



REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET
POPULAIRE
MINISTRE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA
RECHERCHE SCIENTIFIQUE
ECOLE NATIONALE SUPERIEURE D'INFORMATIQUE

Classes Préparatoires Intégrées (CPI)

2^{ème} année

PROJET

Editeur web WYSIWYG - DANYA

Sujet N° : 03

Equipe N° : 26

1. Gacem Abderraouf(CHEF D'EQUIPE)
2. Mekhalfa Takieddine
3. Cherief Yassine
4. Brahim Yacine
5. Kerris Nadir Islam
6. Djellal Abderrahmane

Encadrée par: Mme BENHAMIDA et Mme SAID_LHADJ

Client : Mr Fouad DAHAK

Année Universitaire : 2016-2017

Remerciements

Avant tout nous remercions Dieu le tout puissant de nous avoir donné le courage et nous avoir guider pour pouvoir mener à bien ce travail

C'est avec toutes reconnaissances qu'on tient à écrire « Merci » sincère à nos encadreuses pour leur soutien, leur enseignement et leurs conseils tout au long de ce projet qui vient de s'écouler, qui ont pris la peine de nous corriger à chaque fois, d'être patientes devant nos bêtises et erreurs, merci de nous apprendre comment vivre un projet.

On tient aussi de remercier tous les membres de l'équipe, pour leurs efforts continus, leur esprit d'équipe et leur sérieux.

Sans oublier tous ceux qui ont participé à la réalisation du projet du pré ou du loin

Table des matières

Remerciements	2
Avant tout nous remercions Dieu le tout puissant de nous avoir donné le courage et nous avoir guider pour pouvoir mener à bien ce travail.....	2
C'est avec toutes reconnaissances qu'on tient à écrire « Merci » sincère à nos encadreuses pour leur soutien, leur enseignement et leurs conseils tout au long de ce projet qui vient de s'écouler, qui ont pris la peine de nous corriger à chaque fois, d'être patientes devant nos bêtises et erreurs, merci de nous apprendre comment vivre un projet.	2
On tient aussi de remercier tous les membres de l'équipe, pour leurs efforts continus, leur esprit d'équipe et leur sérieux.	2
Sans oublier tous ceux qui ont participé à la réalisation du projet du pré ou du loin.....	2
Table des matières.....	3
La table des figures	4
Introduction	5
I. Présentation de l'état de l'art.....	6
1. Le langage de balisage HTML	6
2. Les interpréteurs HTML.....	6
3. L'éditeur Code-Mirror.....	7
II. La présentation du projet :	8
1. Les objectifs à atteindre.....	8
2. Le travail à faire.....	8
III. La conception :	9
1. Méthodes et techniques du travail	9
2. Les allègements portés à l'écriture du code D :	12
3. La maquette de l'interface graphique.	13
4. Découpage modulaire.....	13
IV. La réalisation :	13
1. La mise en évidence des jeux d'essais.....	13
2. La présentation des résultats obtenus	14
3. L'analyse de certains modules.....	15
4. Page d'accueil du site :	16
5. La réalisation de l'interface graphique	17
.....	18
6. Les outils utilisés :	19
7. La Base de Données :	19
.....	20

V.	La conclusion :	20
	Les propres perspectives.....	21
VI.	Annexe :	21
	Les cahiers de charges :.....	21
	Diagramme de Gantt:	33

La table des figures

Figure 1	Le DOM et sa représentation arborescente.....	10
Figure 2	<p> Text </p>	10
Figure 3	La maquette de l'interface graphique	13
Figure 4	un jeu d'essai en langage D	14
Figure 5	visualisation du jeu d'essai (figure4).....	15
Figure 6	page d'accueil de notre site	16
Figure 7	aperçu de l'interface de la plateforme DANYA.....	17
Figure 8	illustration du highLiting et la détection d'erreurs de syntaxe	17
Figure 9	nouveau style des sections spéciales.....	18
Figure 10	différents styles de ruban	18
Figure 11	menu d'utilisation et d'upload d'images	19
Figure 12	menu de sauvegarde et d'exportation de code D	20

Introduction

Un bon canal de communication est le souhait de toute personne ayant quelques choses à transmettre.

Les sites internet constituent, de nos jours, un des moyens les plus efficaces pour réaliser ce type de désir, seulement, concevoir un site web n'est pas une tâche facile pour une personne avec des connaissances réduites en informatique, plus précisément, ceci nécessite des connaissances des langages que traduisent les navigateurs, à savoir HTML et CSS en plus des événements en JavaScript.

Ce sont des langages au sens propre du terme car ils ont une syntaxe bien précise et des mots clés bien définis or ceci n'est pas à la portée d'une personne quelconque vue que l'utilisation de ces langages nécessite une formation et sont donc loin d'être intuitives.

C'est pour ça que des éditeurs d'aide à la création des sites existent, certains se base sur l'aspect graphique pour libérer l'utilisateur à connaissances modestes en informatique de l'écriture du code, alors que pour nous, on simplifie au gens l'accès à la création et la maintenance des sites web par le biais d'une plateforme d'édition appelée DANYA qui se base sur la définition d'un nouveau code, en l'occurrence le code D, qui est un langage simple avec une syntaxe intuitive et des mots clés qui introduisent des fonctions de formatage de texte et de création d'objets via une interface d'aide à l'insertion de ces éléments avec possibilité de visualisation des résultats, sauvegarde et exportation en langage D du code, et maintenance d'un site conçu préalablement en langage D .

« Le langage D est un langage de rédaction d'articles sous la plateforme DANYA. C'est un ensemble d'opérations dont chacune permet de formater le texte ou de créer des objets visuels assez attrayants. Le langage est simple et la syntaxe est des plus intuitives. » comme le site Mr. Dahak créateur de la plateforme DANYA et du langage D.

L'objet de notre projet est de mettre en place une plateforme pour éditer et interpréter le langage D.

I. Présentation de l'état de l'art

Le Web :

Ou « WorldWideWeb » est un système hypertexte fonctionnant sur internet qui permet grâce à un navigateur de consulter des pages appelées site Web. C'est l'une des applications de l'internet.

« Il a été inventé par Mr [Tim Berners-Lee](#) en 1991 le fondateur de W3C (WorldWideWeb Consortium) qui est un organisme de standardisation qui définit les nouvelles versions des langages liés au Web » qu'on trouve entre autres : HTML & CSS. Ces deux langages qui sont à la base du fonctionnement de tous les sites web d'aujourd'hui.

1. Le langage de balisage HTML

Hypertext Markup Language abrégé **HTML** c'est un langage de balisage qui permet d'écrire et de structurer et de mettre en forme le contenu des pages Web et d'inclure des ressources MultiMedia telles que les vidéos les images...etc. Il est souvent utilisé avec JavaScript et CSS.

Il est connu comme étant un langage de description de format de document permettant ainsi d'enrichir l'information textuelle en utilisant les **balises** comme unités syntaxiques délimitant une séquence de caractères ou marquant une position précise à l'intérieur d'un flux de caractères.

« La structure d'un document html peut être vue comme étant une véritable structure d'un arbre ayant une racine contenant tous les autres éléments ». Chaque élément fait partie du contenu d'un autre élément, cette structure d'arbre est utilisée par la structure de formatage qui en est dérivée pour l'application des feuilles de style en cascade.

HTML a connu plusieurs évolutions depuis sa création : commençant par HTML1.0 puis HTML2.0, HTML3.2 et HTML4.0 puis on est passé à la version actuelle (la plus récente) HTML5.0.

2. Les interpréteurs HTML

Modifier le contenu d'une page web sans fouiller dans son code n'était pas possible avant l'apparition des éditeurs HTML qui avaient pour but de faciliter non seulement aux débutants dans HTML mais aussi pour les experts le formatage et la mise en forme de la page web.

Il existe deux types d'éditeurs html : les éditeurs WYSIWYG, « What You See Is What You Get », et les éditeurs HTML de codes standards.

Les éditeurs WYSIWYG sont particulièrement bien adaptés si vous voulez créer un site sans connaissances ou peu des langages HTML et CSS. Toutefois, certaines notions de programmation sont nécessaires pour corriger les « bugs. ». En effet le principal problème des Wysiwyg est le fait qu'ils génèrent souvent de mauvais codes ; des balises superflues, des parties de code difficiles à comprendre et parfois quelques bugs qui doivent être corrigés.

Les éditeurs HTML de codes standards sont pour leur part très utiles car ils sont dotés des fonctionnalités facilitant aux développeurs de taper leurs code en leurs fournissant des raccourcis pour générer des parties du code, de fermer une balise ouvrante, ils leur permettent aussi de **mettre en surbrillance et d'appliquer une coloration au code.** (À noter que ces éditeurs sont aussi très pratiques pour les autres langages de programmation).

