

TypingTest

NoorMohammad Alizadeh – CIN4A

ETML – Ecole des Métiers Lausanne

Lundi 09 mai 2022 – Mercredi 08 juin 2022

Chef du projet : Raphael Pasche

Experts : Alain Roy, Yves Bertino

Table des matières

1	ANALYSE PRÉLIMINAIRE	4
1.1	INTRODUCTION	4
1.2	OBJECTIFS	4
1.3	PRÉREQUIS	4
2	PLANIFICATION INITIALE	4
3	ANALYSE/CONCEPTION	6
3.1	CONCEPT	6
3.1.1	Interface	6
3.2	MATÉRIEL ET LOGICIELS À DISPOSITION	7
3.3	LES RISQUES TECHNIQUES	7
3.4	POINTS À DÉCOUVRIR	7
3.5	STRATÉGIE DE TEST	7
3.6	PLANIFICATION DÉTAILLÉE	8
3.7	DOSSIER DE CONCEPTION	11
3.7.1	Environnement de développement	11
3.7.2	Outil de versioning	11
3.7.3	Serveur WEB	11
3.7.4	Figma	11
3.7.5	Carbon	11
3.7.6	Préparation de matériel	12
3.8	PRÉPARATION DE L'ENVIRONNEMENT	13
3.8.1	Documentation	13
3.8.2	Développement	13
4	RÉALISATION	14
4.1	DOSSIER DE RÉALISATION	14
4.1.1	Version des matériels/librairies utilisées	14
4.1.2	Construction la partie statique du site web (HTML)	14
4.1.3	Construction la partie statique du site web (CSS)	17
5	TESTS	22
5.1	DOSSIER DES TESTS	22

6	CONCLUSION	23
6.1	BILAN DES FONCTIONNALITÉS DEMANDÉES	23
6.2	BILAN DE LA PLANIFICATION	23
6.3	BILAN PERSONNEL.....	23
7	DIVERS	23
7.1	JOURNAL DE TRAVAIL.....	23
7.2	TABLE DES ILLUSTRATIONS	23
7.3	GLOSSAIRE	24
7.4	BIBLIOGRAPHIE.....	24
7.5	WEBOGRAPHIE.....	24
8	ANNEXES.....	25

1 ANALYSE PRÉLIMINAIRE

1.1 INTRODUCTION

TypingTest est un site web pour entrainer sa dextérité à la dactylographie. Il y aura deux modes de test différent sur le site, un mode par nombre de mots et le deuxième mode est par compte à rebours.

1.2 OBJECTIFS

- 1) Les différents modes de test : par nombres de mots, par compte à rebours
- 2) Un texte à recopier : les mots français et les mots générés avec des caractères aléatoires
- 3) Un chronomètre qui démarre quand l'utilisateur commence à taper et s'arrête quand il finit de taper ou le temps arrive à zéro dépendant quelle mode j'ai choisi
- 4) Gestion des utilisateurs à l'aide des cookies

1.3 PRÉREQUIS

- Modules ICT orienté WEB
- Usage des Cookies
- JavaScript

2 PLANIFICATION INITIALE

La planification initiale a été envoyée le premier jour du TPI aux experts et au chef du projet par email.

Cette planification visualise ce que j'ai globalement prévu comme tâches et me permettra de voir une liste de mes tâches à faire durant ce projet.

