|  |  |  |
| --- | --- | --- |
| **Expéditeur(s)** | | AB |
|  | | Direction Technique Central |
|  | | |
|  | | |
| **Destinataire(s)** | | Prénom Nom, Prénom Nom |
| **Copie(s)** | | Prénom Nom, Prénom Nom |
|  | | |
|  | | |
| **Date** | | 01/08/2018 |
| **Objet** | Le Cahier des charges BIM BI | |

|  |  |
| --- | --- |
|  | |
| **NOM DE LA RÉGION**  00 Nom de la Rue, F-00000 Nom de la Ville CEDEX  tél : 00 00 00 00 00 fax : 00 00 00 00 00 www.bouygues-immobilier.com |  |

|  |
| --- |
|  |
|  |
| **Sujet(s) abordé(s)** |
|  |

# Le Cahier des charges BIM BI

Le Cahier des charges BIM Bouygues Immobilier existe en 2 formats : un PDF, et un site internet. Le site internet est généré à partir de GitBook. Cette note a pour but d’expliquer la structure du GitBook, de quelle façon il a été généré et codé. Sa structure est très similaire à la façon de concevoir un livre (d’où le nom GitBook).

Table des matières

[Le Cahier des charges BIM BI 1](#_Toc521339196)

[1) Structure générale / Sommaire 1](#_Toc521339197)

[2) Installations des logiciels 2](#_Toc521339198)

[a) GitBook 2](#_Toc521339199)

[b) Visual Studio Code 3](#_Toc521339200)

[c) Git 4](#_Toc521339201)

[3) Structure point de vue utilisateur 4](#_Toc521339202)

[a) Suivre les instructions et cliquer sur les liens rencontrés 4](#_Toc521339203)

[b) Utiliser le menu de navigation 6](#_Toc521339204)

[4) Structure point de vue rédacteur 7](#_Toc521339205)

[a) Comment éditer le GitBook 7](#_Toc521339206)

[b) Arborescence des fichiers textes du Cahier des charges BIM BI 13](#_Toc521339207)

[c) Liens / appels entre les fichiers 15](#_Toc521339208)

## Structure générale / sommaire

Le format GitBook est un ensemble de fichiers textes organisés selon une logique définie. Le fichier le plus important est celui nommé « SUMMARY.md ». L’extension .md signifie que le fichier est rédigé en langage Markdown, vous trouverez les règles de syntaxe à cette adresse : <https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet>

Ce fichier SUMMARY.md est la page de garde du livre, c’est là que sont listées toutes les pages contenues dans le GitBook. Ces pages peuvent être organisées selon une arborescence, avec des pages principales, et des sous-pages. Dans le SUMMARY, une ligne correspond à une page, et chaque ligne doit commencer par une « \* » suivie d’un espace. Pour créer une ou plusieurs sous-pages d’une page mère, il faut rajouter un espace devant le « \* ». Voici un extrait du SUMMARY.md du GitBook :

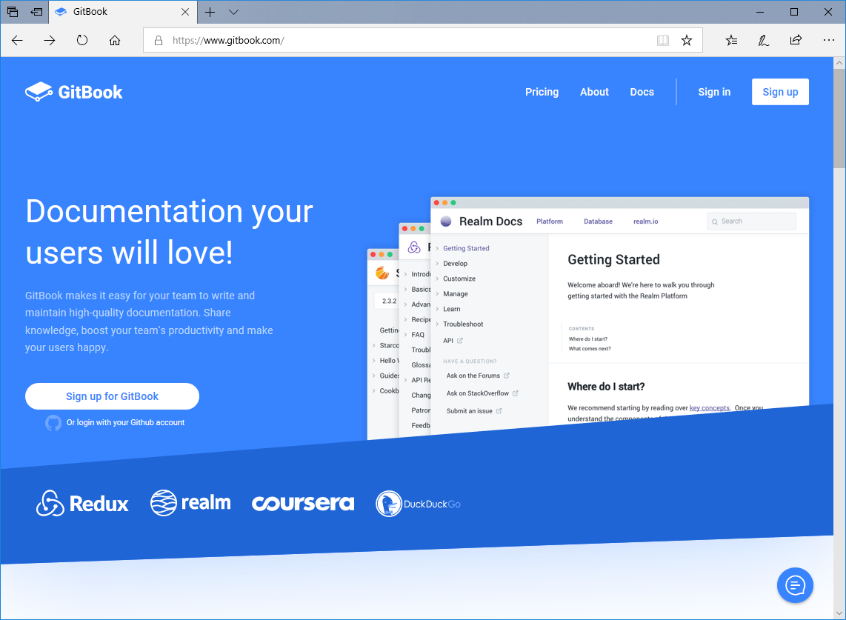
Pour la syntaxe, il faut écrire le titre de la page entre crochets, et le chemin vers l’endroit où est enregistré le fichier texte de la page correspondante. Par exemple, si vous avez enregistré le fichier texte de la page 3 (dont le fichier texte a été nommé « page\_3.md ») dans le dossier\_2, lui-même stocké dans le dossier\_1, il faudra écrire dans le SUMMARY :