Parmi Ces éditeurs HTML on peut citer :

- **NicEdit :**

NicEdit est un éditeur WYSIWYG pour les sites Web. Connu par sa simplicité et sa rapidité du traitement « NicEdit est extrêmement léger et peut être facilement intégré dans n'importe quel site avec un impact minimal tout en offrant aux visiteurs un moyen efficace de s'exprimer en texte enrichi ». Contrairement à certains éditeurs de contenu comme TinyMCE ou FCKEditor. NicEdit est très léger et reste assez minimaliste. Il dispose de certaines fonctionnalités basiques à savoir : Mise en forme du texte et police, alignement, gestion des liens et gestion des images et upload sur imageshack.

- **TinyMCE :**

Connu sous le nom de **Tiny Moxiecode Content Editor** est un éditeur de HTML de type WYSIWYG, écrit en JavaScript, indépendant de la plateforme. L'éditeur propose des outils de mise en forme HTML, comme gras, italique, soulignement, des listes numérotées et des listes à puces, les différents types d'alignements, le placement en ligne d'images et de vidéos. Il permet aussi aux utilisateurs d'un site web d'éditer des documents HTML en ligne.

- **CKEditor :**

Sorti en 2003 CKEditor est un éditeur de texte/HTML de code source ouvert (Open Source) très simple à utiliser offrant plusieurs outils permettant à l'utilisateur de créer sa page web facilement.

CKEditor est devenu l'un des meilleurs éditeurs HTML grâce à ses nombreuses fonctionnalités qui sont tous accessibles sans aucune limitation et surtout du fait qu'ils sont très simples à utiliser, parmi ses fonctionnalités on peut citer :

- **Advanced Paste from MW :** Cette fonctionnalité permet de coller un contenu directement à partir de Microsoft Word et de maintenir la mise en forme du contenu d'origine avec une sortie HTML propre.

3. L'éditeur Code-Mirror

Code-Mirror est un éditeur de texte polyvalent écrit en JavaScript. Il est spécialisé dans l'édition supportant plus de 100 langues, il est doté de plusieurs extensions qui implémentent des fonctionnalités d'édition avancées.

Parmi ses principales caractéristiques on peut citer :

- Syntax Highlightings.
- Indentation automatique afin d'améliorer la lisibilité du code.
- Une grande capacité pour gérer les documents volumineux (qui contiennent des milliers de lignes).
- Une grande interface de programmation applicative.

A noter que cet éditeur de texte est utilisé dans plusieurs outils de développement (devTools) à savoir Firefox, Chrome, Safari, Adobe Brackets et plusieurs autres projets.

II. La présentation du projet :

1. Les objectifs à atteindre

Ce projet simule un travail d'une équipe en service d'un client, on se voit donc avec des objectifs d'un impact aussi bien à long terme qu'à court terme, on site alors :

- Réaliser un projet en équipe : ceci englobe synchronisation et coordination entre les membres de l'équipe
- Planifier un travail sur une durée donnée (ANNEXE)
- Répartition équitable du travail nécessaire pour la réalisation du projet (ANNEXE)
- Apprendre à travailler sous les consignes d'un chef d'équipe
- Apprendre à gérer les problèmes et conflits au sein d'une équipe
- Apprendre à réaliser les tâches dans un cadre d'un volume horaire donné.
- Evaluer et remplir les tâches d'un cahier de charges
- Développer l'aspect de communication entre membres de l'équipe (circulation de l'information)
- Développer l'aspect de communication avec le client (analyse de ses besoins, et présentation du travail réalisé).
- Développer l'aspect de communication avec les encadreurs (présentation périodique de l'état d'avancement, prise en considération des remarques et conseils donnés).
- Apprendre à mettre en valeur son travail par la mise en place d'un rapport résumant les tâches réalisées tout au long du projet.
- Elimination de la limite du langage de programmation en apprenant des nouveaux langages qui correspondent au besoin du projet
- Respecter le délai de livraison
- Mettre en place une charte de rédaction et de codification et la respecter

2. Le travail à faire

Il s'agit dans ce projet de donner accès au public, amateur en informatique, à la création et maintenance de site web, le travail consiste alors à :

- Réaliser un éditeur pour y écrire en langage D :
 - Possibilité de l'opération copier-coller
 - Possibilité de rétablir l'état après modification (annuler les modifications)
 - Différencier les mots clés du langage D du reste du code
- Equiper l'éditeur d'une interface d'aide pour libérer l'utilisateur de l'écriture du code

- L'interface doit être intuitive et doit contenir zone de visualisation du résultat du code D (on parle d'interprétation)
- L'éditeur doit avoir la possibilité de récupérer et de modifier un code déjà existant à partir d'un fichier ou à partir d'une sauvegarde dans le site.
- Mettre en place la possibilité d'exporter les éléments créés sous forme de fichier contenant du code D.

III. La conception :

1. Méthodes et techniques du travail.

- **Vue globale :**

A la première réflexion au projet, On peut sentir directement le besoin d'un moyen pour interpréter graphiquement les composants fondamentaux qui composent un site généré par le langage D, puis si on trouve un moyen, comment peut-on décomposer les instructions D en un ensemble de commandes triviales ? Et comment peut-on gérer les opérations imbriquées qui ne suivent aucun modèle spécifique ?

- **Le DOM :**

« Le **Document Object Model** (en abrégé **DOM**) est une interface de programmation normalisée par le W3C, qui permet à des scripts d'examiner et de modifier le contenu d'une page web »

Travailler avec le DOM était nécessaire pour réaliser l'interprétation, avant d'illustrer l'utilité du **DOM** dans notre travail, on donne une petite explication :

Le DOM est un arbre qui représente la relation parentale, et fraternelle des éléments HTML (paragraphe, image, lien ...), par exemple un paragraphe peut contenir un fils qui constitue une image, et ayant une div (centenaire) comme père.

Ci-dessous une figure illustrant cette définition :

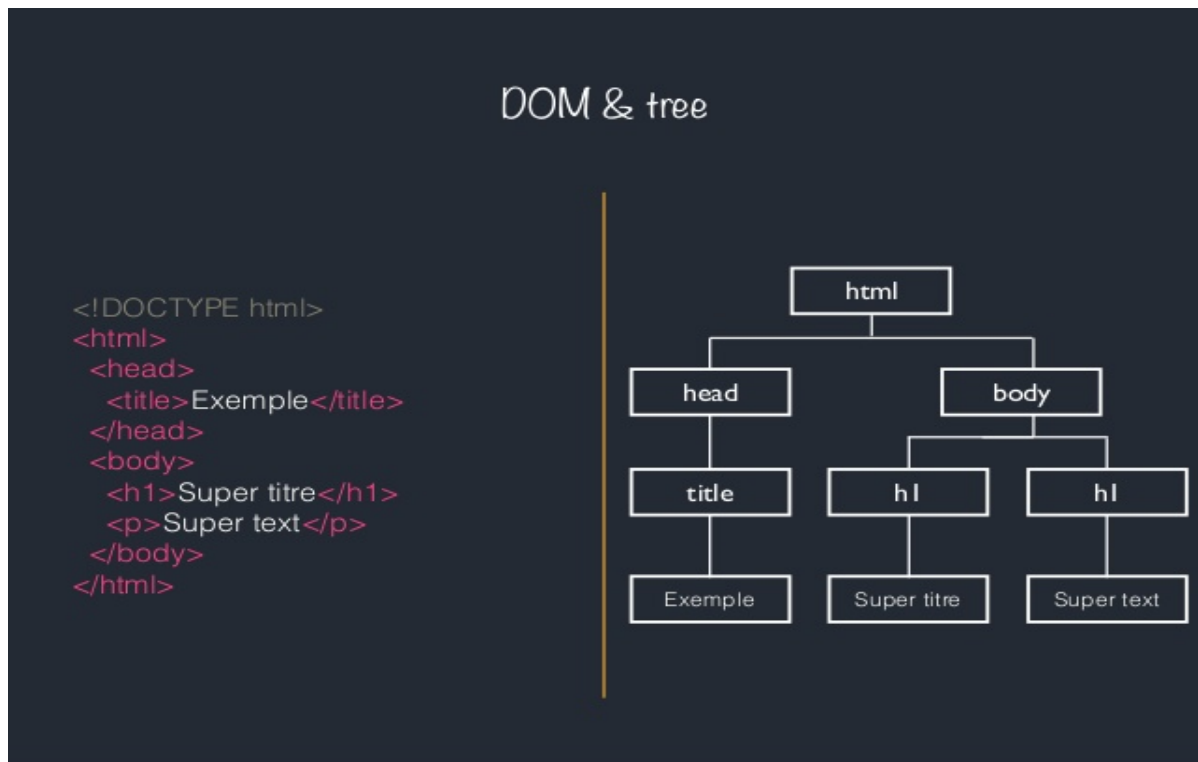


Figure 1 Le DOM et sa représentation arborescente

Donc pour en tirer profit, on l'a utilisé pour ajouter des éléments HTML qui constituent l'interprétation et pour les sauvegarder, pour être plus clair, on donne un exemple :