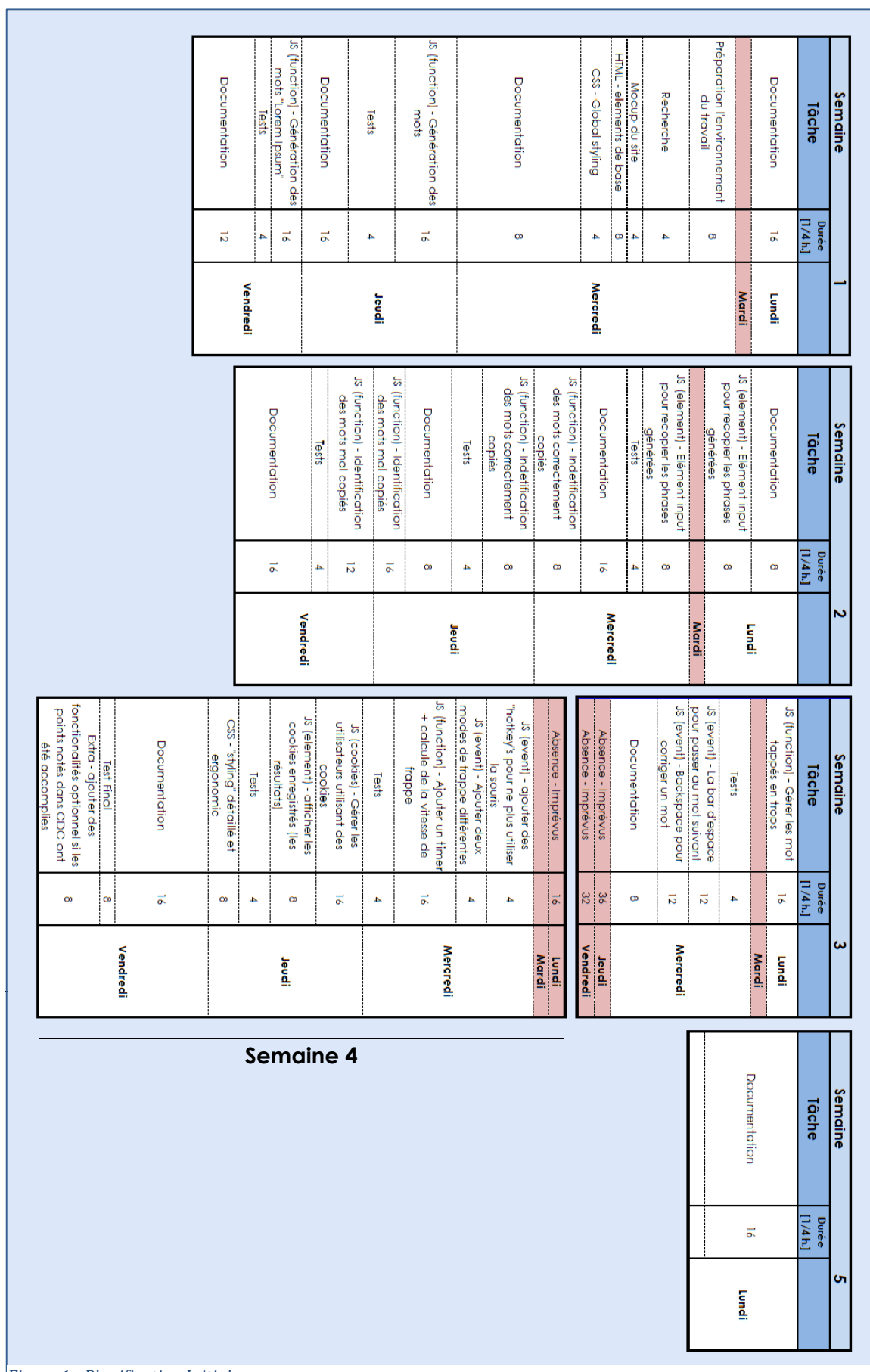


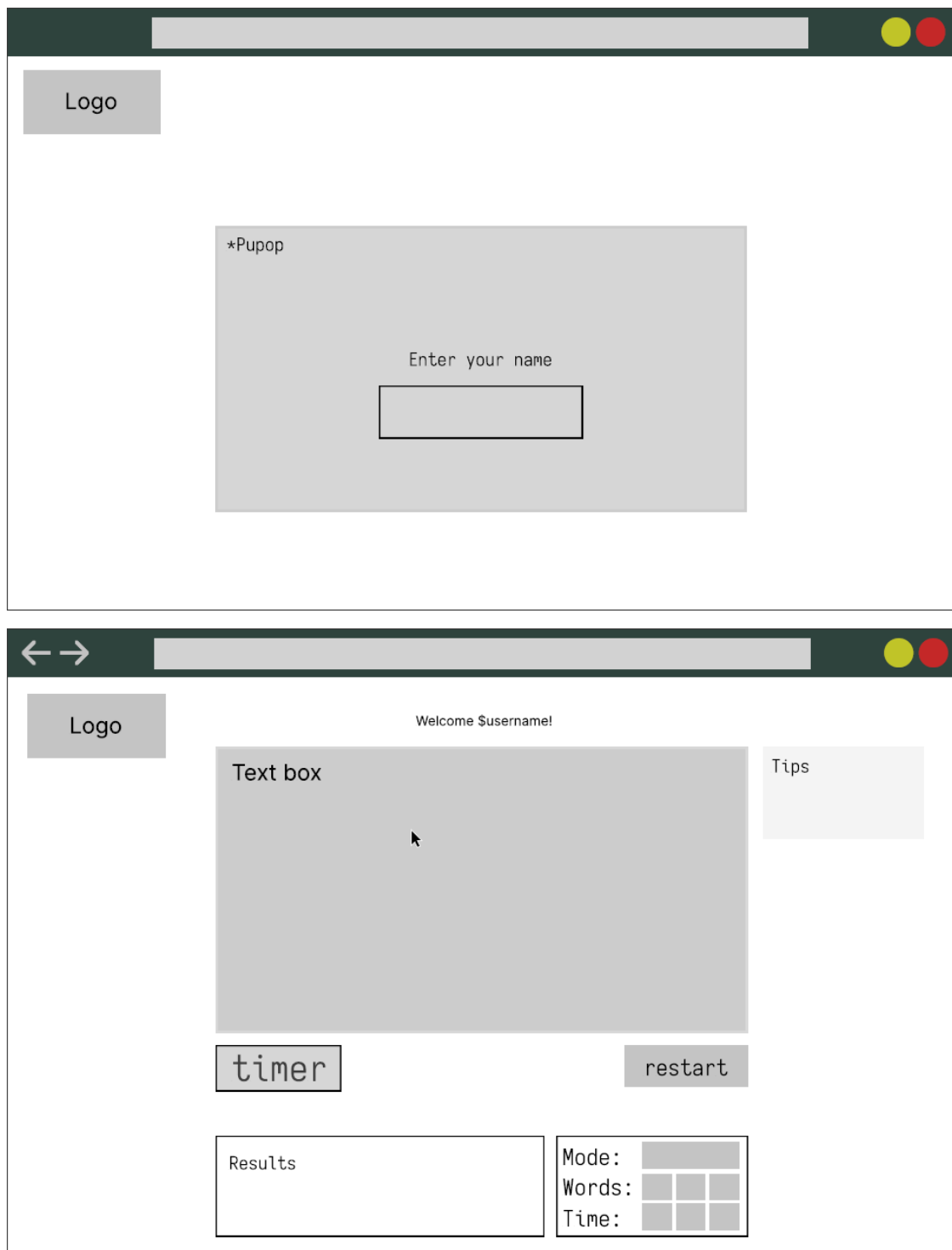
Figure 1 - Planification Initiale

3 ANALYSE/CONCEPTION

3.1 CONCEPT

Le plan du site est relativement simple et tout se passe en une page.

3.1.1 Interface



3.2 MATÉRIEL ET LOGICIELS À DISPOSITION

- Un PC standard de l'ETML (Windows 10)
- Visual Studio Code
- Serveur web sur cloud (Render.com)
- Suite Microsoft Office pour la documentation
- Un dépôt Git privé (GitHub)
- GitHub Desktop

3.3 LES RISQUES TECHNIQUES

Pendant 4-5 ans de ma présence à l'ETML, le module orienté WEB n'était mon point fort et mon projet est majoritairement basé sur JavaScript, le langage dont nous avons pas appris dans nos modules orienté WEB.

