

# DSL : Rapport Projet 2



*Valentin Campello, Yann Martin d'Escrienne, Lucie Morant, Yohann Tognetti, Tigran Neressian*

## I. Description des langages

Lien vers notre projet : <https://github.com/SI5-I-2021-2022/DSL2>

### A. Domain Model

Notre sujet porte sur la génération de style pour les pages d'encyclopédie (wiki) en tout genre. Cela peut être le style d'une encyclopédie classique comme une encyclopédie d'un jeu fait par des fans (wiki-fandom).

Nous avons donc décidé de mettre en place une liste de composants métier qui sont propres à une encyclopédie, par exemple une table des matières, une section référence, une galerie d'image etc... Nous avons fait le choix de pouvoir définir un style particulier pour chacun de ces composants.

Concernant la logique de nos composants métier, un wiki se compose de différents sujets et comporte une navbar commune à tout l'encyclopédie. Chaque sujet peut avoir différents enfants comme une table des matières, un résumé, des références, des chapitres, etc...

Le Domain Model côté style reprend donc les composants métier mais du point de vue des propriétés de stylisation, et non de contenu. On remarque donc que la classe wiki peut se faire référence à elle-même avec un hover ou un DisplaySize. Cela correspond aux styles alternatifs qui peuvent être le style des éléments lorsqu'ils sont en hover à la souris ou bien à un affichage alternatif avec une certaine taille d'écran (géré par DisplaySize).

Les chapitres comportent également des sous-chapitres. Cela permet d'avoir deux niveaux de chapitres avec des styles complètement différents. Un chapitre peut être donc un chapitre classique ou bien une galerie.

Chaque composant va en général comporter un blockStyle et un contentStyle.

Un blockStyle permet de styliser tout ce qui se rapporte à la "boîte" de contenus et de pouvoir y ajouter par exemple des bords, du margin, une couleur de fond, etc...

Le `contentStyle` permet quant à lui de définir un style pour des composants précis comme un bouton, une image, une table ou même du texte.

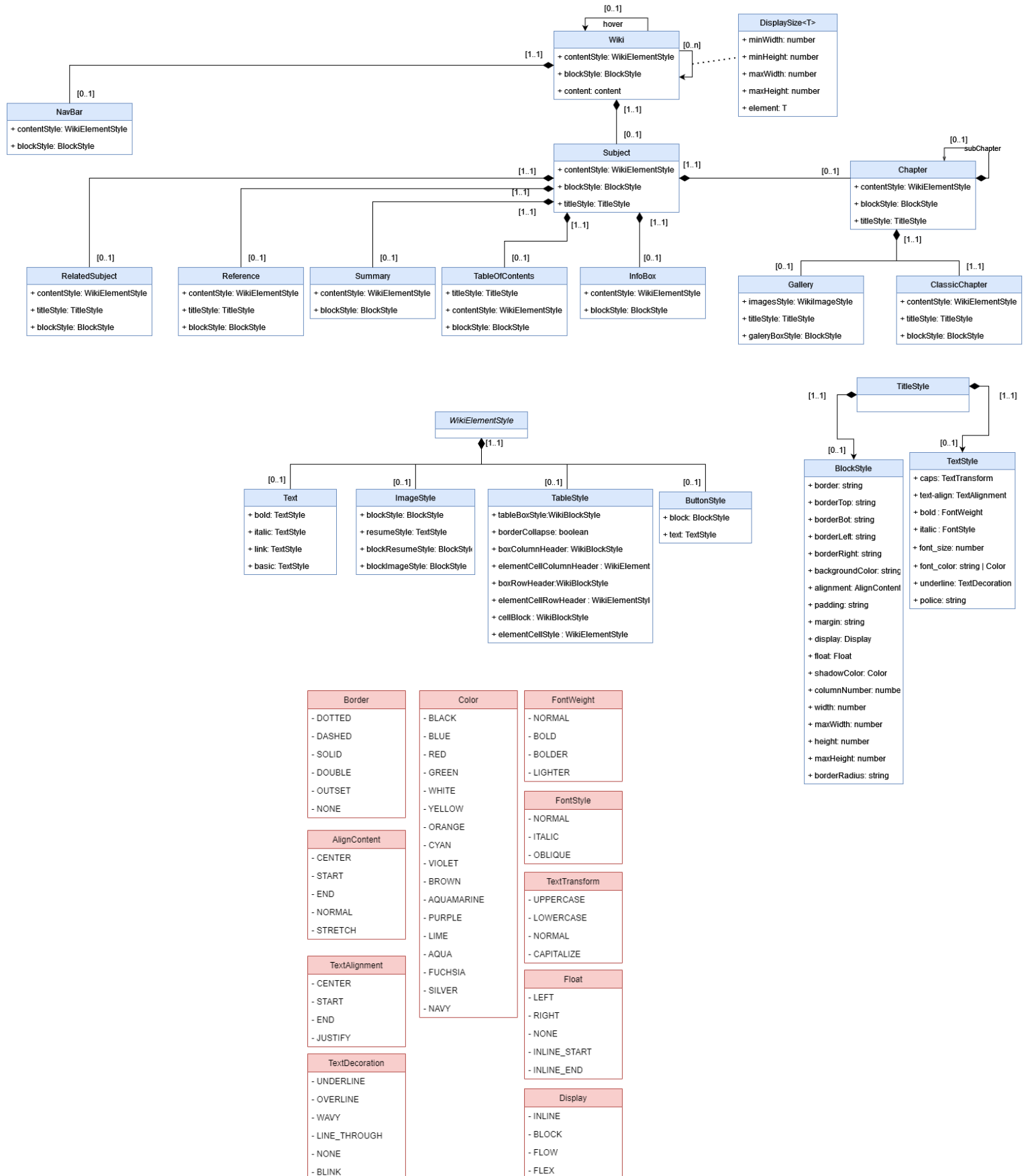
Il est à noter que le texte possède un niveau supérieur de paramétrage car il est possible de définir un style pour le texte normal, mais également pour le texte en gras, en italique ou les liens (gras signifie une balise `<b>`, les liens les balises `<a>` et italic les balises `<i>`). Le style du texte est donc représenté par la classe `TextStyle` qui comporte les propriétés de style propre à du texte.

Il est à noter que nous utilisons de nombreuses énumérations dans notre diagramme. Cela permet de contextualiser et de donner une valeur précise à chaque propriété. Nous détaillerons cela plus précisément dans la partie description de notre langage.