Si on veut faire un paragraphe contenant une petite description puis une image qui nous amène à un autre site web en cliquant dessus, on doit créer le sous arbre suivant:

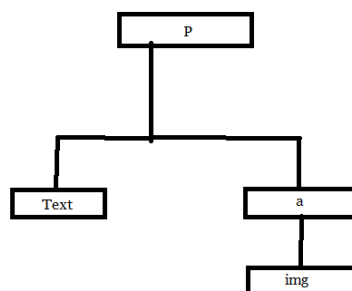


Figure 2 `<p> Text </p>`

- **JavaScript :**

Comme il est dit dans la section précédente, on aura besoin d'un outil qui nous permet de manipuler le contenu du DOM, ceci est possible en utilisant JavaScript, par exemple :

- ✓ La sélection d'un élément de l'arbre DOM : `getElementById`

- ✓ L'ajout d'un élément « élément » à un nœud « node » existant : `node.addElement(element)`

C'est les fonctions fondamentales dans notre méthode et les plus utilisées et ceci constitue la justification du choix de JavaScript et les bibliothèques dérivées comme jQuery et ReactJs qui permettent d'écrire moins de code et de gérer le problème de compatibilité à travers les différents navigateurs.

- **Les méthodes de parsing :**

Après avoir introduire le DOM et les fonctionnalités utilisées en JavaScript, on peut parler des méthodes utilisées dans le parsing.

Notre charte de codification pour cette partie est comme suit : les méthodes commencent par le mot clé parseXXX, par exemple parseImage, parseParagraph ... (vous trouvez le reste avec leurs signatures à l'annexe dans les cahiers de charges).

Chaque méthode se charge à la création d'un nœud (élément) HTML qui correspond à une instruction spécifique du langage D et l'affecter à son nœud père. Comme `<p> ... </p>`, `<table> ... </table>`

- **La récursivité et la résolution des imbrications :**

Notre solution se base sur un algorithme récursif pour créer les éléments HTML (à interpréter) et traiter les imbrications (Des commandes internes dans d'autres commandes), et tout ça d'une façon intuitive.

Supposons qu'on veut interpréter la commande : `-p[Hello -color [world | red]]`

La première chose qu'on rencontre est l'opération de la création d'un paragraphe `-p[...]`. On appelle la méthode du parsing correspondante « `parseParagraph` » mais avant qu'elle termine la création on rencontre l'opération pour colorer le texte en rouge, donc on appelle la méthode « `parseColor` » qui va créer un « `span` » contenant le texte « `world` » stylé en rouge, et dès qu'on termine, la méthode `parseParagraph` ferme le paragraphe qu'elle a commencé avant `parseColor` à la rencontre du crochet ferment.

A la fin on obtient le code HTML suivant :

```
<p> // Appel à parseParagraph
    Hello <span style = "color: red ;" >world</span> // Appel à parseColor
</p>
```

-Ci-dessous un pseudo code du `parseParagraph` (Toutes les opérations sont similaires)

Tanque (on n'a pas parcouru toutes la commande) {

- ✓ Créer un nœud p ; // représentant un paragraphe.
 - Extraire le texte avant de rencontrer une opération, créer un textNode puis l'affecter au nœud p ;
 - Extraire l'opération rencontré –commandexxx[...] , appeler une méthode parseOp qui reconnait la commande xxx puis appel parseXXX
 - Affecter le nœud qui vient d'être crée au nœud p ;
- }

- **L'interprétation en composants graphiques :**

Après la création de tous les nœuds on les affecte à un interpréteur HTML, ayant un engin JavaScript, pour l'affichage graphique des nœuds et la prise en charge des évènements de clicks et de survols de la souris... pour avoir l'interprétation finale.

2. Les allègements portés à l'écriture du code D :

Pour alléger à l'utilisateur l'écriture du code D, on a enlevé quelques restrictions :

- ✓ L'utilisateur n'aura pas besoin d'ajouter le préfixe « get » au opération intermédiaire, ce qui augmente la lisibilité du code et évite les problèmes de syntaxe, donc les deux commandes suivantes sont équivalentes : -ruban[-p[Hello World]] et -ruban[-getp[Hello Wolrd]]
- ✓ Les opérations internes peuvent ne pas être mises dans une opération de sortie, donc l'utilisateur peut écrire -color[Hello Wolrd|red] sans avoir besoin de taper -p[-color[Hello World|red]]

3. La maquette de l'interface graphique.

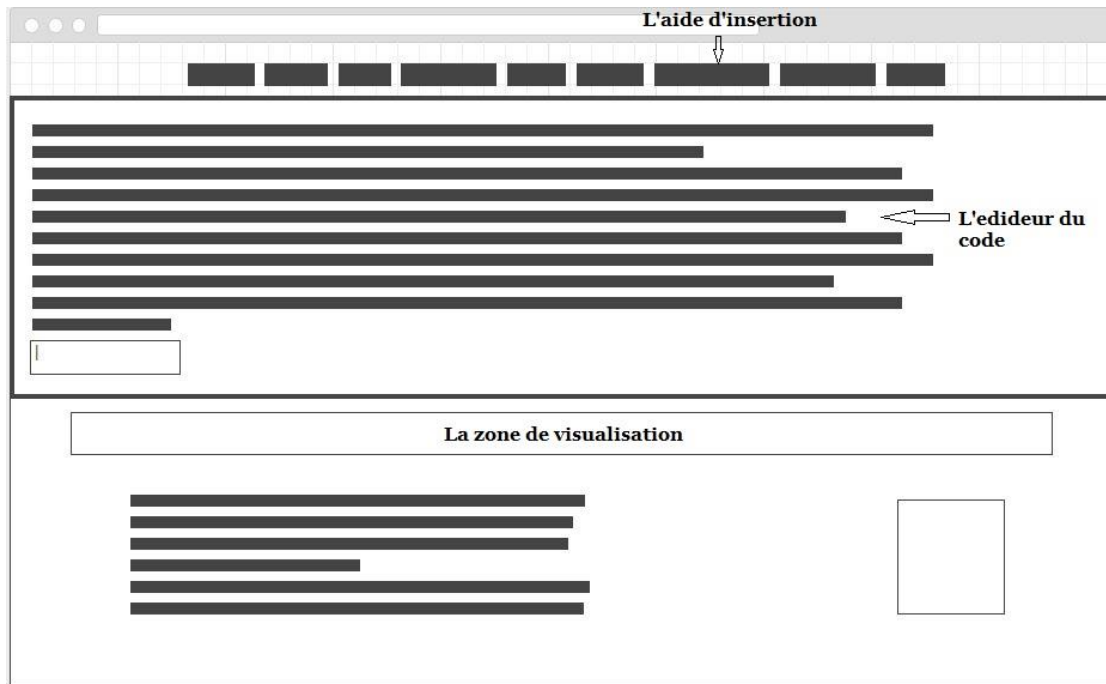


Figure 3 La maquette de l'interface graphique

4. Découpage modulaire

Après la conception, le découpage est devenu évident, chaque fonction du langage D devait être considérée et dans le noyau et dans l'interface en plus de quelques fonctions auxiliaires pour récupérer les paramètres de chaque opération en D et pour synchroniser entre les fonctions réaliser pour interpréter l'ensemble.

Vous trouverez en ANNEXE tous les modules avec leurs descriptions.

