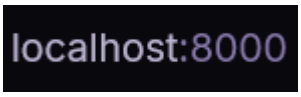
SAE 1.05

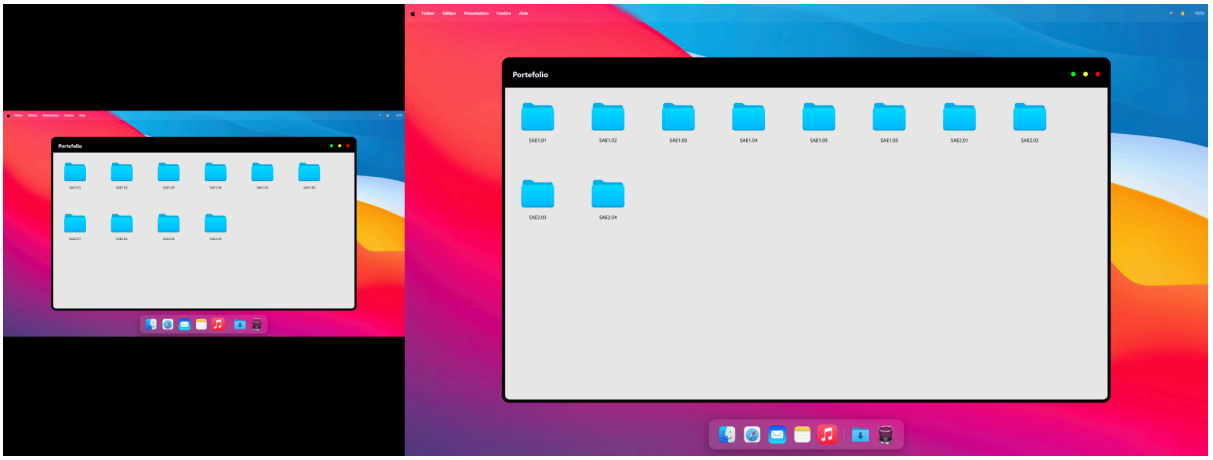
AC14.01 : Exploiter de manière autonome un environnement de développement efficace et productif

Pour s'assurer que les images, favicon et l'ensemble du site web soit fonctionnel lorsqu'il sera mis en ligne, j'ai utilisé le serveur "Python". J'ai un fichier en .bat permettant de mettre en ligne sur le PC local, ainsi, avec localhost:8000 ou encore mon adresse IP suivi du numéro de port, je pouvais voir en temps réel lorsque je chargeais un élément de la page.



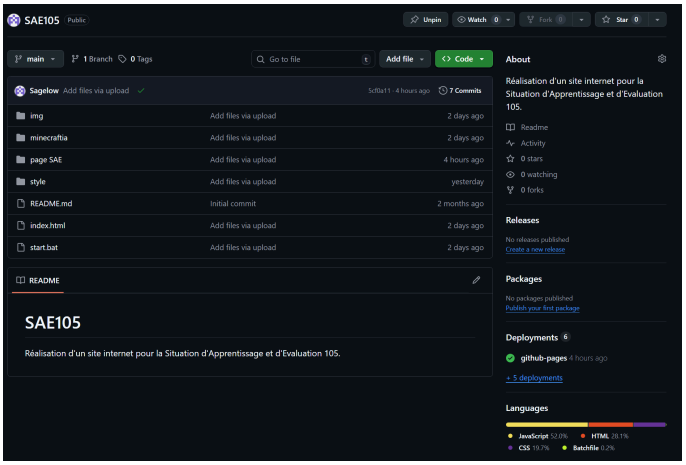

```
C:\Ecole\SAE105-main>python -m http.server
Serving HTTP on :: port 8000 (http://[::]:8000/) ...
::1 - - [07/Jan/2026 12:35:51] "GET / HTTP/1.1" 200 -
::1 - - [07/Jan/2026 12:35:51] "GET /style/style.css HTTP/1.1" 200 -
::1 - - [07/Jan/2026 12:35:51] "GET /img/LFA.jpg HTTP/1.1" 200 -
::1 - - [07/Jan/2026 12:35:51] "GET /img/JDA.jpg HTTP/1.1" 200 -
::1 - - [07/Jan/2026 12:35:51] "GET /img/LLA.jpg HTTP/1.1" 200 -
::1 - - [07/Jan/2026 12:35:51] "GET /img/IUT.jpg HTTP/1.1" 200 -
::1 - - [07/Jan/2026 12:35:51] "GET /img/bgbtn.png HTTP/1.1" 200 -
::1 - - [07/Jan/2026 12:35:51] "GET /minecraftia/Minecraftia-Regular.ttf HTTP/1.1" 200 -
::1 - - [07/Jan/2026 12:35:52] code 404, message File not found
```

J'ai également deux écrans de tailles différentes (1920x1080 et 3840x2160), ce qui me permettait de coder d'un côté tout en aillant le résultat de l'autre.