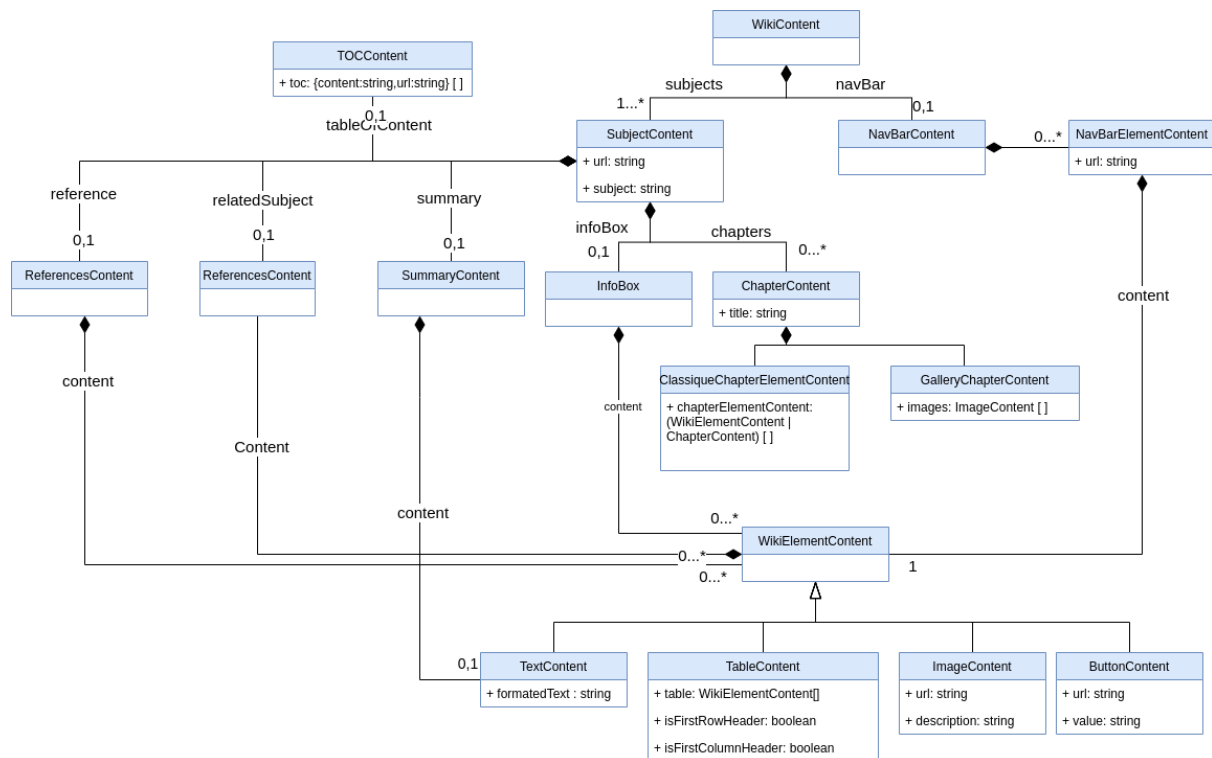
L'intérêt de notre représentation de notre modèle est qu'il est tout à fait possible de définir un style général pour tout le wiki en l'écrivant directement dans la classe `Wiki`, puis d'avoir un rendu plus précis pour un composant particulier en descendant dans l'arborescence de style.

Le `Domain Model` côté content permet de représenter simplement le contenu de façon assez parallèle à ce qui est fait côté style. Nous ne détaillerons pas forcément cette partie car elle n'est pas dans le périmètre du sujet. Cela nous permet juste présenter du contenu (déjà écrit) et de pouvoir le styliser avec notre DSL.

## 1. Notre Domain Model (côté style)



## 2. Notre Domain Model (côté contenu)



## B. Notre syntaxe sous forme BNF

### WikiBuilder:

```

(wikiBuilder=WikiBuilder ''
    (('editOnHoverStyle()'::WikiBuilder)
    | ('editAlternativeDisplayStyle({'minWidth=number','minHeight=number','
        maxWidth=number','maxHeight=number'})::WikiBuilder)
    | ('returnToNormalDisplayStyle()'::WikiBuilder)
    | ('editContentBoxStyle()'::BlockStyleBuilder)
    | ('editSubjectStyle()'::SubjectBuilder)
    | ('editContentStyle()'::WikiElementStyleBuilder)
    | ('editNavBarStyle()'::NavBarBuilder))
) | ( 'WikiBuilder.createWiki()'::WikiBuilder)
    
```

### NavBarBuilder:

```

navBarBuilder=NavBarBuilder ''
    (('editContentBoxStyle()'::BlockStyleBuilder)
    | ('editContentStyle()'::WikiElementStyleBuilder))
    
```

### SubjectBuilder:

```

subjectBuilder=SubjectBuilder ''
    (('editTitleStyle()'::TitleStyleBuilder)
    | ('editContentBoxStyle()'::BlockStyleBuilder)
    | ('editContentStyle()'::WikiElementStyleBuilder)
    | ('editChapterStyle()'::ChapterBuilder)
    
```

```
|('editTableOfContentStyle()':TOCBuilder)
|('editSummaryStyle()':SummaryBuilder)
|('editInfoBoxStyle()':InfoBoxBuilder)
|('editReferenceStyle()':ReferenceBuilder)
|('editRelatedSubjectStyle()':RelatedSubjectBuilder))
```

**TOCBuilder:**

```
tableOfContentBuilder=TOCBuilder ''
(('editTitleStyle()':TitleStyleBuilder)
|('editContentBoxStyle()':BlockStyleBuilder)
|('editContentStyle()':WikiElementStyleBuilder))
```

**SummaryBuilder:**

```
summaryBuilder=SummaryBuilder ''
(('editContentBoxStyle()':BlockStyleBuilder)
|('editContentStyle()':WikiElementStyleBuilder))
```

**InfoBoxBuilder:**

```
infoBoxBuilder=InfoBoxBuilder ''
(('editContentBoxStyle()':BlockStyleBuilder)
|('editContentStyle()':WikiElementStyleBuilder))
```

**ReferenceBuilder:**

```
referenceBuilder=ReferenceBuilder ''
(('editTitleStyle()':TitleStyleBuilder)
|('editContentBoxStyle()':BlockStyleBuilder)
|('editContentStyle()':WikiElementStyleBuilder))
```

**RelatedSubjectBuilder:**

```
relatedSubjectBuilder=RelatedSubjectBuilder ''
(('editTitleStyle()':TitleStyleBuilder)
|('editContentBoxStyle()':BlockStyleBuilder)
|('editContentStyle()':WikiElementStyleBuilder))
```

**ChapterBuilder:**

```
chapterBuilder=ChapterBuilder ''
(('editTitleStyle()':TitleStyleBuilder)
|('editContentBoxStyle()':BlockStyleBuilder)
|('editContentStyle()':WikiElementStyleBuilder)
|('editClassicChapterStyle()':ClassicChapterBuilder)
|('editGalleryStyle()':GalleryBuilder)
|('editSubChapterStyle()':ChapterBuilder))
```

**GalleryBuilder:**

```
galleryBuilder=GalleryBuilder ''
(('editTitleStyle()':TitleStyleBuilder)
|('editGalleryBox()':BlockStyleBuilder)
|('editImagesStyle()':ImageStyleBuilder))
```

```
|('endEditGalleryStyle()'::ChapterBuilder))
```

#### ClassicChapterBuilder:

```
classicChapterBuilder=ClassicChapterBuilder ``  
  (('editTitleStyle()'::TitleStyleBuilder)  
  |('editContentBoxStyle()'::BlockStyleBuilder)  
  |('editContentStyle()'::WikiElementStyleBuilder))
```