IV. La réalisation :

1. La mise en évidence des jeux d'essais

Pour donner une idée sur les objets qui peuvent être créés, on considère le jeu d'essai suivant :

```

1 |ruban[ -couleur[-taille[DANYA PROJECT | 5 ] | #ffffff]]
2
3 -p[Danya Editor, the Editor is similar to a desktop based Text editor with many
4 awesome features including multiple selections, multiple editing, Smart syntax
5 Highlighter for D Markup Language, smart Code autocomplete and easy user interface
6 for inserting code through shortcuts and many other features to discover.
7 ]
8 -observation[Danya Editor, the Editor is similar to a desktop based Text editor with many
9 awesome features including multiple selections, multiple editing, Smart syntax
10 Highlighter for D Markup Language, smart Code autocomplete and easy user interface
11 for inserting code through shortcuts and many other features to discover.
12 ]
13 -alerte[Danya Editor, the Editor is similar to a desktop based Text editor with many
14 awesome features including multiple selections, multiple editing, Smart syntax
15 Highlighter for D Markup Language, smart Code autocomplete and easy user interface
16 for inserting code through shortcuts and many other features to discover.
17 ]
18
19 -slide[540.jpg | /*le lien correspondant*/]
20 -slide[560.jpg | /*le lien correspondant*/]
21 -slide[520.jpg | /*le lien correspondant*/]
22 -slide[550.jpg | /*le lien correspondant*/]
23 -afficherslide
24
25
26 -entetetable[test1|test2|test3]
27 -lignetable[-lvu[ test lien | google.dz ] | -lvu[ test lien | google.dz ] | -lvu[ test lien | google.dz ] ]
28 -lignetable[-lvu[ test lien | google.dz ] | -lvu[ test lien | google.dz ] | -lvu[ test lien | google.dz ] ]
29 -affichertable

```

Figure 4 un jeu d'essai en langage D

Ceci est un simple test qui est loin de présenter tous ce qui peut être réaliser en langage D, il sert juste à donner une idée sur l'interprétation qu'on réalise.

2. La présentation des résultats obtenus

Pour le bout de code précédent, le résultat est :

Visualisation



DANYA PROJECT

Danya Editor, the Editor is similar to a desktop based Text editor with many awesome features including multiple selections, multiple editing, Smart syntax Highlighter for D Markup Language, smart Code autocomplete and easy user interface for inserting code through shortcuts and many other features to discover.



Danya Editor, the Editor is similar to a desktop based Text editor with many awesome features including multiple selections, multiple editing, Smart syntax Highlighter for D Markup Language, smart Code autocomplete and easy user interface for inserting code through shortcuts and many other features to discover.



Danya Editor, the Editor is similar to a desktop based Text editor with many awesome features including multiple selections, multiple editing, Smart syntax Highlighter for D Markup Language, smart Code autocomplete and easy user interface for inserting code through shortcuts and many other features to discover.

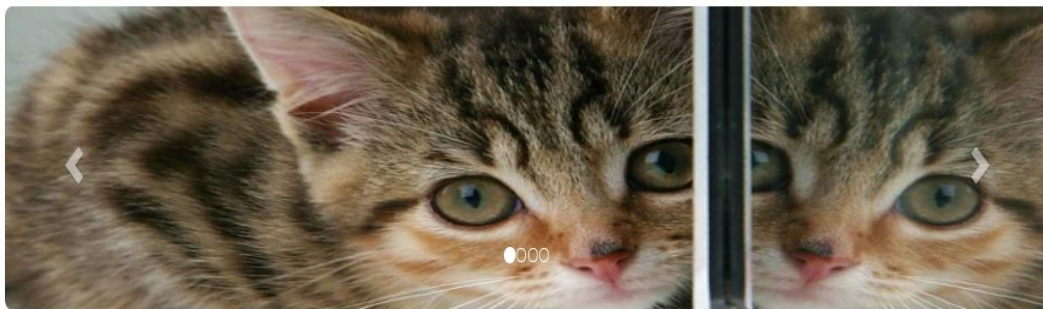


Figure 5 visualisation du jeu d'essai (figure4)

3. L'analyse de certains modules

La fonction `getCommand(commandParts[],command,cursor)` : étant donné un bout de code écrit en langage D contenant un certain nombre d'instructions, le rôle de cette fonction est de retourner le type de l'instruction ainsi que ses différents paramètres séparément dans un tableau (`commandParts`) et d'incrémenter le curseur afin d'avancer dans le code pour connaître le type de l'instruction suivante.

La fonction `parseOperation(operation,commandParts[])` : Prend en paramètre l'opération à interpréter (`operation`) et ses différents paramètres (`commandParts`). Son rôle est d'appeler la fonction la plus adéquate (`parseXXX` pour interpréter l'opération donnée en paramètre) en lui fournissant les paramètres nécessaires pour l'interprétation(`commandParts`).

La fonction `parseParagraph(commandParts[])` : Elle interprète l'instruction (`-p[]` et `-getp[]` du langage D) en HTML. La première case du tableau contient le contenu du paragraphe et éventuellement des instructions imbriquées, la 2^{ème} case étant

facultatif elle contient la position du paragraphe. A noter que presque tous les autres fonctions **parseXXX** repose sur le même principe que **parseParagraph**.

4. Page d'accueil du site :

Notre projet a comme fruit un site web qui contient l'interface avec l'éditeur et tous ce qui était mentionnés et ce qui le sera, ce site est hébergé sur internet (il est donc accessible en ligne) et l'adresse d'accès est : <https://0m3ga-k0der.000webhostapp.com> . Nos tests ont été réalisés en locale sur serveur, la nécessité de ce dernier vient du fait qu'on a des scripts PHP et qu'on dialogue avec une base de données dont on expliquera l'intérêt

Ci-dessous un aperçu de la page d'accueil qui fait savoir de notre site (fonctionnalités, perspectives, membres de l'équipe, outils de réalisation) :

