctionnalités qui sont incluses par défaut. L'icône suivante permet d'installer, vérifier et activer/désactiver les extensions.

C.1 GIT (inclus)

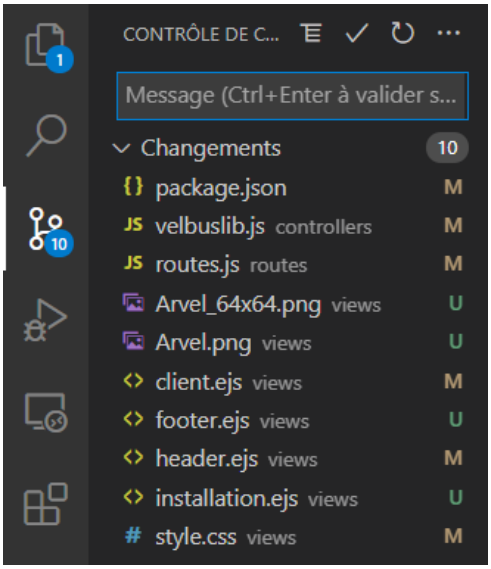
C.1.1 Gestion locale

La gestion GIT est incluse par défaut dans la barre verticale à gauche.

En cliquant sur le symbole de "contrôle de code source" (raccourci CTRL+SHIFT+G comme GIT) VSCode affiche la liste des fichiers qui ont été modifiés (M) ou non suivis (U comme Unfollowed), etc.

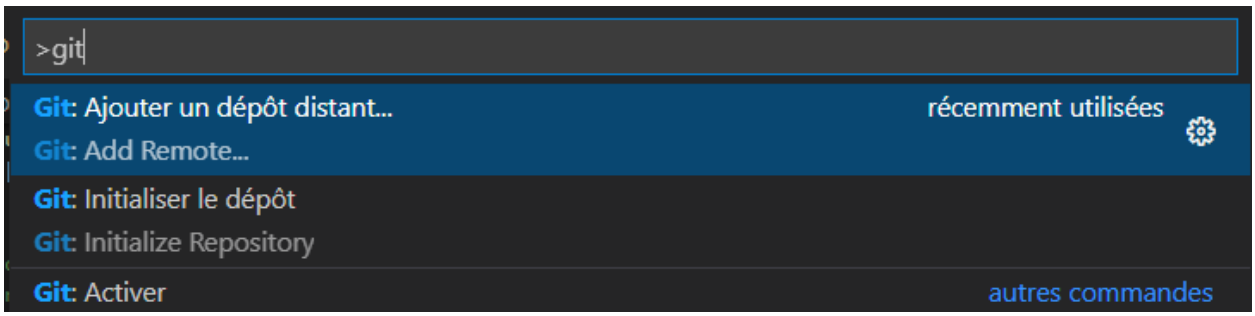
En passant la souris sur les fichiers, il est possible de les ajouter au prochain "commit" en cliquant sur le symbole +.

Enfin, en haut de l'interface, on peut rédiger un commentaire et valider sur la coche (ou bien CTRL+Enter).

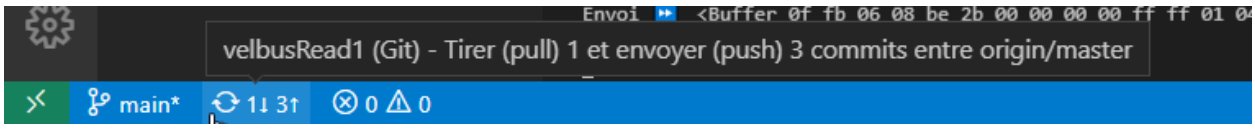


C.1.2 Gestion distante (remote)

Une configuration est nécessaire pour indiquer où se trouve le projet (serveur). Cette configuration est accessible par le raccourci CTRL+SHIFT+P et le mot clé "git"



Par la suite, la mise à jour d'un dépôt distant se fait en cliquant dans la barre d'état, sur le symbole représentant un cycle.



C.2 JSDoc (inclus)

Il est utile de générer la documentation d'un code à partir du code lui-même, c'est ce que permettent JavaDOC, DOXygen et d'autres outils. VSCode sait pré-générer la documentation d'une méthode, en utilisant la suite de symbole /** sur la ligne juste avant cette méthode puis d'appuyer sur la touche [Entrée].