Le développement WEB m'a toujours passionné, même si je n'étais pas assez bien dans le domaine. Durant mon stage d'entreprise au Gymnase de Chamblandes, j'ai pu beaucoup travailler sur JavaScript et m'améliorer.

3.4 POINTS À DÉCOUVRIR

3.5 STRATÉGIE DE TEST

Les tests seront faits pour les fonctionnalités JavaScript vue les nombres des points à réaliser listés dans le cahier de charges. À la fin de création de chaque fonction, j'effectuerai un test pour valider si je peux passer à la fonctionnalité suivante.

Vous verrez une liste détaillée de mes tests dans le point 5.1.

3.6 PLANIFICATION DÉTAILLÉE

Semaine		1	
Tâche	Durée [1/4 h.]		Explications: qu'est-ce qui se fait et comment ?
Documentation	16	Lundi	Matin: Visite avec l'expert Fin matinée: Commencer à faire la planification initiale
Préparation l'environnement du travail	8	Mercredi	_Préparation du git et mes répertoires du travail _Configuration ssh pour accéder au git depuis terminal en local _Installation et configuration Vscod
Recherche	4		_Recherche des sources et des images différents pour le site _Trouver un utile pour créer le mockup du site _Chercher des idées pour le look du site
Mocup du site	4		
HTML - elements de base	8		_Commencer à créer la page HTML et mettre les balises nécessaires
CSS - Global styling	4		_Ajouter un styling global pour tout le site et pouvoir distinguer les elements html sur la page
Documentation	8		_Commencer à écrire le rapport du projet > Spécification > CDC > Analyse _Noter les tâches que j'ai fait pour Mercredi > Réalisation _enregistrer la dernière version de ces doc en format PDF pour les envoyer au chef du projet et aux experts
JS (function) - Génération des mots	16	Jeudi	_Créer la fonction quoteGen() qui s'active en appuyant sur un bouton et prendre les citations depuis un fichier json et les affiche aléatoirement dans le site. _Créer la fonction wordGen() qui permet de gerere les mot français aléatoires et les affiche dans le site web
Tests	4		_ Tester voir si les citations sont corectement afficher et à chaque rafraichissement du site une nouvelle citation s'afficher _ Le même test mais pour des mots français
Documentation	16		_Noter les tâches dans le journal du travail _Avancer dans le rapport et noter ces dernières tâches en plus de la résultat de leurs tests > Tests
JS (function) - Génération des mots "Lorem Ipsum"	16	Vendredi	_Créer la fonction charGen() qui génère des mots apartir des différents types de characters
Tests	4		_Tester si à chaques rafraichissement une novelle phrase se génère
Documentation	12		_Mettre à jour le journal du travail _Rapport > Réalisation _Préparer les PDF pour les envoyer
Total semaine	120		Max. 120

Figure 2 - Planification détaillée (Semaine 1)

Semaine		2	
Tâche	Durée [1/4 h.]		Explications: qu'est-ce qui se fait et comment ?
Documentation	8	Lundi	_Mettre à jour le rapport > Tests > Sources
JS (element) - Élément input pour recopier les phrases générées	8		_Ajouter un moyen de taper le texte dans une case et pouvoir le comparer avec les textes générés
JS (element) - Élément input pour recopier les phrases générées	8	Mercredi	_**continuation la tâche précédente
Tests	4		_donner un message d'alert quand le texte tapé n'est pas egale à la phrase générée
Documentation	16		_Mettre à jour le journal du travail avec les tâches du lundi et mercredi _Noter les résultats de testes que j'ai fait pour cette tâche _Préparer les DPF
JS (function) - Indetification des mots correctement copiés	8		_Ajouter un moyen d'identifier les character correctement tapés
JS (function) - Indetification des mots correctement copiés	8	Jeudi	_**
Tests	4		_Tester la distinction entre les mot corrects et les autres, couleur jaune pour les charachters correctement copié
Documentation	8		_Rapport > Réalisaition > Tests
JS (function) - Identification des mots mal copiés	16		_Ajouter un moyen d'identifier les character mal tapés
JS (function) - Identification des mots mal copiés	12	Vendredi	_Continuer sur l'identifcation des mots mal copiés _Distinguer les mots correctement tapés avec des mot incorrecte, les mots mal tappé seront barés
Tests	4		_Tester si on voit bien la difference des mots correcte et incorrectes
Documentation	16		_Noter les nouvelles tâches _Rapport > Réalisation > Tests _Préparation PDF
Total semaine	120		

Figure 3 - Planification détaillée (Semaine 2)

Semaine 3			
Tâche	Durée [1/4 h.]		Explications: qu'est-ce qui se fait et comment ?
JS (fonction) - Gérer les mot tapés en trop	16	Lundi	Créer la fonction qui detecte les caracteres tapés en trop et pousse les mots suivants si je continue à taper des caracteres en trop
Tests	4	Mercredi	_Tester les mots sont bien poussés, les caracteres en trop devraient pas aller en dessus des mots suivants
JS (event) - La bar d'espace pour passer au mot suivant	12		_Quand un mot est correctement tapé, en tapant sur l'espace je passe au mot suivant et je continue à taper ce nouveau mot
JS (event) - Backspace pour corriger un mot	12		_En tapant sur backspace, je pourrais corriger un mot
Documentation	8		_Journal des tâches effectués cette semaine _Noter les trois fonctionnalités ajouter cette semaine > Réalisation
Absence - Imprévis	36	Jeudi	Jeudi de l'Ascension
Absence - Imprévis	32	Vendredi	Pont de l'Ascension
Total semaine	120		Max. 120

Figure 6 - Planification détaillée (Semaine 3)