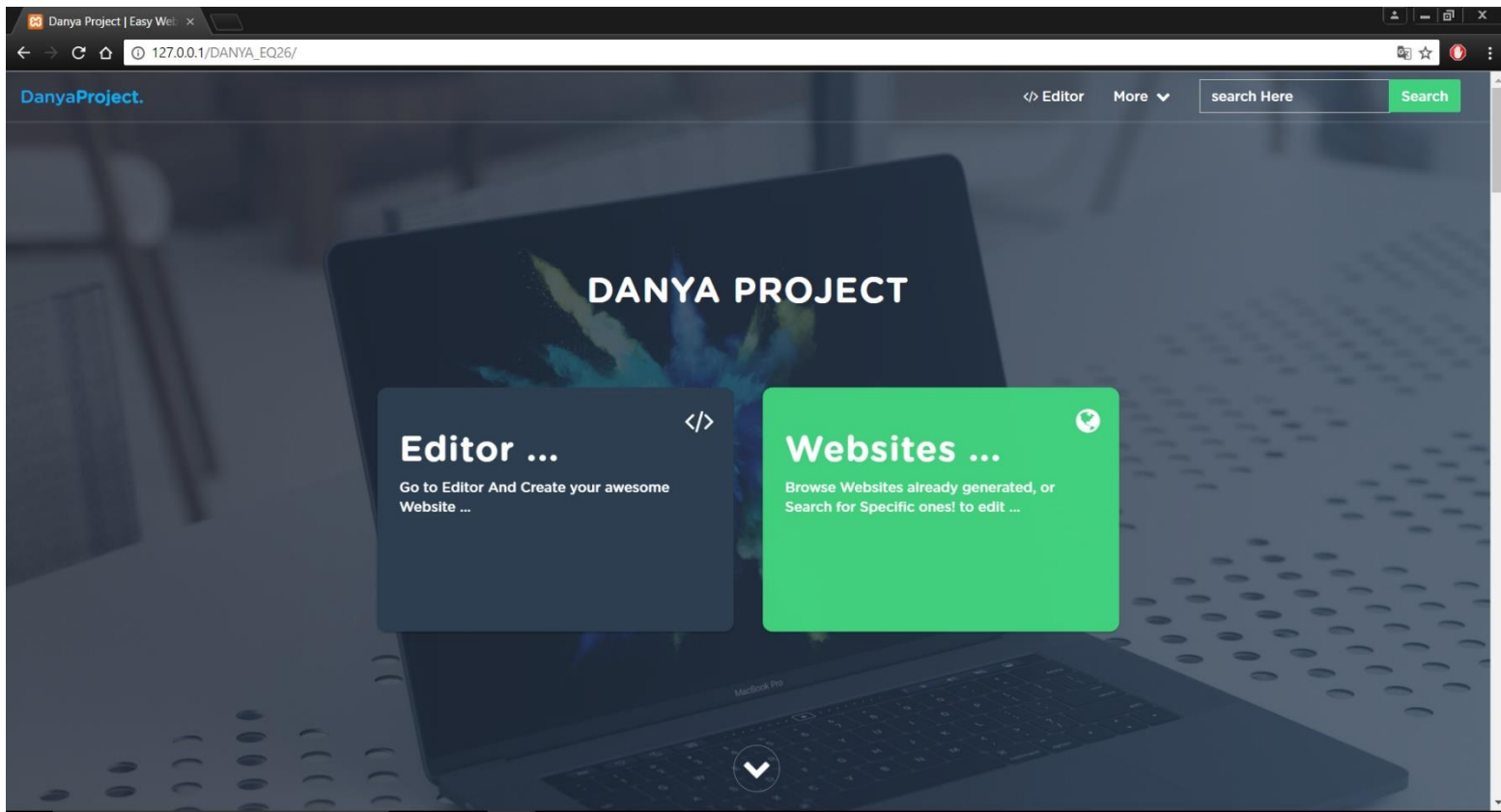


Figure 6 page d'accueil de notre site

5. La réalisation de l'interface graphique

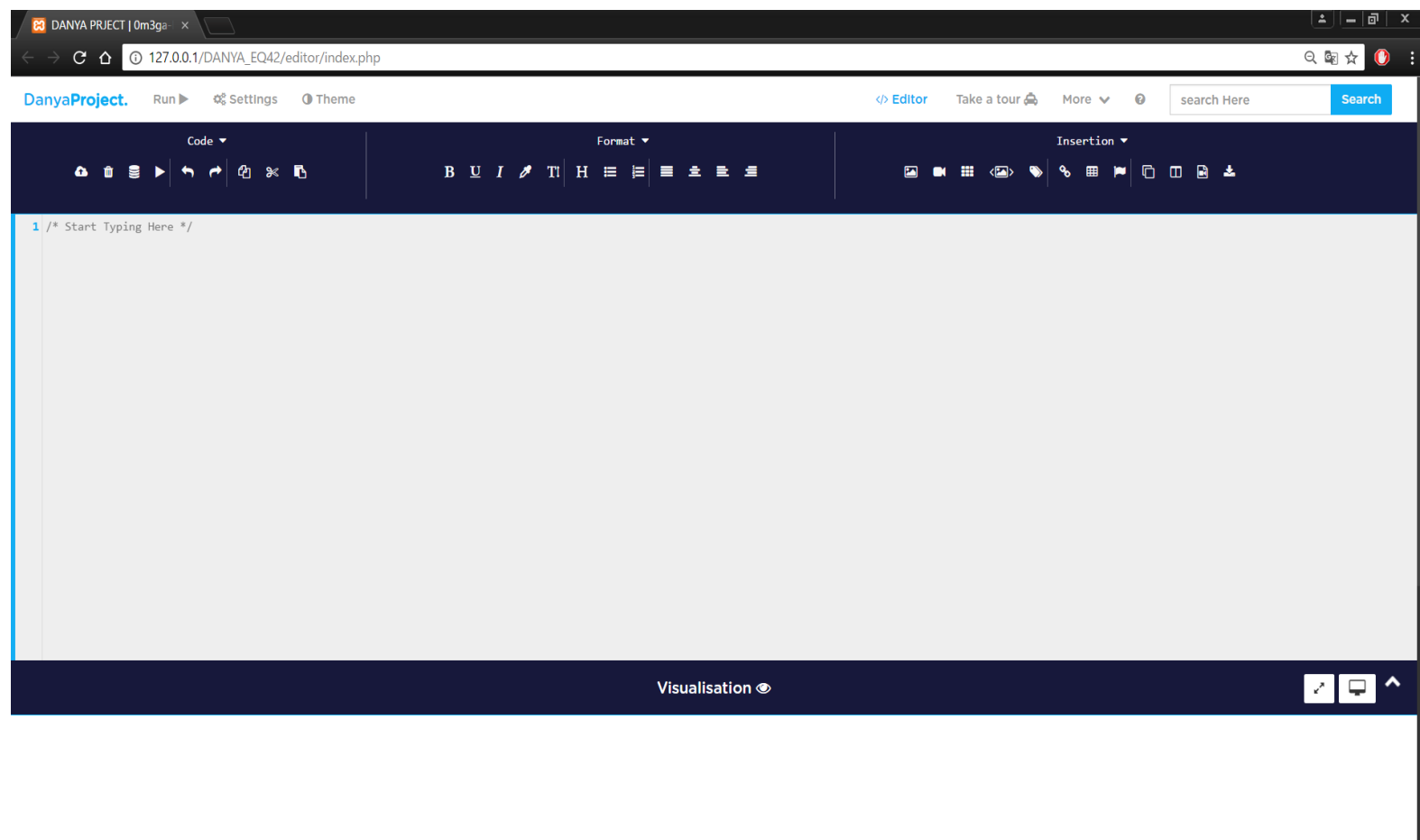


Figure 7 aperçu de l'interface de la plateforme DANYA

La réalisation de l'interface graphique est faite en deux phases essentielles.

La première phase était consacrée à l'implémentation d'un éditeur du code avec toutes les fonctionnalités (le highLighting , prévisualisation des couleurs (figure7), l'auto-complétions, la recherche des sites sauvegardés, copier-coller et plus...), une partie de visualisations pour l'interprétation des différentes commandes et la gestion du côté serveur et base de données.

```
-P[Hello There -couleur[How Are you | #11c000] /*doing*/ | C ]
-slide[560.jpg | /*le lien correspondant*/]
-afficherslide
-wrongCommand[] /* Invalid Command */

/*Command*/ -P[Paragraph|C] -ruban[Danya Project]
/*Numbers*/ 1337 13.77 0xDEADBEEF
/*Hex Color */ #0FACF3 /*Shade of Blue*/
```

Figure 8 illustration du highLiting et la détection d'erreurs de syntaxe

La deuxième phase était consacrée pour la mise en place des outils d'aide à l'insertion des commandes principales (insertion des tableaux, images, ruban, tabs et beaucoup d'autres), l'ajout d'autres styles pour les commandes D comme plusieurs formes des rubans et le nouveau style des sections spéciales (figure8-9) en gardant par défaut les styles définis préalablement par le client lui-même.



Figure 9 nouveau style des sections spéciales



Figure 10 différents styles de ruban

6. Les outils utilisés :

bootStrap, javaScript, JQuery pour la mise en forme du site et la gestion évènementielle. MySql comme base de données, Apache comme serveur et CodeMirror pour l'éditeur du code.

7. La Base de Données :

Notre éditeur est doté de plusieurs fonctionnalités qui nécessitent une base de données ce qui justifie notre choix de l'utiliser, Parmi ces fonctionnalités on peut citer :

- Upload des Images, vidéos et fichiers : Permet à l'utilisateur d'uploader ses images et vidéos vers la base de données évitant ainsi à les re-Uploader une autre fois pour une éventuelle utilisation. (Figure10)
- Sauvegarde des articles : c'est une fonctionnalité qui permet à l'utilisateur de sauvegarder son code dans la base de données pour qu'il puisse le modifier plus tard. (Figure11)
- Rechercher un article : elle offre à l'utilisateur la possibilité de rechercher un article qu'il a déjà écrit et le charger sur l'éditeur pour une éventuelle modification.