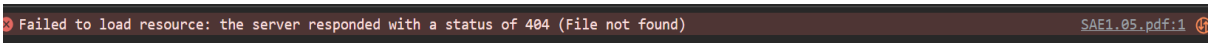


Je pouvais alors voir si mon code était responsive d'un écran à un autre.

J’utilisais également Github pour transférer les fichiers lorsque je travaillais à domicile ou à l’université.



J’ai aussi utilisé des outils de débogage comme le console.log pour vérifier les informations dans certaines variables et voir lorsque j’avais des problèmes.



J’ai aussi privilégié Visual Studio code plutôt que Sublime Text dont je me suis familiarisé pour apprendre une nouvelle plateforme. Il avait des avantages comme ouvrir et fermer automatiquement la balise lorsqu’on tapait le nom en question. De plus, on l’utilisait lors des ressouces ce qui m’a poussé à l’utiliser définitivement pour la SAE 1.05

AC14.02 : Produire des pages Web fluides incluant un balisage sémantique efficace et des interactions simples

J’ai essayé de diversifier au maximum les balises possibles tout en restant cohérent avec mon site. Par exemple, j’ai utilisé dans une grande majorité la balise `<div></div>` pour les textes mais j’ai aussi utilisé la balise `<p></p>` .
J’ajoutais presque automatiquement des classes aux balises pour pouvoir méticuleusement placer chaque objet où je le voulais sur le site.

J’ai utilisé des raccourcis clavier comme **CTRL+F5** pour rafraîchir une page lorsque le cache était saturé. Il arrivait que, malgré le rafraîchissement de la page, cela ne s'actualise pas. J’ai appris qu’il pouvait y avoir un problème de cache et ce raccourci pouvait résoudre ce problème.
Un autre raccourci qui m’a été utile est : **ALT+SHIFT+F**. Je le faisais automatiquement pour avoir plus de clarté et de lisibilité dans le code. Cela servait aussi à m’assurer que mon indentation était correcte.
Généralement, je retourne toujours à la ligne pour les balises sauf lorsqu’il y a du texte et que ce dernier n’est pas trop long.

Exemple :

```html
<a href="#" id="goSecond">
    <h1>Welcome</h1>
</a>
```

```html
<div class="block">
    <div class="txt">Jeanne d'Arc est une école avec laquelle j'ai une affinité pariculière. J'ai fait
        ma maternelle et mon école primaire là-bas. Je suis revenu en 3e pour réaliser mon brevet des
        collèges puis j'ai quitté à la fin de la seconde.</div>
    <img src="img/JDA.jpg" alt="école Jeanne d'Arc">
</div>
```

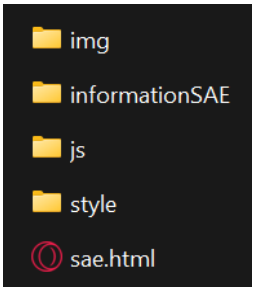
Puisque j'utilise Github, je mets mes commentaires et mes classes pratiquement toujours en anglais. Cela permet, si un jour quelqu'un de non français tombe sur mon code, puisse le comprendre. J'en mets uniquement lorsqu'il y a lieu d'être pour ne pas plonger le lecteur dans un code rempli de vert avec plus de commentaire que de ligne de code. J'essaye au maximum de structurer mes fichiers pour que mes commentaires soient des sections où se trouve l'information.

Exemple :

```css
/*****************************************************************/
/* Button style + Centered (https://codepen.io/joexmdq/pen/EOMLzg) */
/*****************************************************************/

.center {
    display: flex;
    justify-content: center;
    align-items: center;
    padding-bottom: 64px;
}

.btn {
    display: flex;
    width: 20%;
    border: 4px solid #000;
    background: url("/img/bgbtn.png") center / cover;
    image-rendering: pixelated;
    text-shadow: 2px 2px #000A;
    font-size: 24px;
    align-items: center;
    justify-content: center;
    text-decoration: none;
    color: #f4f3f3;
    cursor: pointer;
    box-shadow: inset -4px -8px #0006, inset 4px 4px #FFF7
}

.plus {
    display: flex;
    width: 100%;
    height: 100%;
    padding: 2vh 0 1.3vh;
    box-sizing: border-box;
    align-items: center;
    justify-content: center;
    font-size: 24px;
    color: #f4f3f3;
    text-shadow: 2px 2px #000A;
    box-shadow: inset -4px -8px #0006, inset 4px 4px #FFF7;
}

/*************************/
/* Animation when mouse on */
/*************************/
```

Ce commentaire montre que ces 3 classes sont dédiées au style d'un bouton.

L'arborescence à été pensé pour avoir à chaque fois un index qui mène vers un autre .html mais en ayant son propre dossier et les sous-dossiers qui lui est dédié. Les noms des fichiers/dossiers ont été nommés judicieusement pour indiquer sa signification.



Pour l'expérience utilisateur (UX), j'ai structuré le site de manière claire et intuitive :

- Une page d'accueil avec un message de bienvenue et un lien vers les sections suivantes.
- Des sections séparées pour le parcours et les SAE pour que l'utilisateur puisse se repérer facilement.
- Des boutons et liens visibles et cohérents, avec un feedback visuel au survol (hover) pour indiquer l'interactivité.