Semaine 4			
Tâche	Durée [1/4 h.]		Explications: qu'est-ce qui se fait et comment ?
Absence - Imprévis	16	Lundi	Lundi de Pentecôte
JS (event) - ajouter des "hotkey"s pour ne plus utiliser la souris	4	Mercredi	_Creer des "event handlers" pour naviguer dans le site sans utiliser la souris: charger nouvelle phrase à taper, refaire un teste de frappe, arreter le timer, etc...
JS (event) - Ajouter deux modes de frappe différentes	4		_Ajouter la fonctionnalité de passer aux différentes modes de frapper différents : normale, mots français. Lorem Ipsum
JS (fonction) - Ajouter un timer + calcule de la vitesse de frappe	16		_Ajouter un timer qui commence à compter dès que la personne commence à taper un mot et il s'arrête quand il fini à taper la phrase en entier ou il réussi pas à taper les mot avant que le timer arrive à zéro.
Tests	4		_Calculer nombre des mots tapés par minutes et donner le MPM, vitesse de frappe
JS (cookies) - Gérer les utilisateurs utilisant des cookies	16	Jeudi	_Tester que le timer est correcte, il arrête bien au moment dont le dernier character est tapé
JS (element) - afficher les cookies enregistrés (les résultats)	8		_Demander aux utilisateurs de taper leur nom pour accéder au site et ça sera enregistré dans un cookie
Tests	4		_Pouvoir afficher tout les cookies enregistrés dans le site dans la page de resultats
CSS - "styling" détaillé et ergonomic	8		_tester les cookies sont biens associé à la bonne personne ainsi ses resultats MPM, PDF(precision de frappe), DEH(date et heure)
Documentation	16	Vendredi	_Commencer à travailler sur l'ergonomie du site et le rendre agréable aux yeux
Test Final	8		_Journal du travail _Rapport > Réalisation > Tests > Conclusion > Blians
Extra - ajouter des fonctionnalités optionnel si les points notés dans CDC ont été accomplies	8		_Faire les tests finale pour donner un vue ensemble des testes dans le rapport
			_Si le temps me permet, j'ajouterai certains fonctionnalité dans le site
Total semaine	112		Max. 120

Figure 5 - Planification détaillée (Semaine 4)

Semaine 5			
Tâche	Durée [1/4 h.]		Explications: qu'est-ce qui se fait et comment ?
Documentation	16	Lundi	_Remplir et finir le journal du travail _Faire des dernières modifications dans le rapport, ajouter les photos de mes tâches (JDT), finir la partie des tests _Mise en page du rapport _Préparer les PDFs pour les envoyer aux experts et mon chef du projet _Imprimer mes documents pour les envoyer par courrier aux experts

Figure 4 - Planification détaillée (Semaine 5)

3.7 DOSSIER DE CONCEPTION

Pour ce projet les matériels à dispositions ont été proposé dans cahier de charge et je suis la plupart de ces propositions comme le choix des système d'exploitation car utilisation Windows 10 est plus pratique quand je suis à l'école.

3.7.1 Environnement de développement

Personnellement j'utilise souvent NeoVIM qui est une continuation d'un utile de traitement du texte connue s'appelé VIM. Main pour ce projet j'ai décidé de continuer avec la proposition dans mon cahier de charge. **Visual Studio Code**, car NeoVIM n'est pas un utile optimisé pour être utilisé dans environnement Windows. Une autre raison c'est que je peux aussi utiliser le « Five Server », une extension à VS Code permettant avoir un server PHP en live et se met à jour à chaque fois il y a un changement dans mon code HTML, CSS, JS ou PHP.

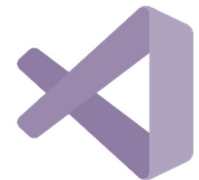


Figure 7 - Visual Studio Code

3.7.2 Outil de versioning

Normalement j'utilise GitLab parce que il a une meilleur offre gratuite comparant au GitHub **mais** dans ce projet j'utilisera **GitHub** comme utile de versioning. Et la raison est seulement que je ne veux pas mélanger mes projets de l'école avec mes projets personnels et je voulais pas créer un nouveau compte GitLab.

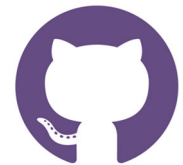


Figure 8 – GitHub.com

3.7.3 Serveur WEB

Dans ce projet, on m'a demandé d'utiliser un serveur WEB local comme uWamp. Mais après mes discussions avec mon chef du projet et mon expert, j'utilisera **Render**. Avec cet utile je peux déployer mon site rapidement depuis mon dépôt GitHub et le site sera accessible dans quelque petite minute, mon site sera accessible sur internet. Mais pour les petits tests en local j'utiliserai l'extension VS code que j'ai déjà nommée.



Figure 9 - Render.com

3.7.4 Figma

Figma c'est un outil récemment connu entre les développeurs WEB. Il permet de créer des maquets/interface pour les sites WEB. Il est disponible sur Windows, MacOS et Linux pour créer des interfaces et sur Android/iOS permettant de visualiser des prototypes Figma.



Note : Figma a la capacité de tourner le prototype créé en code HTML et CSS. Mais je l'utiliserai uniquement pour créer le schéma de mon site web et savegrader en format image.

Figure 10 - Figma

3.7.5 Carbon

Pour avoir une meilleure présentation d'un extrait de mon code dans ce rapport, j'ai utilisé un site web qui s'appelle **Carbon** (carbon.now.sh). Je peux copier des morceaux de code et les coller dans **Carbon**, ensuite je peux télécharger une image claire et bien lisible.



Figure 11 - Carbon.now.sh

3.7.6 Préparation de matériel

- **Editeur du texte** : Les configurations pour mon éditeur du texte étaient assez simples car j'ai toujours la même configuration même si je travaille chez moi ou à l'école. Il me fallait donc connecter le Visual Studio Code à mon compte GitHub pour synchroniser les configurations et installer toutes les extensions que j'utilise constamment.
- **Git** : Mon dépôt git à une structure simple et je ne vais pas utiliser des différentes branches parce que ce n'est pas un projet trop compliqué à gérer.