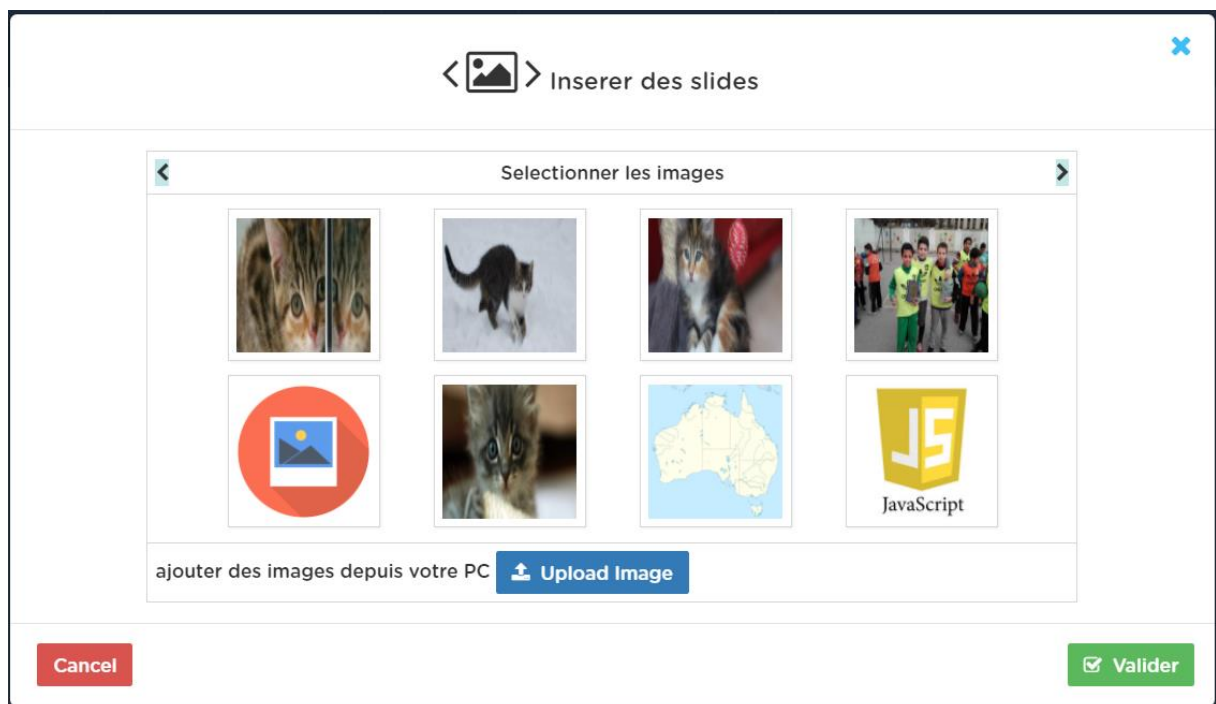
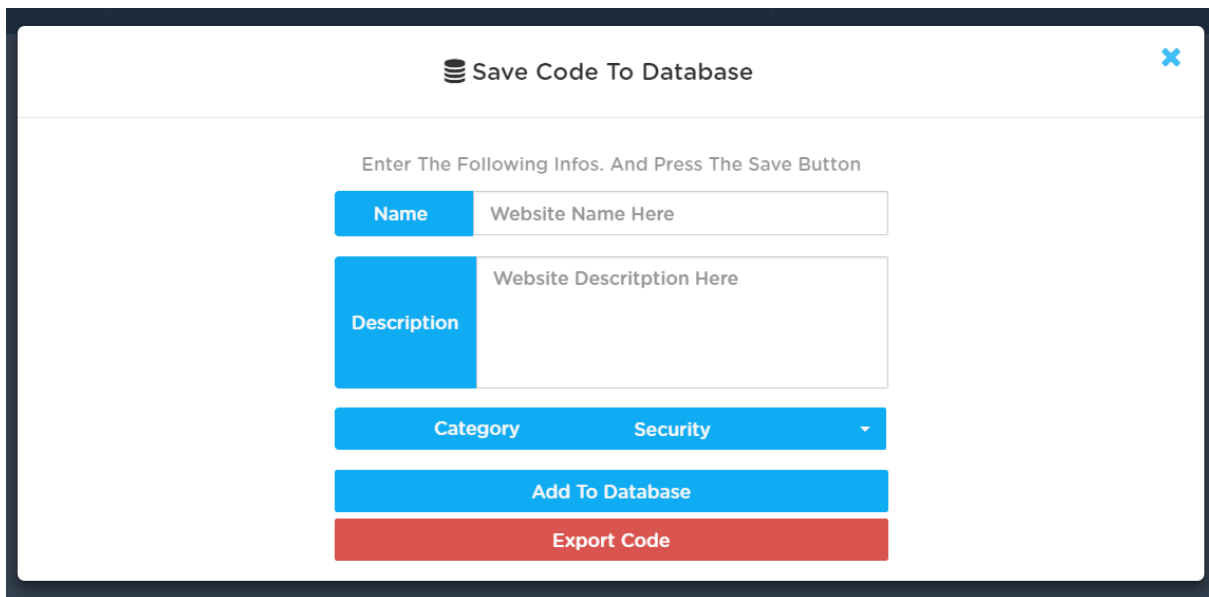


Figure 11 menu d'utilisation et d'upload d'images



Save Code To Database

Enter The Following Infos. And Press The Save Button

Name Website Name Here

Description Website Description Here

Category Security

Add To Database

Export Code

Figure 12 menu de sauvegarde et d'exportation de code D

V. La conclusion :

Dans le cadre de notre projet nous devons construire un éditeur graphique de type «WYSIWYG » permettant à la fois : d'éditer le code D (écrit en langage D) facilement et d'une manière fluide grâce aux différentes fonctionnalités qu'il possède et d'interpréter le code obtenu (en langage D) vers une représentation graphique.

Après avoir bien compris le problème et le travail demandé, nous avons tout de suite commencé à faire une étude de cas qui consiste à découvrir les différents éditeurs Web WYSIWYG existants dans le Web et leurs fonctionnalités ainsi que leurs DESIGN, Cette étude nous a été très bénéfiques car les éditeurs découverts durant cette étude étaient notre source d'inspiration.

Après l'étude de cas, on s'est mis d'accord que notre travail va être divisé en deux grandes phases : La transformation du code D vers une représentation graphique et la réalisation de l'interface graphique.

- **Phase1 :**

Avant de commencer la conception de notre éditeur nous étions obligés à se documenter sur le langage D et d'apprendre toutes ses instructions, sa syntaxe, les différents types d'imbrications tolérées ainsi que les limites du langage, cette étape ne nous a pas posé de problèmes vu la simplicité du langage D.

Après avoir pris le bagage nécessaire en langage D que nous avons jugé nécessaires pour continuer notre travail, nous avons poursuivi notre documentation cette fois-ci sur les langages recommandés par les encadreurs : HTML, CSS, JavaScript, PHP. Cette

étape nous a pris un peu de temps vu que certains membres de l'équipe ne possèdent pas de larges connaissances dans ces langages.

Ensuite nous avons commencé la partie conceptuelle et ce en faisant le découpage modulaire puis on a réparti les modules sur les membres de l'équipe en rédigeant un cahier des charges propre à l'équipe, puis chaque membre a commencé la réalisation des modules mentionnés dans le cahier des charges.

- **Phase2 :**

Durant cette phase nous avons établi un nouveau cahier des charges propre à l'équipe contenant la répartition des tâches (concernant l'interface graphique) sur les membres de l'équipe. Cette phase était la plus importante en terme de temps car elle nous a pris énormément de temps et ce à cause de : la mise en place de nouvelles fonctionnalités qui n'existaient pas auparavant et quelques petits problèmes dues à l'intégration des différentes parties réalisées par chaque membre.

Les propres perspectives

A la fin de ce projet pluridisciplinaire, nous estimons avoir atteint l'objectif principal qui est la construction d'un éditeur graphique de type « WYSIWYG » permettant d'interpréter des articles depuis le langage D vers une représentation graphique.

Cependant l'éditeur peut être encore amélioré. Pour cela nous proposons les améliorations suivantes :

- Ajout de nouvelles instructions dans le langage D (Audio, navBar, style de l'arrière-plan, choisir le font de la police).
- Création d'une application mobile pour l'éditeur.
- Intégrer le travail réalisé cette année avec celui de l'année passée pour avoir un éditeur WYSIWYG complet.

Pour conclure, nous estimons que le travail réalisé est très bénéfique pour l'ensemble des éléments du groupe dans le sens où il nous a permis de développer notre esprit d'équipe, de confronter nos idées pour atteindre l'objectif fixé, d'apprendre à cerner le temps pour être dans les délais impartis, et aussi essentiellement d'enrichir nos connaissances dans les différents langages WEB.

VI. Annexe :

Les cahiers de charges :

Pour mener à bien le projet et en équipe, deviser le travail s'est vu indispensable pour séparer les tâches et travailler en indépendance les uns des autres tout en préparant la voie à l'intégration (normalisation de l'écriture du code et des noms de modules). Pour cela nous avons formulé notre propre cahier de charge en deux temps (noyau, et interface) dans lesquelles on met en évidence le découpage modulaire avec description des modules. Ceci permet, après répartition des tâches, à un membre d'utiliser des modules d'un autre membre de l'équipe sans pour autant que ce dernier ait réalisé le module en question :

- PREMIER CAHIER DE CHARGE :

Ecole Supérieure d'Informatique (ESI) –Projets CPI

Découpage modulaire projet 3 Equipe 26

-Convention d'écriture JavaScript :

-Style camelCase :

- Nous avons adopté pour les déclarations de variables, les noms de fonctions ou de paramètre le style camelCase qui consiste à mettre une majuscule aux mots composants un nom, sauf au premier mot (pour une raison que nous verrons plus bas).
- Pour les variables globales le nom doit commencer par « _ ».

-Commenter votre code :

Il est toujours nécessaire de commenter les fonctions complexes ou les portions de code qui appellent plusieurs fonctions disséminées ailleurs.

Et, lorsque c'est nécessaire donner un exemple en expliquant l'usage des paramètres, mêmes si ceux-ci vous paraissent triviaux.

-Utiliser des noms signifiants :

Donnez toujours des noms représentatifs de l'utilisation que vous allez faire de la variable ou la fonction créée.

-Recommandations :

- Essayer d'utiliser les fonctions prédéfinies en JavaScript, surtout pour le traitement des tableaux et chaînes de caractères :
http://www.w3schools.com/jsref/jsref_obj_array.asp
http://www.w3schools.com/js/js_string_methods.asp
- Il est fortement conseillé de manipuler les Strings comme un tableau, en utilisant .split() et .join()...