#### TitleStyleBuilder:

```
titleStyleBuilder=TitleStyleBuilder<parentBuilder = ParentBuilder> ``  
  (('endTitleEdit()'::ParentBuilder)  
  |('editContentBoxStyle()'::BlockStyleBuilder)  
  |('editContentStyle()'::TextStyleBuilder))
```

#### WikiElementStyleBuilder:

```
wikiElementStyleBuilder=WikiElementStyleBuilder ``  
  (('editTextStyle()'::TextBuilder)  
  |('editImageStyle()'::ImageStyleBuilder)  
  |('editButtonStyle()'::ButtonStyleBuilder)  
  |('editTableStyle()'::TableStyleBuilder))
```

#### ButtonStyleBuilder:

```
buttonStyleBuilder=ButtonStyleBuilder ``  
  (('editContentBoxStyle()'::BlockStyleBuilder)  
  |('editButtonTextStyle()'::TextStyleBuilder)  
  |('endButtonEdit()'::WikiElementStyleBuilder))
```

#### BlockStyleBuilder:

```
blockStyleBuilder=BlockStyleBuilder<parentBuilder = ParentBuilder> ``  
  (('setBorder('width = number``; color? = Color | string``;  
    border? = Border')'::BlockStyleBuilder)  
  |('setBorderRadius('size = number``)'::BlockStyleBuilder)  
  |('setBorderTop('width = number``; color? = Color | string``;  
    border? = Border')'::BlockStyleBuilder)  
  |('setBorderBot('width = number``; color? = Color | string``;  
    border? = Border')'::BlockStyleBuilder)  
  |('setBorderLeft('width = number``; color? = Color | string``;  
    border? = Border')'::BlockStyleBuilder)  
  |('setBorderRight('width = number``; color? = Color | string``;  
    border? = Border')'::BlockStyleBuilder)  
  |('setBackgroundColor('color = Color | string')'::BlockStyleBuilder)  
  |('centerContent()'::BlockStyleBuilder)  
  |('setAlignment('align = AlignContent')'::BlockStyleBuilder)  
  |('setPadding('value = number``; unit? = UnitySize')'::BlockStyleBuilder)  
  |('setPaddingSides('top = number ``; right = number ``; bottom = number ``;  
    left = number``;unit? = UnitySize')'::BlockStyleBuilder)  
  |('setMargin('value = number``; unit? = UnitySize')'::BlockStyleBuilder)  
  |('setMarginSides('top = number ``; right = number ``; bottom = number ``;  
    left = number``;unit? = UnitySize')'::BlockStyleBuilder)  
  |('setWidth('value = number``; unit? = UnitySize')'::BlockStyleBuilder)
```

```
|('setMaxWidth('value = number',' unit? = UnitySize')::BlockStyleBuilder)
|('setHeight('value = number',' unit? = UnitySize')::BlockStyleBuilder)
|('setMaxHeight('value = number',' unit? = UnitySize')::BlockStyleBuilder)
|('displayElementInBlock()::BlockStyleBuilder)
|('displayElementInFlex()::BlockStyleBuilder)
|('displayElementInline()::BlockStyleBuilder)
|('displayElementInFlow()::BlockStyleBuilder)
|('setMaxHeight('float = Float')::BlockStyleBuilder)
|('addShadow('shadowColor? = Color')::BlockStyleBuilder)
|('displayInColumn('columnNumber? = number')::BlockStyleBuilder)
|('endBlockEdit()::ParentBuilder))
```

#### TableStyleBuilder:

```
tableStyleBuilder=TableStyleBuilder ''
  (('editTableBox()::BlockStyleBuilder)
  |('editCellBox()::BlockStyleBuilder)
  |('editCellContent()::WikiElementStyleBuilder)
  |('editHeaderRowCellBox()::BlockStyleBuilder)
  |('editHeaderRowCellContent()::WikiElementStyleBuilder)
  |('editHeaderColumnCellBox()::BlockStyleBuilder)
  |('editHeaderColumnCellContent()::WikiElementStyleBuilder)
  |('setBorderCollapse('value = boolean')::BlockStyleBuilder)
  |('endTableEdit()::WikiElementStyleBuilder))
```

#### ImageStyleBuilder:

```
imageStyleBuilder=ImageStyleBuilder<parentBuilder = ParentBuilder> ''
  (('editBoxStyle()::BlockStyleBuilder)
  |('editAbstractTextStyle()::TextStyleBuilder)
  |('editImageMaxWidthContainer('value = boolean '
                                unit? = UnitySize')::TextStyleBuilder)
  |('editImageMaxHeightImage('value = boolean '
                              unit? = UnitySize')::TextStyleBuilder)
  |('editImageBox()::BlockStyleBuilder)
  |('editImageResumeBox()::BlockStyleBuilder)
  |('endImageEdit()::ParentBuilder))
```

#### TextBuilder:

```
textBuilder=TextBuilder ''
  (('editBoldTextStyle()::TextStyleBuilder)
  |('editItalicTextStyle()::TextStyleBuilder)
  |('editLinkTextStyle()::TextStyleBuilder)
  |('editTextStyle()::TextStyleBuilder)
  |('endTextEdit()::WikiElementStyleBuilder))
```

#### TextStyleBuilder:

```
textStyleBuilder=TextStyleBuilder<parentBuilder = ParentBuilder> ''
  (('capitalizedText('caps? = TextTransform')::TextStyleBuilder)
  |('putInBold('bold? = FontWeight')::TextStyleBuilder)
  |('italicize('italic? = FontWeight')::TextStyleBuilder)
  |('setPolice('police = string')::TextStyleBuilder)
```

```
|('setFontSize('size = number'; unit? = UnitySize')::TextStyleBuilder)
|('setFontColor('color = Color | string')::TextStyleBuilder)
|('underlined('underline? = TextDecoration')::TextStyleBuilder)
|('centerText()::TextStyleBuilder)
|('setTextAlign('align = TextAlignment')::TextStyleBuilder)
|('endTextEdit()::ParentBuilder))
```

AlignContent:

'CENTER' | 'START' | 'END' | 'NORMAL' | 'STRETCH'

Border:

'DOTTED' | 'DASHED' | 'SOLID' | 'DOUBLE' | 'OUTSET' | 'NONE'

Color:

'BLACK' | 'BLUE' | 'RED' | 'GREEN' | 'YELLOW' | 'WHITE' | 'ORANGE' | 'CYAN' | 'GRAY' | 'VIOLET'  
'BROWN' | 'AQUAMARINE' | 'PURPLE' | 'LIME' | 'AQUA' | 'FUCHSIA' | 'NAVY' | 'SILVER'

Display:

'INLINE' | 'BLOCK' | 'FLOW' | 'FLEX'

Float:

'LEFT' | 'NONE' | 'RIGHT' | 'INLINE\_START' | 'INLINE\_END'

FontStyle:

'NORMAL' | 'ITALIC' | 'OBLIQUE'

FontWeight:

'NORMAL' | 'BOLD' | 'BOLDER' | 'LIGHTER'

TextAlignment:

'START' | 'END' | 'CENTER' | 'JUSTIFY'

TextDecoration:

'UNDERLINE' | 'OVERLINE' | 'WAVY' | 'LINE\_THROUGH' | 'NONE' | 'BLINK'

TextTransform:

'UPPERCASE' | 'LOWERCASE' | 'NORMAL' | 'CAPITALIZE'

UnitySize:

'PERCENT' | 'PIXEL' | 'EM' | 'VIEWPORTHEIGHT' | 'VIEWPORTWIDTH' | 'POINT' | 'CENTIMETER'

### C. Description du langage et de son implémentation

Comme on peut le voir à l'aide de la représentation en BNF, notre DSL est un langage interne. Celui-ci se base sur du TypeScript pour son implémentation. Sa génération se fait par chaînage de méthodes, chaque méthode se rapportant à un builder. Chaque builder possède son ensemble de méthodes définies qui mènent chacune vers un autre builder. Cela nous permet de ne proposer que les méthodes appartenant à un élément précis de notre modèle.