Il ne reste plus qu'à préciser les informations manquantes que VSCode ne peut déterminer :

- La première ligne décrit la fonction
- Les lignes @param décrivent le type (string, number, etc) des paramètres
- La ligne @returns explique ce que la fonction renvoie.

```
142   /**
143    *
144    * @param {*} A
145    * @param {*} B
146    * @returns
147    */
148   function Max(A, B) {
149       if (A > B) return A
150       return B
151   }
152
```

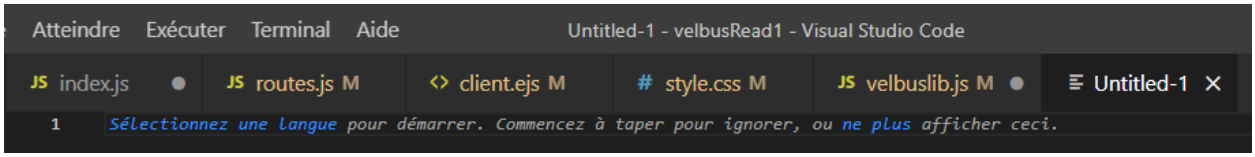
Si vous voulez plus d'information sur les paramètres utilisés dans JSDoc[1], il existe un support de cours portant sur la qualité du code.

L'important est de savoir que VSCode permet d'ajouter facilement ce genre de code.

C.3 Emmet (inclus)

De la même manière que JSDoc, VSCode permet de générer rapidement du code HTML en utilisant les abbréviations Emmet[2].

Dans un document vide, choisir le format de fichier HTML (clic sur le lien en bleu "Sélectionnez une langue" ou bien sauvegardez le document avec l'extension HTML).



Par la suite, utilisez les abréviations Emmet comme HTML:5 puis appuyez sur la touche [Entrée]

VSCode génère les instructions HTML pour un fichier au format HTML5.

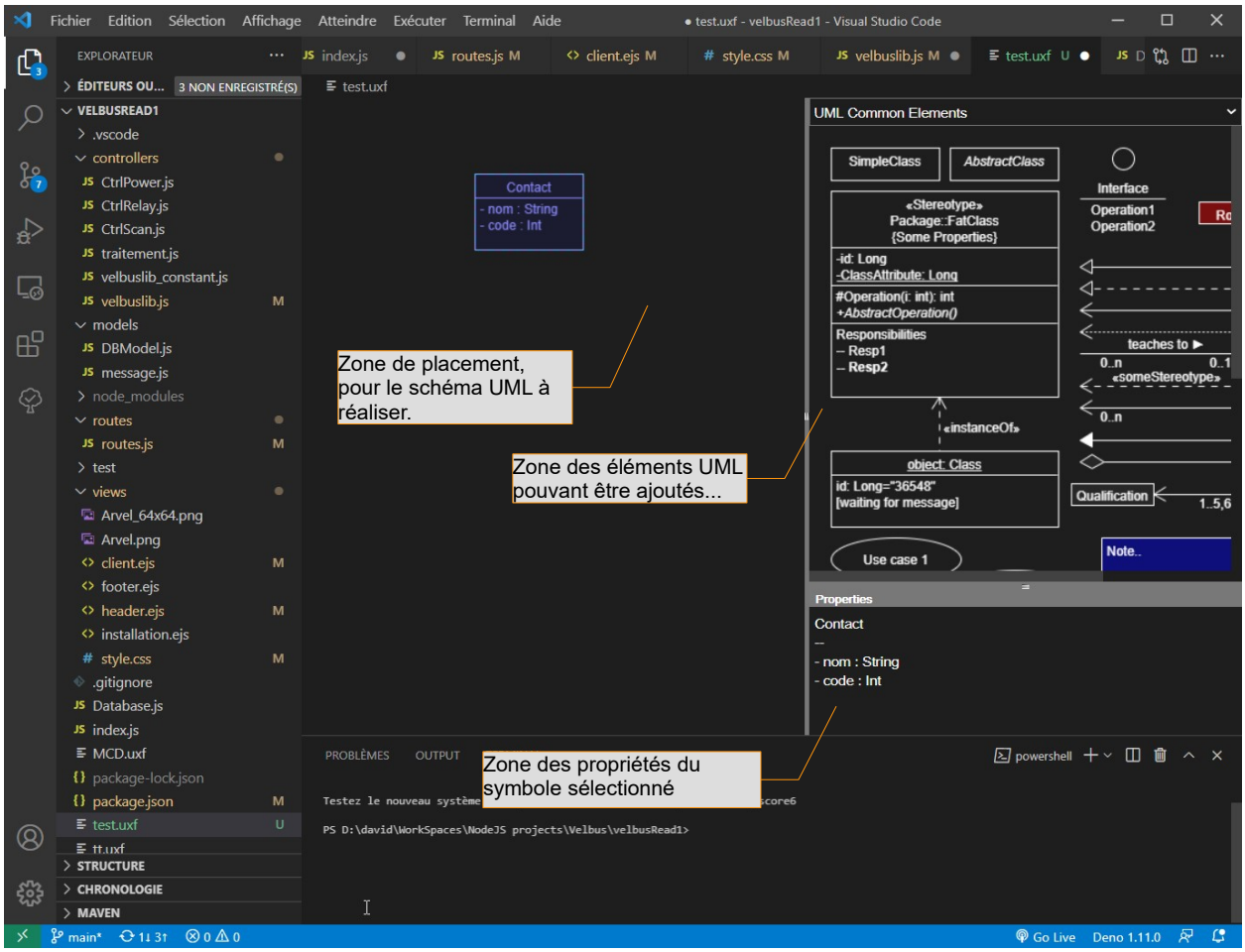
Vous trouverez aussi des informations complémentaires sur Emmet dans les cours de première année, premier semestre.