3.8 PRÉPARATION DE L'ENVIRONNEMENT

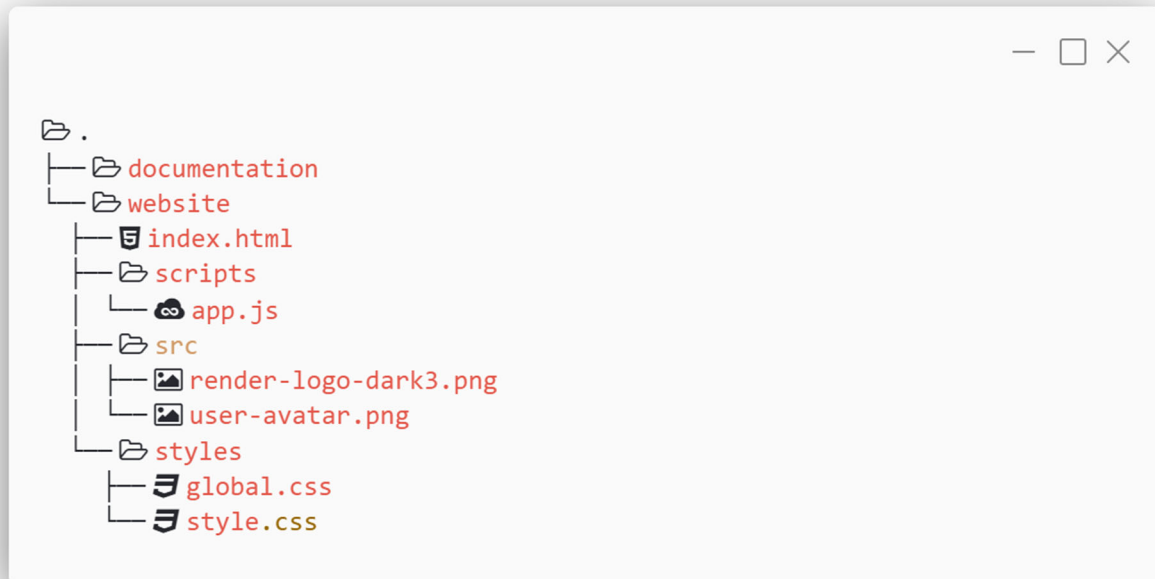


Figure 12 - Répertoires du travail

3.8.1 Documentation

3.8.2 Développement

Pour mon environnement du travail, j'ai une structure assez simple. Dans le répertoire racine, j'ai deux dossiers, le premier dossier s'appelle « documentation » qui contient tous mes documents comme mon rapport, planification et journal du travail. Le deuxième dossier contient les fichiers de mon site. Dans ce dossier j'ai organisé les fichiers pour « CSS » dans le dossier « styles », les fichiers « JavaScript » dans le dossier « scripts » et les images dans le dossier « src ». Et en fin mon fichier index Html se trouve dans la racine de dossier « website ».

Le fichier « index.html » est la page que vous verrez quand vous entrez dans le site. Le « app.js » qui contient toutes les fonctionnalités de mon site. Le fichier « global.css » contiendra la présentation globale de mon site et le fichier « style.css » contiendra la présentation spécifique de chaque élément dans mon site.

TODO : autres environnements

4 RÉALISATION

4.1 DOSSIER DE RÉALISATION

4.1.1 Version des matériels/librairies utilisées

- Visual Studio Code v1.67.1
- FontAwesome Icon Library v6.1.1 and v4.7.0
- Live Server (Five Server) v0.1.4
-

4.1.2 Construction la partie statique du site web (HTML)

J'ai commencé le développement de mon projet par mettre en places les différents éléments visuels de mon site et séparer le site par des différentes sections.



```
<head>
  <link rel="stylesheet" href="//use.fontawesome.com/releases/v4.7.0/css/all.css">
  <link rel="stylesheet" href="//use.fontawesome.com/releases/v5.15.4/css/all.css">
  <link rel="stylesheet" href="/styles/global.css">
  <link rel="stylesheet" href="/styles/style.css">
  <script src="/scripts/app.js" defer></script>
  <title>TypingTest</title>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="author" content="NoorMohammad Alizadeh">
  <meta name="description" content="Un site pour tester la vitesse de frappe.">
  <link rel="shortcut icon" href="/src/favicon.png" type="image/x-icon">
</head>
```

Figure 13 - HTML<head>

Pour la balise **head** de mon fichier HTML :

- a. J'ai ajouté deux liens vers les différentes versions de FontAwesome qui me permet s'accéder à une grande librairie des icônes pour mon site.
- b. Deux liens pour accéder à mes fichiers css, « global.css » et « style.css ».
- c. Un lien vers mon fichier JavaScript, « app.js ». Dans cette ligne j'ai aussi ajouté l'attribut « defer », cet attribut « lazy load » ma page JavaScript et permet au fichier HTML de charger complètement et ensuite charger le fichier JavaScript. La raison pour laquelle je charge le fichier script après le fichier html est la lourdeur de ma page script, cela peut créer des problèmes car mon fichier script ne verra pas les éléments html qui sont en dessous de lui et ne pourra donc pas les gérer.
- d. Les balises **META** qui sont pour définir l'encodage de caractère dans le site. Ajouter la comptabilité avec des différentes versions de l'Internet Explorer. Comptabilité avec des appareils mobiles ou des appareils avec des différentes tailles de l'écran. L'auteur et description du site.
- e. Lien pour afficher l'icône de mon site.

J'ai divisé la balise **body** en quatre sections principales, ces sections sont **header**, **main**, **aside** et **footer**.