En plus de notre kernel, nous avons donc un ensemble de classe de builder possédant chacun une méthode *createModel*, nous permettant de générer l'objet du kernel avec les bons paramètres.

Comme dit dans la description de notre Domain Model, nous utilisons de nombreuses énumérations. Cela nous permet de restreindre l'utilisateur sur les valeurs de certaines propriétés. Nous mettons également en place des méthodes avec des arguments facultatifs. S'ils ne sont pas renseignés, ils prendront alors une valeur par défaut.

De plus, en suivant la logique de notre modèle, notre implémentation de construction du modèle par chaînage permet de descendre de plus en plus précisément dans les éléments et d'y définir les propriétés à chaque étape. Lorsque l'on souhaite finir de modifier un élément et de remonter dans l'arborescence du chaînage (ce qui n'est pas toujours possible), nous avons ajouté des méthodes *endELEMENTEdit* qui permettent de marquer la fin de l'édition de l'élément courant et de remonter au builder parent. Cela se fait à l'aide des sélecteurs du fichier CSS qui sera généré en sortie.

Nous avons également ajouté un vérificateur lors de notre génération. Cela permet de prévenir l'utilisateur lorsqu'il effectue une action qui pourrait compromettre la mise en place du style ou bien celle de la génération.

Pour ce qui est de la vérification, nous nous appuyons tout d'abord sur la vérification des IDE supportant TypeScript. Ils nous permettent d'avertir l'utilisateur lorsqu'il fait des erreurs qui pourraient poser des problèmes lors de la génération, comme de mauvais types dans les valeurs passées aux méthodes ou des erreurs dans le nom de celles-ci.

En plus de ces vérifications, nous avons ajouté des warnings dans la console sur des éléments qui pourraient être sources d'erreurs ou nous semblent illogiques. Par exemple, il est possible dans le chaînage de méthode d'enchaîner les éditions de style de subchapter, ce qui n'a pour effet que de surcharger le style des subchapters à chaque appel. L'utilisateur est alors averti sur la console du potentiel d'erreur de cette surcharge. D'autres avertissements concernent les valeurs passées aux fonctions et qui ne semblent pas pertinentes, comme des tailles négatives pour certains éléments.

#### **D. Génération du code cible**

Pour ce qui est de la génération du code source du style, l'utilisateur doit se placer dans le projet user-repository. Comme tout projet avec npm, il est nécessaire que l'utilisateur possède npm et fasse la commande "npm i" pour installer les dépendances de celui-ci.

Une fois dans ce répertoire, l'utilisateur doit écrire son style à l'aide de notre builder (du nom de WikiBuilder). Nous avons mis en place des exemples de style dans le répertoire style-exemple. En cas d'écriture dans un autre fichier, l'utilisateur doit importer son fichier dans index.ts, qui est le point d'entrée de la génération, et doit le fournir dans le WikiCssGenerator.

Une fois le style écrit, il suffit de faire la commande `npm run generate` qui va automatiquement générer le style du wiki via un fichier CSS qui sera mis en place dans le projet react.

Pour générer le code exécutable, nous avons créé un fichier en TypeScript qui va lire ce que l'utilisateur a écrit dans le langage de notre DSL. La lecture se fait selon la hiérarchie décrite sur notre diagramme de classes, c'est-à-dire que l'on regarde Wiki, puis Subject et Navbar, puis Infobox, etc... Ce générateur fonctionne à l'aide de boucle "if" afin de vérifier si l'élément a été déclaré par l'utilisateur (car celui-ci a la liberté de ne pas vouloir d'un élément dans son encyclopédie). Si c'est le cas, on traduit l'élément et les différents paramètres qui le décrivent en CSS. Lorsque tous les éléments ont été parcourus, un fichier nommé generate.css, qui transcrit toutes les propriétés que l'utilisateur a écrit, est créé dans le projet React afin que l'encyclopédie ait l'aspect voulu.

En parallèle, l'utilisateur peut ouvrir l'application react qui va servir à la visualisation du contenu. Celui-ci se trouve dans le projet wiki-react dans lequel il est aussi nécessaire de faire la commande "npm i" pour installer les dépendances. Pour lancer l'application, la commande "npm run start" lancera automatiquement un serveur de développement à l'url : <http://localhost:3000>

Pendant que l'application tourne, si vous modifiez le style dans l'autre projet, il vous suffit de lancer de nouveau la commande "npm run generate" pour mettre à jour en temps réel le style du wiki. Cela permet une édition du style beaucoup plus simple en hot-reload sans devoir recharger le projet à chaque fois

La partie contenu ne doit pas être modifiée par le designer. Pour des raisons de simplicité et de **hors scope**, le contenu affiché dans l'application est renseigné dans user-repository/model/exemple/, qui est dupliqué dans l'application react à l'aide du script shell "load-model.sh", qui se trouve à la racine du projet. Dans le cadre du déploiement de la solution, le contenu serait retiré de cet endroit pour être stocké dans une base de données par exemple. Toutes modifications du contenu dans le projet user-repository ne sera pas effectuée automatiquement dans l'application react et il sera nécessaire de redémarrer l'application pour avoir le nouveau contenu, en plus de devoir lancer le script "load-model.sh".

## II. Scénarios implémentés

Il est possible de retrouver le code de notre DSL pour chacun de ses styles dans le dossier "style-exemple" dans le projet user-repository. Il est également possible de générer le fichier de style pour le projet react en important le style dans le WikiCssGenerator du fichier index.ts (du projet userRepository) comme dans l'exemple ci-dessous :

```
10
11  const cssGenerator = new WikiCssGenerator();
12  cssGenerator.generateCssFile(SlayTheSpireModel)
13
```

### A. Style Wiki Simple

[Lien du scénario](#)

Le but du scénario est de reproduire un wiki simple, ressemblant au style du wiki Larousse.

## Tropidophorus matsuii

- endémique
- originale
- Masafumi Matsui

*Tropidophorus matsuii* est une [espèce](#) de [sauriens](#) de la famille des [Scincidae](#).

### Table des matières

1. [Répartition](#)
2. [Étymologie](#)
3. [Publication originale](#)
4. [Liens externes](#)

### Répartition

Cette espèce est [endémique](#) de l'Est de la [Thaïlande](#).