1 JSDoc : https://jsdoc.app/
2 Emmet : https://docs.emmet.io/abbreviations/

C.4 UMLet (non-inclus)

UMLet est une extension permettant de réaliser des schémas UML sur des fichiers avec l'extension **uxf**.

Son utilisation est très simple : il suffit de créer un nouveau document avec l'extension .uxf, de le sauvegarder puis de le réouvrir et un menu latéral apparaît avec des bibliothèques de symboles UML.



Pour rappel, un diagramme UML est souvent utilisé pour modéliser une base de données (Modèle Conceptuel de Données) ou bien un code et ses classes (diagramme de classes). On ajoute aussi les diagrammes d'utilisations pour expliquer le comportement de l'application. Tout cela fait l'objet d'un cours complet en deuxième année de SIO.

Cette extension permet de ne pas sortir de l'environnement de développement pour modifier ses schémas UMLet[3].

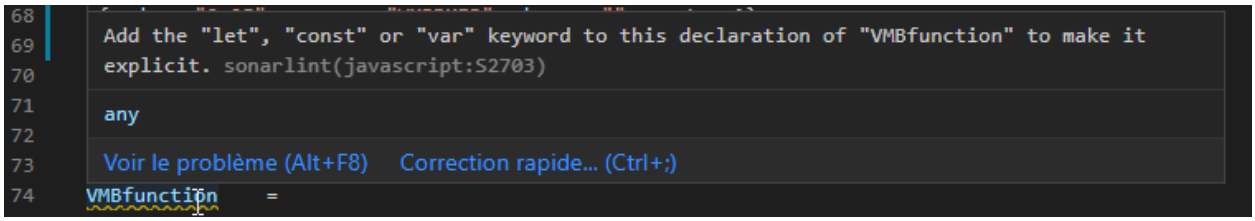
Les extensions pour VSCode sont accessibles sur https://marketplace.visualstudio.com/VSCode !

3 UMLet en ligne : http://www.umletino.com/umletino.html ou en version autonome : https://www.umlet.com/

C.5 SonarLint (non inclus)

SonarQube[4] est un outil de vérification de la qualité du code : il référence de nombreuses règles d'écritures qui peuvent entraîner des erreurs de fonctionnement d'une application.

SonarLint est une extension qui permet de détecter les lignes de vos codes qui correspondent à ces règles à risque. SonarLint indique ainsi le risque et généralement le moyen de l'éviter.



Il faut cependant comprendre que SonarLint est une version légère de l'outil SonarQube, et permet d'avoir facilement des retours rapides au moment du développement.

À l'inverse, SonarQube est un outil puissant qui s'intègre dans un ALM (Application Lifecycle Management). On parle souvent de CI/CD pour Continuous Integration/Continuous Delivery.

SonarLint[5] est disponible pour d'autres IDE, comme Eclipse, IntelliJ, Clion, Visual Studio.

Les règles sont accessibles sur https://rules.sonarsource.com/javascript/ !