- Il y'a des méthodes plus performantes que d'autres, donc une documentation éventuelle est recommandée.
- Chaque membre est libre dans sa méthode d'implémenter une opération du D, mais il est préférable d'inspecter ce qui est fait en DANYA puis s'inspirer.

-Découpage :

Notre projet peut être cerné par les modules suivants :

1. Traitement des chaînes de caractères :

```
-String op:getOperation(Array[] commandParts , String command, int[] cursor){...}
```

Cette fonction doit :

- Retourner une chaîne de caractères qui représente l'opération ex : "p, getp, ruban, couleur..." dans la variable Op.
- Retourner dans le tableau commandParts tous les paramètres contenus dans command. Par exemple : -colonnes[n|P1|P2|...|Pn]=>op="colonnes", commandParts=[P1,P2,...,Pn].
- Cursor est un curseur pour contrôler l'avancement dans la chaîne command, il est déclaré comme un tableau d'une seule case puisqu'il ne y'a pas un passage par variable en JavaScript et on ne veut pas utiliser des objets.

Remarque:

Au cas où on utilise des commentaires dans le code, cette fonction doit les ignorer.

```
-----
-String txt:getText(String commandeBody,int[] cursor){...}
```

Cette fonction doit :

- Retourner tout le texte contenu dans command à partir du cursor jusqu'à la rencontre d'une opération ou bien la fin de la commandeBody.

Attention : une opération doit commencer par un "-" suivi de son nom et un "[" donc l'opération -p() ou bien -Abdo ne doit pas être considérée comme opération.

Remarque : Ces deux fonctions doivent être remises le plutôt possible, car elles sont nécessaires pour commencer les petits tests de l'idée principale.

2. Traitement des opérations :

```
-htmlElement elem:parseOperation(String op,Array[]commandParts[]){...}
```

Cette fonction doit :

- Retourner dans elem un élément Html correspondant à op et construit à partir de commandParts passée en paramètres. Par exemple :
 “-p[Hello –couleur[World] !] donne=>elem===<p>Hello World !</p>”
- La fonction doit chercher dans un répertoire de fonctions et appeler une fonction appropriée a op pour faire un traitement spécial.

-----Op Catégorie Titre-----

-htmlElement elem:parseRuban(Array[] commandParts[]){...}

-htmlElement elem:parseTitle(Array[] commandParts[]){...}

-htmlElement elem:parseTitle_2(Array[] commandParts[]){...}

-htmlElement elem:parseTitle_3(Array[] commandParts[]){...}

Ces fonctions doivent retourner dans elem un élément Html construit à partir de commandParts passée en paramètres.

-----Op Catégorie Paragraphe-----

-htmlElement elem:parseParagraph(Array[]commandParts[]){...}

-----Op Catégorie Style du texte-----

-htmlElement elem:parseBIU(String op,Array[] commandParts[]){...}

Cette fonction doit:

- Retourner un élément Html correspond à toutes combinaison des opération – i,-g et -s. Par exemple: “-gi[...], -i[...], -igs[...]”.
- Remarque:** celui qui va prendre en charge la fonction parsOperation doit faire attention à ce point.

-htmlElement elem:parseColor(Array[]commandParts[]){...}

-htmlElement elem:parseSize(Array[]commandParts[]){...}

-----Op Catégorie Puces et Numerotation-----

-htmlElement elem:parsePuce(Array[]commandParts[]){...}

Remarque:

Il faut se référencier vers la feuille de style du client (Style spécial pour les puces non ordonnées).

```
-htmlElement elem:parseNum(Array[]commandParts[]){...}
```

```
-----Op Catégorie Images-----
```

```
-htmlElement elem:parseImage(Array[]commandParts[]){...}
```

```
-----Op Catégorie Liens et signets-----
```

```
-htmlElement elem:parseLvm(Array[]commandParts[]){...}
```

```
-htmlElement elem:parseLvs(Array[]commandParts[]){...}
```

```
-htmlElement elem:parseLvu(Array[]commandParts[]){...}
```

```
-----Op Catégorie Sections speciales-----
```

```
-htmlElement elem:parseNote(Array[]commandParts[]){...}
```

```
-htmlElement elem:parseAlert(Array[]commandParts[]){...}
```

```
-htmlElement elem:parseObservation(Array[]commandParts[]){...}
```

```
-htmlElement elem:parseRule(Array[]commandParts[]){...}
```

```
-htmlElement elem:parseDef(Array[]commandParts[]){...}
```

```
-htmlElement elem:parseform(Array[]commandParts[]){...}
```

Remarque:

- On recommande de définir une class CSS pour alléger les fonctions du traitement, les styles vont se trouver dans une feuille de style, ceci facilitera l'entretien du site web.
- On recommande aussi de se référencier à la feuille de style du client pour cette partie.

```
-----Op Catégorie Vidéos-----
```

```
-htmlElement elem:parseYoutube(Array[]commandParts[]){...}
```

```
-htmlElement elem:parseIframe(Array[]commandParts[]){...}
```

```
-htmlElement elem:parseVideo(Array[]commandParts[]){...}
```

```
-----Op Catégorie Fichier-----
```

```
-htmlElement elem:parseDownload(Array[]commandParts[]){...}
```

```
-----Op Catégorie Colonnes-----
```

```
-htmlElement elem:parseColumns(Array[]commandParts[]){...}
```

```
-----Op Catégorie Tableaux-----
```

```
-null elem:parseEnteteTable(Array[]commandParts[]){...}
```

```
-null elem:parseLigneTable(Array[]commandParts[]){...}
```

```
-htmlElement elem:parsePrintTable(Array[]commandParts[]){...}
```

Remarque:

- C'est une opération un peu spéciale, les deux fonctions parseEnteteTable et parseLigneTable qui traitent les opérations d'ajout d'informations au tableau ne génèrent aucuns effets sur la visualisation, elles servent à stocker le contenu du tableau dans une variable globale, et celle-ci est utilisée pour afficher le tableau lors de l'appelle de l'opération -affichertable ou gettable en langage D et donc parsePrintTable en javascript. L'emplacement de l'affichage du tableau se fait en fonction de l'emplacement de l'appelle de affichertable ou gettable et non pas des autres fonctions. Par exemple :

Ce code donne :



The screenshot shows a web application interface. At the top, there is a green banner with the text "Abser 1". Below the banner, there is a text input field containing "abser3". To the right of the input field, there is a text output field displaying "abser world!". To the right of the output field, there is a table with two columns: "nom" and "prenom". The table contains two rows of data: "kerris" and "cherief" under the "nom" column, and "nadir" and "yacin" under the "prenom" column.

nom	prenom
kerris	nadir
cherief	yacin

- S'il y a déclaration d'un nouvel entête avant l'affichage d'un tableau déjà présent dans la variable globale, ce dernier sera perdu.
- ATTENTION, lors des appels à parseOperation dans le cas des instructions imbriquées, il faut prendre en considération que parseEnteteTable et parseLigneTable retourne null.

-----Op Catégorie Postes-----

-null elem:[parsePost](#)(Array[]commandParts[]){...}

-htmlElement elem:[parsePrintPost](#)(Array[]commandParts[]){...}

Remarques:

- Mr Cherief Yacine recommande l'utilisation de BootStrap.
- On peut remarquer la similitude des contraintes avec celles des opérations de catégorie tableaux.

-----Op Catégorie Onglets-----

-null elem:[parseTabs](#)(Array[]commandParts[]){...}

-htmlElement elem:[parseAfficherTabs_1](#)(Array[]commandParts[]){...}

-htmlElement elem:[parseAfficherTabs_2](#)(Array[]commandParts[]){...}

Remarques:

- De même, on peut remarquer la similitude des contraintes avec celles des opérations de catégorie tableaux.

-----Op Catégorie Slides-----

-null elem:[parseSlide](#)(Array[]commandParts[]){...}

-htmlElement elem:[parsePrintSlide](#)(Array[]commandParts[]){...}

Remarques :

- Mr Cherief Yacine recommande l'utilisation de BootStrap.
- Là aussi, on peut remarquer la similitude des contraintes avec celles des opérations de catégorie tableaux.

3. Traitement de l'interface web :

Les tâches qui concernent cette patrie sont :

- Réalisation d'un éditeur de texte pour y coder.
- Equiper l'éditeur avec une option Auto Complete.
- Equiper l'éditeur avec le high lighting des mots réservés.
- Permettre à l'utilisateur d'entrer des commentaires.
- Implémentation des icônes d'aide d'insertion du code.
- Permettre à l'utilisateur d'Uploader un fichier contenant du code.