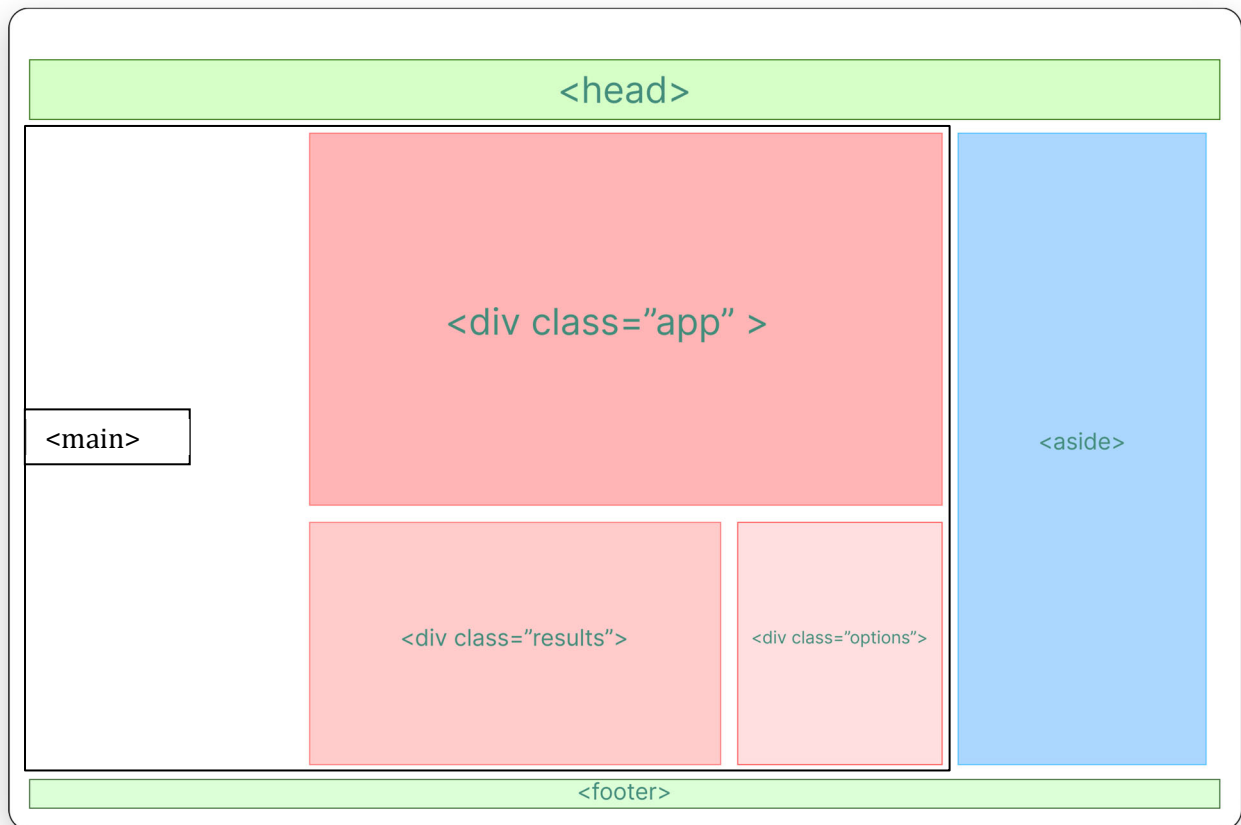


Figure 14 - HTML<body> plan

La balise **header** contiendra logo, nome et la barre de navigation de mon site

```
<header>
  
  <p>TypingTest</p>
  <ul>
    <li><a href="#" id="aboutPop" onclick="aboutPop()">À propos</a></li>
    <li><a href="https://etml.ch">ETML</a></li>
  </ul>
</header>
```

Figure 15 - HTML<body>header>

La balise **<main>** qui fait une grande partie du site, contient mon application donc le TypingTest. Cette balise s'est divisé en plusieurs `<div>` qui me permettront de bien les positionner dans différents endroits en utilisant CSS et pouvoir facilement accéder chaque élément depuis mon fichier Java Script.

Dans la balise **main**, un **div** qui celui s'est divisé en différentes parties. Un header pour le message bienvenu de mon site web, un élément **input** dans laquelle les mots tapés par l'utilisateur seront capturés, un autre **div** pour visualisation de mon texte, un minuteur et en fin un bouton pour redémarrer mon test

Mon bouton a un **onclick** événement permettant à l'utilisateur d'exécuter une fonction JavaScript, dans mon cas, la fonction s'appelle **newQuote()** et je vais en parler dans la chapitre **4.1.4 JavaScript**.

```
<div class="app">
  <div class="header">
    <h3 id="greetMessage">Hello <span id="username">username</span></h3>
  </div>
  <input type="text" id="inputeQuote" name="InputQuote">
  <div id="quote">
    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptatibus non commodi,
    dolores porro architecto id! Laboriosam consequatur, numquam unde magnam perspiciatis temporibus et
    corporis aperiam, id possimus adipisci repellat! Minima.</p>
  </div>
  <p id="timer">00</p>
  <button name="Restart" id="restartBtn" onclick="newQuote()">Redémarrer</button>
</div>
```

Figure 16 - HTML<body>main>div.app>

La prochaine balise **div** dans mon code est de class « content » qui contiendra mes résultats actuels d'un test et historiques des anciens résultats.

```
<div class="results">
  <div class="curr-results">
    <p id="wpmLabel">00</p> MPM
    <p id="accLabel">00</p> %
  </div>
  <div class="past-resutls">
    <p>Historiques</p>
    <ul class="past-result-head">
      <li></li>
    </ul>
    <p></p>
  </div>
</div>
```

Figure 17 - HTML<body>main>div.content: results

Ensuite, j'arrive à une balise **div** de classe « options ». Cette balise rassemble les différentes options qui sera implémentées dans mon site web. Il y a un bouton qui sera utilisé pour changer le mode de test donc le mode « Words » et le mode « Time ». J'ai mis trois boutons qui définissent nombres des mots dans le mode *Words*. L'utilisateur peut choisir 20, 30 et 70 mots pour son test.