Du côté de l'interface utilisateur (UI), j'ai tenté différents styles:

- Une typographie minecraftia et son bouton de référence
- Un design se rapprochant à l'interface MacOS et son design Liquid Glass
- Une description des SAE avec une page sobre, en se basant sur des couleurs personnels

J'ai diversifié car je n'arrivais pas à me décider quant au design du site.

AC14.01 : Générer des pages Web à partir de données structurées

Afin de rendre le site plus dynamique et plus facile à maintenir, j'ai utilisé la structure de données de type JSON donné. Toutes les données (titres, descriptions, compétences, apprentissages critiques, ressources) sont centralisées dans un fichier JavaScript structuré comme un objet JSON. Cette approche permet de séparer les données du contenu HTML. Le HTML sert uniquement de structure, tandis que le JavaScript se charge de lire les données et de les appliquer dans le DOM.
Ainsi, nous pouvons ajouter des informations dans le JSON sans avoir à modifier la page HTML. Le CSS permettait d'agir comme "Template" avec un design prédéfini.

(Personnel)

J'ai pu constater le véritable potentiel en ayant codé un site internet, qui se rafraichissait tous les jours à minuit avec un message au centre de la page. Je pouvais mettre en avance

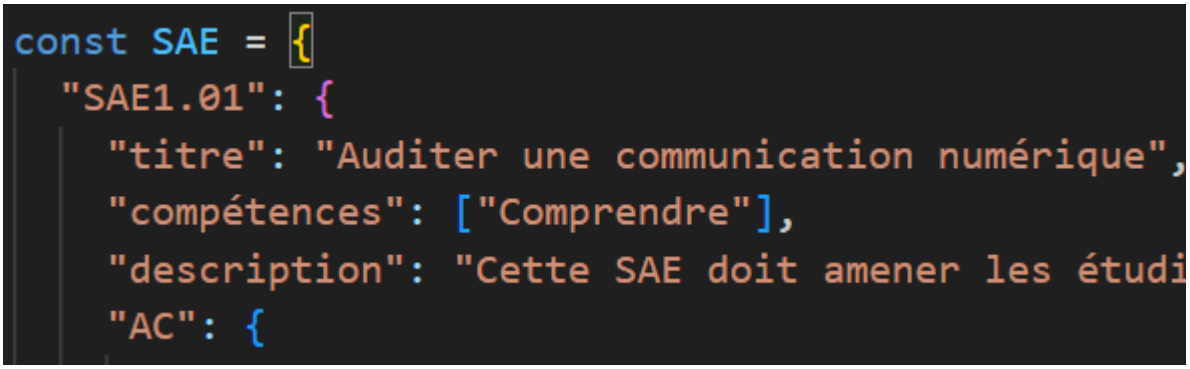
les messages sans devoir modifier le HTML et ces messages pouvaient être affichés des mois après que cela marcherait toujours.

Les Apprentissages Critiques sont stockés sous forme d'objets clé-valeur, où chaque clé correspond à un code d'AC et chaque valeur à sa description. Pour les afficher correctement, j'ai utilisé une méthode adaptée à cette structure de données.

Plutôt que de traiter les AC comme un tableau, j'ai parcouru l'objet à l'aide d'une boucle permettant de récupérer chaque clé et sa valeur. Cette approche permet :

- D'afficher clairement le code de l'AC suivi de sa description
- De conserver la structure officielle des Apprentissages Critiques
- D'obtenir un affichage lisible et cohérent pour l'utilisateur

Chaque AC est ensuite affiché sous forme de liste, ce qui facilite la lecture et permet à l'utilisateur de comprendre rapidement les compétences évaluées dans chaque SAE. Cette méthode garantit un affichage fidèle aux données tout en restant simple et efficace. Toutes les données du JSON est possible grâce à la constante SAE.



AC14.05 : Mettre en ligne une application Web en utilisant une solution d'hébergement standard

Une fois le site développé et testé en local, j'ai procédé à son transfert vers un serveur distant afin de le rendre accessible en ligne. Cette étape est essentielle pour vérifier le bon fonctionnement du site dans un environnement réel.

Dans un premier temps, j'ai utilisé la commande **python -m http.server** afin de lancer un serveur local. Cette méthode m'a permis de simuler le comportement du site sur un serveur web, de tester les chemins des fichiers, et de m'assurer que les liens, scripts et ressources se chargeaient correctement avant la mise en ligne.

L'utilisation de WinSCP m'a permis de transférer du PC au serveur local. Nous pouvons également regarder le résultat en mettant notre prénom et nom + iutmmmiweb02.uha.fr et verifier le contenu ainsi que la structure du site web. On pouvait voir le rendu du PC local par rapport au PC chez soi.

Cette approche m’a permis de passer efficacement d’un environnement de développement local à un environnement de production, tout en assurant la cohérence et le bon fonctionnement du site une fois hébergé en ligne.