### Étymologie

Cette espèce est nommée en l'honneur de [Masafumi Matsui](#).

### Publication originale

Cette espèce est [endémique](#) de l'Est de la [Thaïlande](#).



un lézard

[wikipedia page](#)

Dans ce scénario, nous avons aussi ajouté un style onHover alternatif simpliste lorsque l'on passe sur une image présente dans une galerie.

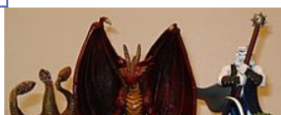
### Galerie



Dungeons & Dragons miniature figures. The grid mat underneath uses one-inch squares.



Dungeons & Dragons miniature figures. The grid mat underneath uses one-inch squares.




## B. Style Wiki Fandom Slay The Spire

### [Lien du scénario](#)


Ce scénario permet de montrer un exemple d'un rendu visuel que l'on peut obtenir avec une stylisation plus poussée. Il est inspiré du wiki fanDom du jeu SlayTheSpire. Il montre le fait de mettre

en place un style général dans le wiki puis des styles plus précis comme pour les sous-chapitres ou les références et sujets liés.


<div> <div></div> <div> <p><b>1.</b> matsui</p> <p><b>2.</b> Mésopotamie</p> <p><b>3.</b> Dungeons &amp; Dragons</p> </div> </div>
<div>Tropidophorus matsuii</div> <ul style="list-style-type: none"> <li><div><span></span> endémique</div></li> <li><div><span></span> originale</div></li> <li><div><span></span> Matsui's Matsui</div></li> </ul> <div>Tropidophorus matsuii est une <b>espèce de sauriens</b> de la famille des <b>Scolecidae</b>.</div> <div> <div>Table des matières</div> <div> <ol style="list-style-type: none"><li>Répartition</li><li>Étymologie</li><li>Publication originale</li><li>Liens externes</li></ol> </div> </div> <div>Répartition</div> <div>Cette espèce est <b>endémique</b> de l'Est de la <b>Thaïlande</b>.</div> <div>Étymologie</div> <div>Cette espèce est nommée en l'honneur de <b>Matsui's Matsui</b>.</div> <div>Publication originale</div> <div>Cette espèce est <b>endémique</b> de l'Est de la <b>Thaïlande</b></div> <div>  <p>en haut</p> </div> <div> <div>voir aussi page</div> <div>Un sous chapitre</div> <div>Cette espèce est <b>endémique</b> de l'Est de la <b>Thaïlande</b></div> <div>voir aussi page</div> </div>

Other influences include the works of Robert E. Howard, Edgar Rice Burroughs, A. Merritt, H. P. Lovecraft, Fritz Leiber, L. Sprague de Camp, Fletcher Pratt, Roger Zelazny, and Michael Moorcock.<sup>[22]</sup> Monsters, spells, and magic items used in the game have inspired by hundreds of individual works such as *A.E. Van Vogt's* "Black Destroyer", *Cornel* (the *Displacer Beast*), *Lewis Carroll's* "Jabberwocky", *Cornel* sword and the *Book of Genesis* (the clerical spell "Blade Barrier" was inspired by the "*flaming sword*" which turned every way" at the gates of *Eden*).<sup>[23]</sup>


**Gallerie**




Dungeons & Dragons miniature figures: the grid mat underneath uses one-inch squares.




Dungeons & Dragons miniature figures: the grid mat underneath uses one-inch squares.




Dungeons & Dragons miniature figures: the grid mat underneath uses one-inch squares.




Dungeons & Dragons miniature figures: the grid mat underneath uses one-inch squares.




Dungeons & Dragons miniature figures: the grid mat underneath uses one-inch squares.




Dungeons & Dragons miniature figures: the grid mat underneath uses one-inch squares.




Dungeons & Dragons miniature figures: the grid mat underneath uses one-inch squares.




Dungeons & Dragons miniature figures: the grid mat underneath uses one-inch squares.




Dungeons & Dragons miniature figures: the grid mat underneath uses one-inch squares.




Dungeons & Dragons miniature figures: the grid mat underneath uses one-inch squares.




Dungeons & Dragons miniature figures: the grid mat underneath uses one-inch squares.




Dungeons & Dragons miniature figures: the grid mat underneath uses one-inch squares.



Dungeons & Dragons miniature figures: the grid mat underneath uses one-inch squares.



Dungeons & Dragons miniature figures: the grid mat underneath uses one-inch squares.



Dungeons & Dragons miniature figures: the grid mat underneath uses one-inch squares.

**Voir aussi**

- Official website
- Dungeons & Dragons et Culte
- OT:Box* (June 20, 2013) *"Dungeons & Dragons and the Influence of Tabletop RPGs"*. PBS.

**Références**

<sup>[1]</sup> *"DND Basic Set"*. Rulebooks and Sets. occum.com. *Archivé* from the original on June 24, 2011. Retrieved May 16, 2013.

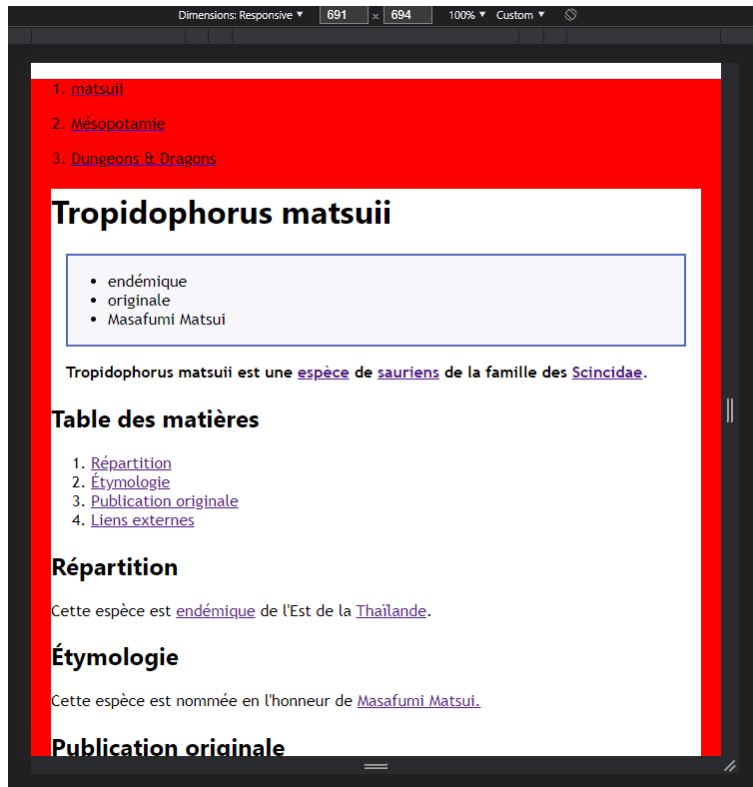
<sup>[2]</sup> *"DND: The 'What does that stand for' list"*. Geek Nostalgia. February 18, 2019. Retrieved February 26, 2020.