Trois derniers boutons servent à changer le temps du chronomètre dans le mode *Time*.

```
<div class="options">
  <p>Mode :</p>
  <button name="Mode" onclick="toggleMode()">Normal</button>
  <p>Words :</p>
  <button id="wordsBtnOne" class="optionBtn" onclick="setBtnOne()">20</button>
  <button id="wordsBtnTwo" class="optionBtn" onclick="setBtnTwo()">50</button>
  <button id="wordsBtnThree" class="optionBtn" onclick="setBtnThree()">70</button>
  <p>Time :</p>
  <button id="timesBtnOne" class="optionBtn" onclick="setTimerBtnOne()">15</button>
  <button id="timesBtnTwo" class="optionBtn" onclick="setTimerBtnTwo()">30</button>
  <button id="timesBtnThree" class="optionBtn" onclick="setTimerBtnThree()">60</button>
</div>
```

Figure 18 - HTML<body>main>div.content: options

Maintenant en dehors de la balise *main* j'ai mes dernières parties de mon code HTML qui sont les balises « aside » et le « footer ».

Dans la balise aside se trouvera les messages d'erreurs ou des astuces pour les utilisateurs. Le footer est aussi explicite et il contient le Copyright du site web.

```
<aside>
  <div id="tips">
    <p></p>
  </div>
</aside>
<footer>
  <p class="copyright">Copyright © 2022 ETML - TypingTest</p>
</footer>
```

Figure 19 - HTML>body>aside&footer

4.1.3 Construction la partie statique du site web (CSS)

Quand j'ai mis toutes les éléments HTML du site, j'ai commencé la mise en forme de ces éléments, de cette façon je peux trouver et différencier les éléments sur le site pour que je puisse facilement commencer la partie JavaScript de mon projet.

:root { Je commence par importer deux différentes polices et ensuite le premier sélecteur que j'ai mis est le « :root », ce sélecteur vise la racine de ma page HTML, donc la balise *<html>*. Je place mes variables CSS dans ce sélecteur parce que je veux y avoir accès depuis n'importe où dans mon fichier CSS.

Mes trois variables initiales sont deux polices et la bordure test pour que je puisse différencier les éléments dans le site.

```
@import url('https://fonts.googleapis.com/css2?family=Roboto&display=swap');
@import url('https://fonts.googleapis.com/css2?family=Source+Code+Pro:wght@600&display=swap');

:root{
  --font-mono: 'Roboto', sans-serif;
  --font-scp: 'Source Code Pro', monospace;
  --border-test: 1px solid red;
}
```

Figure 20 - CSS:root

body { Prochain sélecteur est le « body », j'ai appliqué zéro pixel du marge et écart de remplissage à tous les éléments du site pour ne pas confondre les tailles de ses éléments. C'est plutôt ma propre manière de commencer le CSS pour n'importe quel projet et je ne sais pas si c'est la manière standard pour CSS. J'ai aussi défini la police par défaut du mon site.

```
body {
  margin: 0px;
  padding: 0px;
  font-family: var(--font-mono);
}
```

Figure 21 - CSS:body

header { Pour la mise en forme des éléments, je commence par le sélecteur **header** qui représente la barre en haut dans ma page web. Elle aura 100 pourcent de largeur et une hauteur de 70 pixel. Sa position est fixée en haut de la fenêtre et pas le document HTML, c'est-à-dire que quand ma page est assez large et j'aurai besoin de défiler la page, cette barre reste à sa position en haut de la fenêtre.

Ma variable bordure est utilisée pour monter à quoi mon élément **header** ressemble.

Affichage de cet élément est en mode grille (grid), ce qui me permet de mieux organiser les éléments enfant dans **header**. J'ai aussi défini combien de colonne mon **header** devrait avoir. Je veux 5 colonne avec la même taille, donc j'utilise la fonction **repeat()** pour répéter ma valeur de la taille 5 fois. L'unité de mesure que j'ai utilisée est une *unité fractionnaire*. **1fr** est **une partie** de ma grille, donc 5 fois **1fr** veut dire que j'ai 5 parties (colonne) égales dans ma grille.

NOTE : La raison pour laquelle j'ai utilisé ces symboles (>) comme ceci (**body > header**), c'est que si je ne les mets pas. Tous les **headers** dans le secteur **body** seront affectés, mais dans ce cas précis je veux uniquement modifier le **header** qui se trouve juste après **body**.

```
body > header {
  width: 100%;
  height: 70px;
  position: fixed;
  top: 0px;
  border-bottom: var(--border-test);
  /*----*/
  display: grid;
  grid-template-columns: repeat(5, 1fr);
}
```

Figure 22 – CSS - header

Finalement ma barre ressemble à l'image ci-dessous. Les cellules sont aussi visibles, 5 cellules font exactement la même taille.

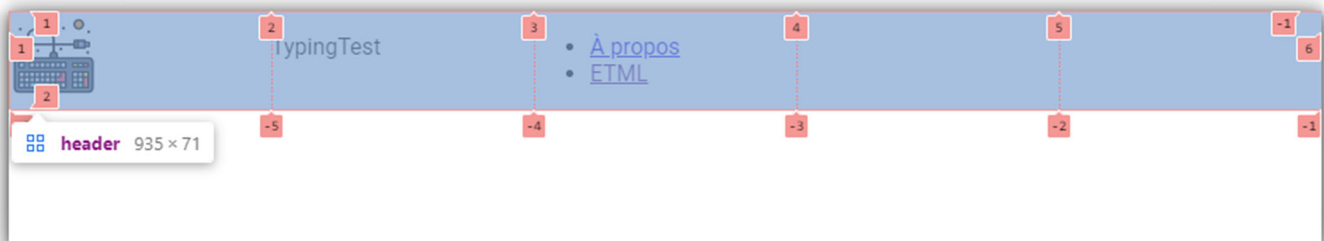


Figure 23 - CSS – header affichage

Pour les éléments dans la grille, je commence de la gauche au droit, l'élément image commence par la première colonne et il a une marge de dix pixel à gauche.