4. Répartition des taches :

Brahim Yacine :

- parseParagraph
- parseTitle, parseTitle_2, parseTitle_3
- parseColor //On n'utilise pas la balise .
- parseOperation.
- parseNote

Djellal Abdou :

- parseRuban
- parseAlerte
- getText //Urgente.
- parseForm
- parseYoutube

Mekhalfa Taki Eddine :

- parseBIU
- parseSize
- parseNum.
- parseRule.
- parseColumns.

Gacem Abderaouf :

- parseImage.
- parseLvm,parseLvu,parseLvs.
- parseDef.
- parseDownload.

Kerris Nadir :

- parseVideo.
- parseIframe.
- parsePuce.
- parseObservation.

Cherief Yassine :

- Réalisation d'un éditeur de texte pour y coder, et une zone de visualisation. //Priorité 1.
- Equiper l'éditeur avec une option Auto Complete.
- Equiper l'éditeur avec le high lighting.
- Implémentation des icônes d'aide d'insertion du code. //Priorité 3
- Permettre à l'utilisateur d'Uploader un fichier contenant du code. //Priorité 2
- getOperation. //Priorité 0

Remarque :

Les modules restants seront affectés ultérieurement.

Ces modules sont :

- L'opération du traitement des tableaux
- Les Slides.
- Les Postes.
- Les Onglets.

- **DEUXIEME CAHIER DE CHARGE :**

Ecole Supérieure d'Informatique (ESI) –Projets CPI**Projet 3 Equipe 26**

L'éditeur de l'interface doit être muni de boutons et des menus d'aide à l'édition de toutes les instructions en langage D, ces éléments seront associés à des événements qui déclenche des fonctions retournant une chaîne de caractères représentant l'instruction correspondante en langage D avec indentation et commentaires, celle-ci sera ensuite insérée dans la zone d'édition du code à interpréter.

La création de boîtes de dialogue et de boutons peut être nécessaire.

Voici les ce qui est imaginé pour les différentes instructions en D :

- -image :

C'est un bouton faisant partie d'une liste déroulante 'insert element' (la liste ne nous concerne pas), en cliquant sur 'image' on s'attend à une boîte de dialogue (créée par

le concerné) qui apparait, elle donnera possibilité de sélectionner l'image à insérer (soit une image existante dans le serveur [une partie de la boîte], soit l'utilisateur parcourt le chemin de l'image à utiliser [deuxième partie de la boîte], elle devra aussi avoir une prévisualisation et des champs : width , height, et position (liste déroulante :gauche [valeur :g] , droite [valeur :d] , centrer [valeur : c]). Dès que l'image est sélectionnée, dans le cas où elle n'est locale, elle sera ajoutée instantanément à la liste des images existantes et sélectionnée pour être insérée. Un boutons 'add' sera présent pour déclencher la fonction (string) `handleImageOp (imageName,width, height,position)`. la fonction retourne

```
{-
image['imageName'/*legende*//*largeur*/width/*hauteur*/height/*position*/
'position']}
```

- -video,-youtube :

C'est semblable à -image, en s'attend à une fenêtre (crée par le concerné) qui donne la possibilité de sélectionner une vidéo interne ou insérer un lien vers youtube et deux champs pour la taille. Le bouton de validation 'add' déclenche la fonction (string) `handleVideoOp(boolYoutube,width,height)` qui retourne `{-video[videoName/*legende*//*largeur*/width/*hauteur*/height]}` ou

`{-youtube[videoURL/*legende*//*largeur*/width/*hauteur*/height]}` celons boolYoutube

- -tableau :

Déjà réalisée par Mekhalifa Taki Eddine, son code sera donné comme exemple. La fonction déclenchée est

(string)`handleTableOp(nbLine,nbColonne,width[nbcolonne],boolAffich)`

- -colonnes :

C'est un boutons qui inclue une zone pour taper le nombre de colonnes souhaité, et une zone de validation qui déclenche (string) `handleColonneOp(nbColonne)`, celle-ci retourne `{-colonnes[nbcolonne/*contenu1*//*contenu2*/]}`

- -puce, -num :

C'est un bouton avec deux zones, la première contient une image significative, lorsqu'on clique dessus les puce (/les num) seront standard. La deuxième zone donne accès à une liste déroulante pour sélectionner des types de puces avec prévisualisation. Le type de puces à utiliser est stocké dans une variable globale. Les deux zones font appelle à la fonction (string) `handlePuceOp()` (/ (string) `handleNumOp()`) qui retourne `{-puce[/*item1*//*item2*//*item3*/]}` ou `{-num[/*item1*//*item2*//*item3*/]}`

- -iframe :

C'est un bouton avec image significative, le click déclenche la fonction `(string)handleIframeOp()` qui retourne `{-iframe[/url*/[/legende*/[/largeur*/[/hauteur*/]]]}`

- -ruban :

Elle fait partie de la liste 'insert element', le click fait apparaître une fenêtre qui offre une visualisation de plusieurs type de ruban, le ruban sélectionné est stocker dans une variable globale. Le bouton de validation déclenche la fonction `(string)handleRubanOp()` qui retourne `{-ruban[/contenu*/]}`

- -titre* :

C'est un bouton qui donne accès à une liste déroulante pour choisir l'un des trois type de titres, la fonction `(string) handleTitleOp(titleType)` devrait se déclencher retournant `{-titre[/contenu*/]}` ou -titre2 ou -titre3 celons titleType.

- -couleur :

C'est un boutons qui donne accès au colorPicker du navigateur (ou personnalisé) et récupère la valeur en hexadécimal de la couleur choisie, la fonction `(string)handleColorOp(color)` retourne `{-couleur[/contenu*/|color]}`

- -taille :

C'est barre glissante qui englobe une plage de tailles en pixels avec un boutons valider qui déclenche la fonction `(string)handleSizeOp(size)` qui retourne `{-taille[/contenu*/|size]}`

- -lv* :

Comme -titre, la sélection permet à la fonction `(string)handleLvOp(linkType)` de retourner `{-lvm{/nom de l'article*/}} ou {-lvs {/nom de l'ancree*/}} ou {-lvu{/url*/}}` celons linkType

- Les sections spéciales :

Un bouton dans la liste 'insert element' qui fait apparaître une fenêtre qui donne des visualisations des différentes sections spéciales, la sélection déclenche la fonction `(string)handlePanelOp(panelType)` qui retourne `{-sectionSpeciale[/contenu*/]}` , sectionSpeciale appartient à {note,definition,alerte,forme,regle,observation}

- -telecharger :

un bouton avec image significative qui fait apparaître une fenêtre qui donne une liste des fichiers à télécharger disponible dans le serveur et un champs pour la taille de l'icône , la sélection déclenche la fonction `(string)handleDownloadOp(NomDuFichier,taille)` qui retourne `{-telecharger['nomDuFichier'|false[/titreDuLien*/|boolTaille]}`

- -slide , -post , -tabs:

Nécessitent plus de réflexions, ce traitement sera discuté ultérieurement.

Remarques :

- Veuillez commentez votre code et respecter l'indentation pour une meilleure lisibilité
- Les fonctions suivantes sont à revisiter :
 - parseNum : le lien des vignettes utilisées doit être choisi selon la variable globale
 - parsePuce : le lien des vignettes utilisées doit être choisi selon la variable globale
 - parseRuban : une construction de plusieurs rubans doit être implémentée avec un switch case selon la variable globale
 - getText : prendre en considération les commentaires
 - getOperation : prendre en considération les commentaires
- Une documentation sur les événements en JavaScript s'impose, éventuellement vous pouvez utiliser jQuery
- Pour les styles des boîtes de dialogues, vous pouvez vous baser sur les styles de Cherief Yassine.
- L'interface vous sera envoyée, vous pouvez tester la réaction de l'éditeur avec vos fonctions en tapant une instruction dont la syntaxe vous sera aussi envoyée

Repartitions des tâches :

1. Brahim Yacine :

- Titre
- Couleur
- Sections spéciales

2. Kerris Nadir :

- Video, youtube
- Puce
- Colonnes

3. Djellal Abderrahmane :

- Ruban
- Iframe

4. Gacem Abderraouf :

- Image
- Taille
- telecharger

5. Mekhalifa Taki Eddine :

- Tableau
- Num
- Iv

Remarque :

Le délai de remise est de 11 jours à partir de la réception du cahier de charge.

Diagramme de Gannt:

Durant la période de travail, le suivi de l'état d'avancement se faisait par le biais d'un logiciel de gestion de projet « GanntProject ».

On a établi un planning de travail sur 10 semaines pour contrôler l'avancement individuel et collectif par rapport à ce qui était prévu.

Le diagramme correspondant au planning est le suivant :