<sup>[3]</sup>  Jump up to: <sup>[a]</sup> <sup>[b]</sup> <sup>[c]</sup> <sup>[d]</sup> <sup>[e]</sup> <sup>[f]</sup> <sup>[g]</sup> <sup>[h]</sup> <sup>[i]</sup> <sup>[j]</sup> <sup>[k]</sup> <sup>[l]</sup> <sup>[m]</sup> <sup>[n]</sup> <sup>[o]</sup> <sup>[p]</sup> <sup>[q]</sup> <sup>[r]</sup> <sup>[s]</sup> <sup>[t]</sup> <sup>[u]</sup> <sup>[v]</sup> <sup>[w]</sup> <sup>[x]</sup> <sup>[y]</sup> <sup>[z]</sup> <sup>[aa]</sup> <sup>[ab]</sup> <sup>[ac]</sup> <sup>[ad]</sup> <sup>[ae]</sup> <sup>[af]</sup> <sup>[ag]</sup> <sup>[ah]</sup> <sup>[ai]</sup> <sup>[aj]</sup> <sup>[ak]</sup> <sup>[al]</sup> <sup>[am]</sup> <sup>[an]</sup> <sup>[ao]</sup> <sup>[ap]</sup> <sup>[aq]</sup> <sup>[ar]</sup> <sup>[as]</sup> <sup>[at]</sup> <sup>[au]</sup> <sup>[av]</sup> <sup>[aw]</sup> <sup>[ax]</sup> <sup>[ay]</sup> <sup>[az]</sup> <sup>[ba]</sup> <sup>[bb]</sup> <sup>[bc]</sup> <sup>[bd]</sup> <sup>[be]</sup> <sup>[bf]</sup> <sup>[bg]</sup> <sup>[bh]</sup> <sup>[bi]</sup> <sup>[bj]</sup> <sup>[bk]</sup> <sup>[bl]</sup> <sup>[bm]</sup> <sup>[bn]</sup> <sup>[bo]</sup> <sup>[bp]</sup> <sup>[bq]</sup> <sup>[br]</sup> <sup>[bs]</sup> <sup>[bt]</sup> <sup>[bu]</sup> <sup>[bv]</sup> <sup>[bw]</sup> <sup>[bx]</sup> <sup>[by]</sup> <sup>[bz]</sup> <sup>[ca]</sup> <sup>[cb]</sup> <sup>[cc]</sup> <sup>[cd]</sup> <sup>[ce]</sup> <sup>[cf]</sup> <sup>[cg]</sup> <sup>[ch]</sup> <sup>[ci]</sup> <sup>[cj]</sup> <sup>[ck]</sup> <sup>[cl]</sup> <sup>[cm]</sup> <sup>[cn]</sup> <sup>[co]</sup> <sup>[cp]</sup> <sup>[cq]</sup> <sup>[cr]</sup> <sup>[cs]</sup> <sup>[ct]</sup> <sup>[cu]</sup> <sup>[cv]</sup> <sup>[cw]</sup> <sup>[cx]</sup> <sup>[cy]</sup> <sup>[cz]</sup> <sup>[da]</sup> <sup>[db]</sup> <sup>[dc]</sup> <sup>[dd]</sup> <sup>[de]</sup> <sup>[df]</sup> <sup>[dg]</sup> <sup>[dh]</sup> <sup>[di]</sup> <sup>[dj]</sup> <sup>[dk]</sup> <sup>[dl]</sup> <sup>[dm]</sup> <sup>[dn]</sup> <sup>[do]</sup> <sup>[dp]</sup> <sup>[dq]</sup> <sup>[dr]</sup> <sup>[ds]</sup> <sup>[dt]</sup> <sup>[du]</sup> <sup>[dv]</sup> <sup>[dw]</sup> <sup>[dx]</sup> <sup>[dy]</sup> <sup>[dz]</sup> <sup>[ea]</sup> <sup>[eb]</sup> <sup>[ec]</sup> <sup>[ed]</sup> <sup>[ee]</sup> <sup>[ef]</sup> <sup>[eg]</sup> <sup>[eh]</sup> <sup>[ei]</sup> <sup>[ej]</sup> <sup>[ek]</sup> <sup>[el]</sup> <sup>[em]</sup> <sup>[en]</sup> <sup>[eo]</sup> <sup>[ep]</sup> <sup>[eq]</sup> <sup>[er]</sup> <sup>[es]</sup> <sup>[et]</sup> <sup>[eu]</sup> <sup>[ev]</sup> <sup>[ew]</sup> <sup>[ex]</sup> <sup>[ey]</sup> <sup>[ez]</sup> <sup>[fa]</sup> <sup>[fb]</sup> <sup>[fc]</sup> <sup>[fd]</sup> <sup>[fe]</sup> <sup>[ff]</sup> <sup>[fg]</sup> <sup>[fh]</sup> <sup>[fi]</sup> <sup>[fj]</sup> <sup>[fk]</sup> <sup>[fl]</sup> <sup>[fm]</sup> <sup>[fn]</sup> <sup>[fo]</sup> <sup>[fp]</sup> <sup>[fq]</sup> <sup>[fr]</sup> <sup>[fs]</sup> <sup>[ft]</sup> <sup>[fu]</sup> <sup>[fv]</sup> <sup>[fw]</sup> <sup>[fx]</sup> <sup>[fy]</sup> <sup>[fz]</sup> <sup>[ga]</sup> <sup>[gb]</sup> <sup>[gc]</sup> <sup>[gd]</sup> <sup>[ge]</sup> <sup>[gf]</sup> <sup>[gg]</sup> <sup>[gh]</sup> <sup>[gi]</sup> <sup>[gj]</sup> <sup>[gk]</sup> <sup>[gl]</sup> <sup>[gm]</sup> <sup>[gn]</sup> <sup>[go]</sup> <sup>[gp]</sup> <sup>[gq]</sup> <sup>[gr]</sup> <sup>[gs]</sup> <sup>[gt]</sup> <sup>[gu]</sup> <sup>[gv]</sup> <sup>[gw]</sup> <sup>[gx]</sup> <sup>[gy]</sup> <sup>[gz]</sup> <sup>[ha]</sup> <sup>[hb]</sup> <sup>[hc]</sup> <sup>[hd]</sup> <sup>[he]</sup> <sup>[hf]</sup> <sup>[hg]</sup> <sup>[hh]</sup> <sup>[hi]</sup> <sup>[hj]</sup> <sup>[hk]</sup> <sup>[hl]</sup> <sup>[hm]</sup> <sup>[hn]</sup> <sup>[ho]</sup> <sup>[hp]</sup> <sup>[hq]</sup> <sup>[hr]</sup> <sup>[hs]</sup> <sup>[ht]</sup> <sup>[hu]</sup> <sup>[hv]</sup> <sup>[hw]</sup> <sup>[hx]</sup> <sup>[hy]</sup> <sup>[hz]</sup> <sup>[ia]</sup> <sup>[ib]</sup> <sup>[ic]</sup> <sup>[id]</sup> <sup>[ie]</sup> <sup>[if]</sup> <sup>[ig]</sup> <sup>[ih]</sup> <sup>[ii]</sup> <sup>[ij]</sup> <sup>[ik]</sup> <sup>[il]</sup> <sup>[im]</sup> <sup>[in]</sup> <sup>[io]</sup> <sup>[ip]</sup> <sup>[iq]</sup> <sup>[ir]</sup> <sup>[is]</sup> <sup>[it]</sup> <sup>[iu]</sup> <sup>[iv]</sup> <sup>[iw]</sup> <sup>[ix]</sup> <sup>[iy]</sup> <sup>[iz]</sup> <sup>[ja]</sup> <sup>[jb]</sup> <sup>[jc]</sup> <sup>[jd]</sup> <sup>[je]</sup> <sup>[jf]</sup> <sup>[jg]</sup> <sup>[jh]</sup> <sup>[ji]</sup> <sup>[jj]</sup> <sup>[jk]</sup> <sup>[jl]</sup> <sup>[jm]</sup> <sup>[jn]</sup> <sup>[jo]</sup> <sup>[jp]</sup> <sup>[jq]</sup> <sup>[jr]</sup> <sup>[js]</sup> <sup>[jt]</sup> <sup>[ju]</sup> <sup>[jv]</sup> <sup>[jw]</sup> <sup>[jx]</sup> <sup>[jy]</sup> <sup>[jz]</sup> <sup>[ka]</sup> <sup>[kb]</sup> <sup>[kc]</sup> <sup>[kd]</sup> <sup>[ke]</sup> <sup>[kf]</sup> <sup>[kg]</sup> <sup>[kh]</sup> <sup>[ki]</sup> <sup>[kj]</sup> <sup>[kk]</sup> <sup>[kl]</sup> <sup>[km]</sup> <sup>[kn]</sup> <sup>[ko]</sup> <sup>[kp]</sup> <sup>[kq]</sup> <sup>[kr]</sup> <sup>[ks]</sup> <sup>[kt]</sup> <sup>[ku]</sup> <sup>[kv]</sup> <sup>[kw]</sup> <sup>[kx]</sup> <sup>[ky]</sup> <sup>[kz]</sup> <sup>[la]</sup> <sup>[lb]</sup> <sup>[lc]</sup> <sup>[ld]</sup> <sup>[le]</sup> <sup>[lf]</sup> <sup>[lg]</sup> <sup>[lh]</sup> <sup>[li]</sup> <sup>[lj]</sup> <sup>[lk]</sup> <sup>[ll]</sup> <sup>[lm]</sup> <sup>[ln]</sup> <sup>[lo]</sup> <sup>[lp]</sup> <sup>[lq]</sup> <sup>[lr]</sup> <sup>[ls]</sup> <sup>[lt]</sup> <sup>[lu]</sup> <sup>[lv]</sup> <sup>[lw]</sup> <sup>[lx]</sup> <sup>[ly]</sup> <sup>[lz]</sup> <sup>[ma]</sup> <sup>[mb]</sup> <sup>[mc]</sup> <sup>[md]</sup> <sup>[me]</sup> <sup>[mf]</sup> <sup>[mg]</sup> <sup>[mh]</sup> <sup>[mi]</sup> <sup>[mj]</sup> <sup>[mk]</sup> <sup>[ml]</sup> <sup>[mn]</sup> <sup>[mo]</sup> <sup>[mp]</sup> <sup>[mq]</sup> <sup>[mr]</sup> <sup>[ms]</sup> <sup>[mt]</sup> <sup>[mu]</sup> <sup>[mv]</sup> <sup>[mw]</sup> <sup>[mx]</sup> <sup>[my]</sup> <sup>[mz]</sup> <sup>[na]</sup> <sup>[nb]</sup> <sup>[nc]</sup> <sup>[nd]</sup> <sup>[ne]</sup> <sup>[nf]</sup> <sup>[ng]</sup> <sup>[nh</sup>