NOTE : ici il n'y a pas de « > » entre *header* et *img*, ça veut dire que ma modification vise tous les *img* qui se trouve dans la balise *<header>*.

Prochain élément est un secteur *p* qui présente le nom de mon site. Il a une police différente de la police par défaut du site. L'unité utilisée pour la taille de la police est une unité relative, relative à la police par défaut du navigateur web depuis lequel mon site a été accédé. Dans ce cas, *2em* est 2 fois plus grand que la police de navigateur. J'ai centré le texte et spécifié que sa cellule commence depuis la troisième colonne.

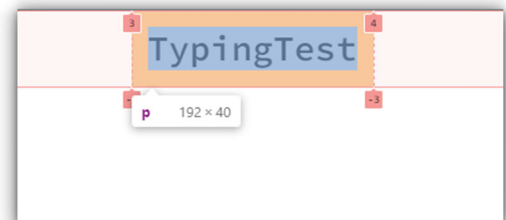


Figure 24 - CSS - header>p element

Le dernier élément dans mon *header* est une liste *ul*. Sa position commence de quatrième colonne à la sixième colonne. Les éléments li dans cette liste sont retiré de leurs flux normale et placé sur le côté droit de liste *ul* et leur mise en forme est désactivé. Les liens ont non plus leur décoration du texte.

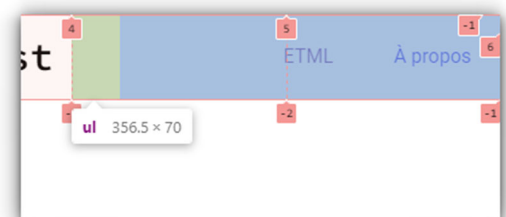


Figure 25 - CSS - header>ul liste

```
body > header img {
  grid-column-start: 1;
  margin-left: 10px;
  height: 60px;
}
body > header p {
  font-family: var(--font-scp);
  font-size: 2em;
  text-align: center;
  margin: 15px;
  grid-column-start: 3;
}
body > header ul {
  grid-column-start: 4;
  grid-column-end: 6;
  margin: 0px;
}
body > header li {
  float: right;
  padding: 25px;
  margin: 0px;
  list-style-type: none;
}
body > header a {
  text-decoration: none;
}
```

Figure 26 - CSS - header img/p/ul/li/a

Le prochain secteur, c'est le *div.app* qui contiendra les textes à afficher, le test du frappe, chronomètre et le bouton redémarrage du test. J'ai fait une mise en forme basique pour cette partie, la taille du canevas (*div.app*) est de 800 à 500 pixel. Une marge de 100 pixel pour donner une espace entre la barre de navigation et ce canevas. Les côtés gauche et droit one **une marge automatique** en plus de sa position qui est « relative » qui fait entièrement centrer le canevas.

Dans ce canevas, il y a le *div.quote* qui contient mes textes, sa longueur prends automatiquement la taille du canevas parent quand il y a une texte dans ce *div*. Son hauteur est de 20em, donc 20 fois plus grand que la taille de la police, autrement dit, ce *div.quote* contiendra 20 lignes (je crois).

```
div.app {  
  width: 800px;  
  height: 500px;  
  margin: 100px auto auto auto;  
  position: relative;  
  padding: 15px;  
  border: var(--border-test);  
}  
div.app > div.quote {  
  width: auto;  
  height: 20em;  
  border: var(--border-test);  
}
```

Figure 27 - CSS - `div.app > div.quote`

Le site web ressemble à l'image ci-dessous après les mises en forme CSS :



Figure 28 - CSS - L'état du site jusqu'ici

5 TESTS

5.1 DOSSIER DES TESTS

6 CONCLUSION

6.1 BILAN DES FONCTIONNALITÉS DEMANDÉES

6.2 BILAN DE LA PLANIFICATION

6.3 BILAN PERSONNEL

7 DIVERS

7.1 JOURNAL DE TRAVAIL

7.2 TABLE DES ILLUSTRATIONS

FIGURE 1 - PLANIFICATION INITIALE	5
FIGURE 2 - PLANIFICATION DÉTAILLÉE (SEMAINE 1)	8
FIGURE 3 - PLANIFICATION DÉTAILLÉE (SEMAINE 2)	9
FIGURE 4 - PLANIFICATION DÉTAILLÉE (SEMAINE 5)	10
FIGURE 5 - PLANIFICATION DÉTAILLÉE (SEMAINE 4)	10
FIGURE 6 - PLANIFICATION DÉTAILLÉE (SEMAINE 3)	10
FIGURE 7 - VISUAL STUDIO CODE	11
FIGURE 8 – GITHUB.COM	11
FIGURE 9 - RENDER.COM	11
FIGURE 10 - FIGMA	11
FIGURE 11 - CARBON.NOW.SH	11
FIGURE 12 - RÉPERTOIRES DU TRAVAIL	13
FIGURE 13 - HTML<HEAD>	14
FIGURE 14 - HTML<BODY> PLAN	15
FIGURE 15 - HTML<BODY>HEADER>	15
FIGURE 16 - HTML<BODY>MAIN>DIV.APP>	16
FIGURE 17 - HTML<BODY>MAIN>DIV.CONTENT: RESULTS	16
FIGURE 18 - HTML<BODY>MAIN>DIV.CONTENT: OPTIONS	17
FIGURE 19 - HTML>BODY>ASIDE&FOOTER	17
FIGURE 20 - CSS:ROOT	18

7.3 GLOSSAIRE

7.4 BIBLIOGRAPHIE

Création : 11.05.2022
Impression : 25.05.2022 17:48
Nooralizadeh-rapport.docx

8 ANNEXES