Respecter ces règles permet de participer à la lutte contre les failles de sécurité OWASP :

JavaScript coverage of OWASP TOP 10 2017

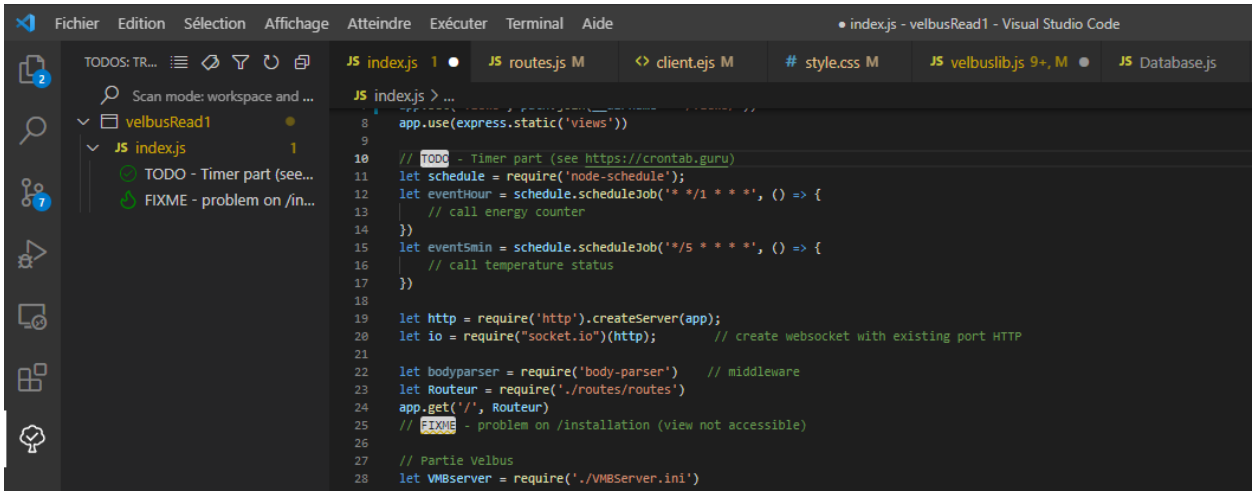
		Security Vulnerability	Security Hotspot
A1	Injection	✓	✓
A2	Broken Authentication	✓	✓
A3	Sensitive Data Exposure	✓	✓
A4	XML External Entities (XXE)	✓	—
A5	Broken Access control	✓	✓
A6	Security misconfigurations	✓	✓
A7	Cross Site Scripting (XSS)	✓	✓
A8	Insecure Deserialization	—	—
A9	Using Components with known vulnerabilities	✓	—
A10	Insufficient logging and monitoring	—	—

4 Site : https://www.sonarqube.org/
5 Site : https://www.sonarlint.org/

C.6 ToDo Tree (non inclus)

Une dernière extension qui n'est pas nécessaire mais pratique est de pouvoir repérer facilement les commentaires comportant les mots-clés **TODO** et **FIXME** : de nombreux développeurs placent ces mots-clés pour se rappeler où ils doivent corriger et terminer leur code.

Cette extension permet de modifier l'apparence de ces mots-clés mais aussi, d'ajouter une icône dans la barre latérale, pour retrouver toutes les positions de code contenant une tâche à faire.



Ce qui est intéressant avec cette extension, et de configurer l'extension pour avoir ses propres icônes et couleurs. Il faut en effet modifier le fichier de préférences de Visual Studio Code. Pour cela, CTRL+SHIFT+P puis choisir "Préférences : afficher les paramètres (en JSON).

Il faut ensuite ajouter les clés JSON suivantes (ici, ma configuration personnelle) :

```
"todo-tree.highlights.defaultHighlight": {
    "icon": "alert",
    "type": "text",
    "foreground": "white",
    "background": "#515151",
    "opacity": 50,
    "iconColour": "cyan"
},
"todo-tree.highlights.customHighlight": {
    "TODO": {
        "foreground": "chartreuse",
        "icon": "pencil",
        "iconColour": "chartreuse",
        "type": "text",
        "gutterIcon": true
    },
    "FIXME": {
        "foreground": "darkorange",
        "iconColour": "darkorange",
        "type": "whole-line",
        "gutterIcon": true
    }
```

Vous pouvez maintenant éditer les préférences en suivant la documentation .