### C. Style Wiki Responsive

[Lien du scénario](#)

Le but de ce scénario est de montrer que l'on peut affecter des styles différents suivant la taille de l'écran de l'utilisateur.

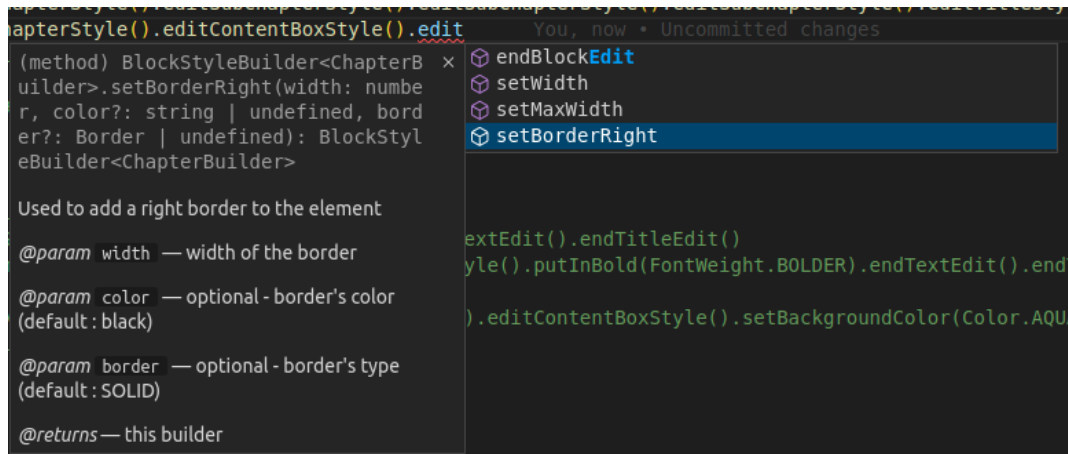


### III. Analyse critique du projet

#### A. Implémentation du DSL pour le cas d'utilisation de l'UXifier

Lors de notre développement, nous avons fait en sorte que notre DSL soit le plus simple à utiliser pour le designer.

Premièrement, nous mettons en place des méthodes les plus explicites possible à travers notre builder. Pour les utilisateurs débutant, nous avons également ajouté une documentation sur chacune de nos méthodes décrivant chaque paramètre que la méthode peut attendre. Elle se présente comme ci-dessous :



Concernant les paramètres, comme dit précédemment, beaucoup d'entre eux sont optionnels. Cela permet à l'utilisateur d'arriver rapidement à ce qu'il veut quand il désire quelque chose de simple en omettant les paramètres tout en lui permettant de définir un comportement plus précis dans la stylisation en les renseignant. Un exemple concret est la méthode `putInBold` qui permet de mettre plus ou moins en gras le texte. Celle-ci peut ne prendre aucun paramètre et elle aura alors la valeur en gras par défaut. Dans le cas d'un désir d'une valeur de mise en gras plus fine, libre à l'utilisateur de le renseigner. Cela est également le cas pour les différentes unités de mesure comme un pourcentage de la page, un pixel, font, ect, qui sont optionnels et dont la valeur par défaut change en fonction du contenu (mesure en point pour un `fontSize` et pixel pour des marges par exemple).

L'utilisation de nombreuses énumérations nous permet également de restreindre l'utilisateur sur les valeurs que celui-ci peut définir dans les paramètres de nos méthodes. Mais plus que dans le but de le restreindre, ces énumérations sont là pour guider l'utilisateur dans les différentes stylisations qu'il peut mettre en place pour ladite propriété.

Pour le cas de certaines méthodes comme celle liée à la couleur, nous offrons une palette par défaut mais donnons également la possibilité de donner une couleur sous format hexadécimal ("`#565252`").

En plus du style de la boîte de contenu et du contenu, si l'utilisateur manipule un élément avec un titre, il lui est possible de faire le style de ce titre autant au niveau du texte qu'au niveau de la boîte de contenu du titre. Cela lui donne encore plus de possibilité de personnalisation.

Enfin, nous offrons des méthodes alternatives éditant la même propriété, notamment pour les propriétés comme le `margin` qui peut se faire de façon générale sur les 4 côtés ou de façon précise sur chacun des côtés. Cela permet à l'utilisateur de pouvoir éditer un style de façon simple ou précise et de rester le plus épuré dans l'appel de chaque méthode. Elle se complexifie seulement au besoin et reste donc accessible pour un utilisateur débutant.

Il est à noter que nous avons essayé de mettre en place des "shortcuts" pour des propriétés récurrentes ou alors celle dont la valeur affectée va souvent être la même. Par exemple la méthode `centerText` qui est équivalente à `setTextAlign` avec la valeur "Center".

Notre DSL comporte néanmoins quelques limites :

Le chaînage d'opérateur peut parfois donner un fichier assez vite volumineux. Si l'utilisateur n'est pas rigoureux avec l'indentation et les retours à la ligne, il peut se retrouver avec quelque chose de difficilement lisible. Nous mettons donc en place les méthodes `endELEMENTedit` permettant de marquer la fin d'édition du style d'un composant. Nous montrons comment rédiger de façon claire un fichier de style avec notre langage dans les fichiers d'exemple de style du projet.

Le placement de la barre de navigation ainsi que les possibilités d'affichage sont limitées pour cet élément mais permet tout de même de naviguer entre les différents wiki du sujet.

Le positionnement des éléments pour qu'il soit tout le temps visible n'a pas été implémenté mais est une évolution qui aurait été envisagée pour la continuité du projet.

Toutes les possibilités de style qui sont théoriquement possible dans une page web ne sont pas implémenter, cela aurait demandé de réaliser de projet sur une période bien plus large pour couvrir l'ensemble de celle-ci

Notre modèle s'adresse aux utilisateurs qui comprennent le vocabulaire des différentes propriétés CSS et demande donc tout de même un minimum de compréhension du fonctionnement. Notamment pour l'affichage en bloc des éléments par exemple.

## B. Choix technologiques

Pour notre DSL interne, nous avons choisi de partir sur un langage embarqué en TypeScript pour des raisons de facilité de portage et d'utilisation. Il nous serait alors facile de créer une librairie que l'utilisateur n'aurait qu'à importer (avec npm par exemple). Nous n'avons pas choisi Javascript, bien qu'encore plus léger, car il n'y a pas de typage strict, ce qui peut perdre l'utilisateur dans la rédaction de son modèle.

Un autre avantage de TypeScript est que l'utilisateur peut savoir quelle fonction est disponible sur le générateur actuel et quel type de paramètre (avec son nom) est attendu dans les fonctions de génération du modèle. De cette façon, notre DSL est plus intuitif et plus facile à prendre en main, surtout lorsque l'utilisateur n'a pas de notion en informatique.

Une critique que nous avons déjà émise lors du premier projet de DSL est que, dans notre langage, rien n'empêche l'utilisateur d'appeler la fonction `createModel()` de nos sous-objets (Subject, RelatedSubject, Summary par exemple) alors que celle-ci ne doit être appelée que par le code de `createModel()` de notre composant Wiki. Cela peut donc générer des sous-objets incomplets et non traduisibles en code CSS.

Également, TypeScript ne permet pas en lui-même de retirer les parenthèses des fonctions, ce qui ajoute des informations superflues pour l'utilisateur, surtout lors d'appel de méthodes de type `.edit...` où aucun paramètre n'a besoin d'être rentré dans la fonction.

Finalement, l'appel de fin d'un élément comme `.endTitleEdit()`, `.endTitleEdit()` n'apporte pas de valeur fonctionnelle dans notre génération mais permet une structuration du code qui peut vite devenir complexe. Cela reste néanmoins quelque chose d'obligatoire dans la façon dont est implémenté notre DSL car il faut délimiter pour chaque builder l'endroit où il se termine afin d'accéder à son parent.

## IV. Répartition des tâches

- ❖ Réflexions sur le diagramme de classes : Toute l'équipe
- ❖ Kernel : Toute l'équipe
- ❖ DSL interne : Toute l'équipe
- ❖ Générateur du code ReactJS : Toute l'équipe
- ❖ Test du code généré : Yann & Yohann

## V. Rationalisation des principaux usages et de toutes les caractéristiques

Notre DSL permet donc de créer un style générique pour une encyclopédie. Tous les sujets de cette encyclopédie et leur contenu seront ainsi uniformes dans leur style. Le côté pratique de notre DSL permet à n'importe quel designer de le prendre assez simplement en main car il se base sur des propriétés qui font références à celles du CSS, langage que nous estimons maîtrisé par tout designer qui se respecte. Mais notre DSL ne se limite pas qu'à une copie du langage CSS car il offre des raccourcis via ses méthodes et ses paramètres optionnels et une vitesse de stylisation plus rapide que l'écriture de CSS directement.

Nous offrons également la possibilité de mettre en place un style généraliste qui peut être écrit dans le wiki et de l'affiner au fur et à mesure des composants en éditant à nouveau les mêmes propriétés dans les composants plus précis. Il est donc facile d'arriver à un rendu visuel rapidement afin de se projeter ensuite dans les détails.



Nous avons mis à disposition la manipulation de nombreux éléments métiers qui sont présents dans la majorité des encyclopédies : une table des matières, une info boîte, un résumé, des chapitres, des références... Pour chacun, l'utilisateur peut définir son style à l'aide des différents paramètres disponibles : la taille de la police, la couleur de la police, la couleur du fond, la marge... Cela lui permet de démarquer certains éléments du wiki vis à vis des autres, comme la section des références par exemple.

Une de nos killer features est le fait que l'utilisateur peut mettre en place un style alternatif en fonction de la taille de l'écran. Par exemple, si celui-ci veut afficher le texte sur plusieurs colonnes quand l'espace lui permet et sur une seule dans le cas contraire cela est possible. La stylisation alternative est aussi précise que la stylisation normale. Il est possible de définir le style alternatif global et de descendre dans le détail pour chacun du contenu des composants et de leur boîte de contenu.

Notre deuxième killer feature est qu'il en va de même pour le style en hover de chaque composants (ou du wiki en général..) avec la même logique pour que pour le style alternatif responsive.

Enfin notre troisième killer feature sera la mise en place de composants tableaux avec un style assez permissif dont notamment des titres avec un placement en haut à gauche ou à droite dans le tableau. Ce composant est quelques chose de complexe et nous avons rendu son design beaucoup plus simple pour le designer.