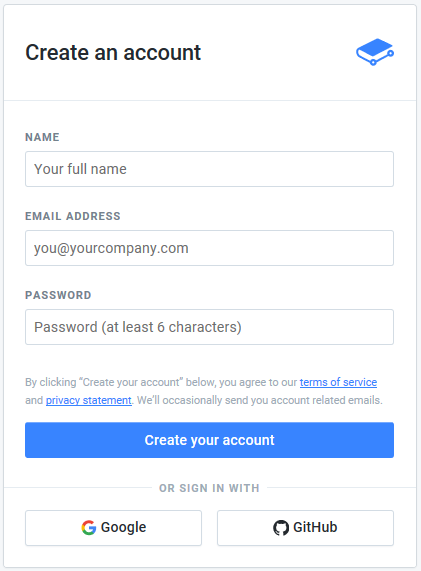
\* [page 3](/dossier\_1/dossier\_2/page\_3.md)

## Installations des logiciels

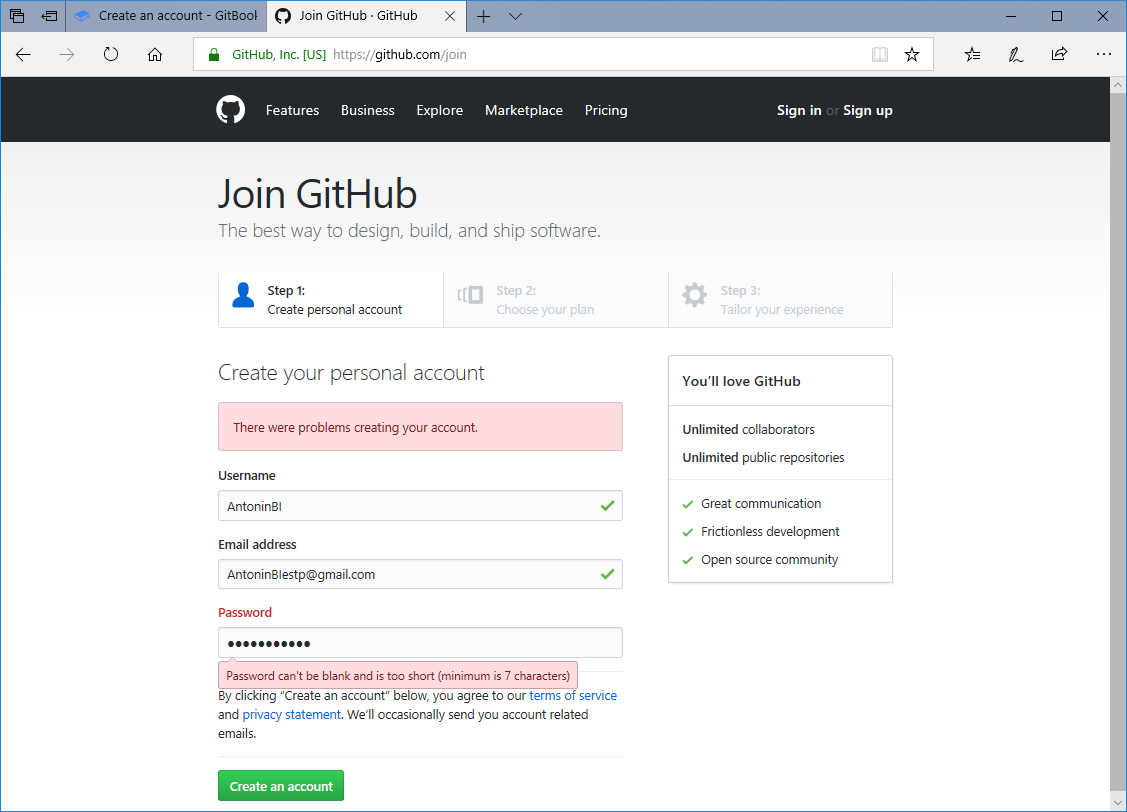
### GitBook

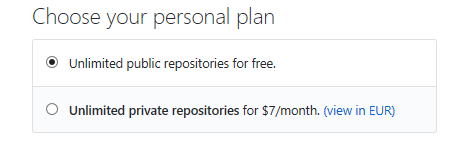
Pour pouvoir éditer le GitBook, il faut être inscrit sur le site <https://www.gitbook.com/>

Pour s’inscrire, cliquer sur Sign up (1). Dans la fenêtre qui s’ouvre on va utiliser l’option OR SIGN IN WITH Github (2). Il faut donc dans un premier temps s’inscrire sur GitHub.

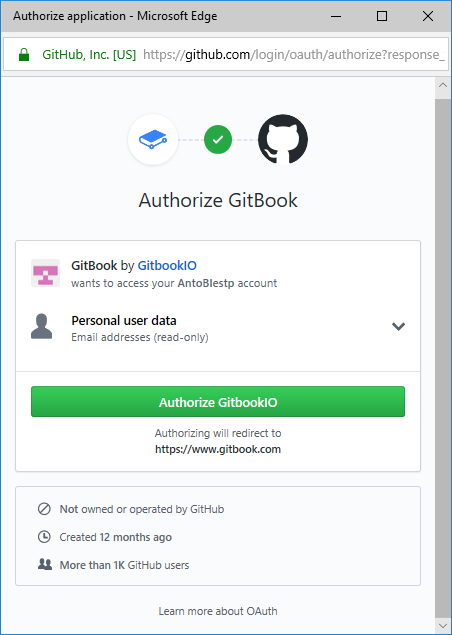


Pour s’inscrire sur GitHub, aller sur https://github.com/ , puis cliquer sur Sign up for GitHub. Renseigner les informations nécessaires à la création de votre compte GitHub (un pseudo, un email et un mot de passe), et cliquer sur Create an account.



Choisir l’option numéro 1 pour le plan personnel, puis valider.

Enfin, répondre aux différentes questions posées à l’étape 3, et cliquer sur Submit. Vous pouvez également cliquer sur Skip this step.

Votre compte GitHub est maintenant créé, vous devez à présent le valider en vous connectant à votre adresse mail et suivre les instructions de validation.

Une fois votre compte validé, vous pouvez vous connecter à GitBook via l’option OR SIGN IN WITH GitHub. Une fenêtre s’ouvre automatiquement, patienter quelque instants, puis cliquer sur Authorize GitobookIO quand cela est possible.

Il faut à présent valider le compte GitBook, en suivant les instructions proposées (un téléphone sera nécessaire).

Vous être maintenant inscrit sur GitBook, et pouvez commencer à rédiger ou éditer des livres.

### Visual Studio Code

Pour créer et éditer les fichiers textes constituant le GitBook, il faut utiliser l’éditeur de texte Visual Studio Code, très adapté pour cette tâche. Pour le télécharger, aller sur <https://code.visualstudio.com/download>, puis choisir la version qui correspond à votre PC. Une fois le fichier exe téléchargé, contacter le service informatique de BI (3333) pour l’installer, car il faut une autorisation administrateur.

### Git

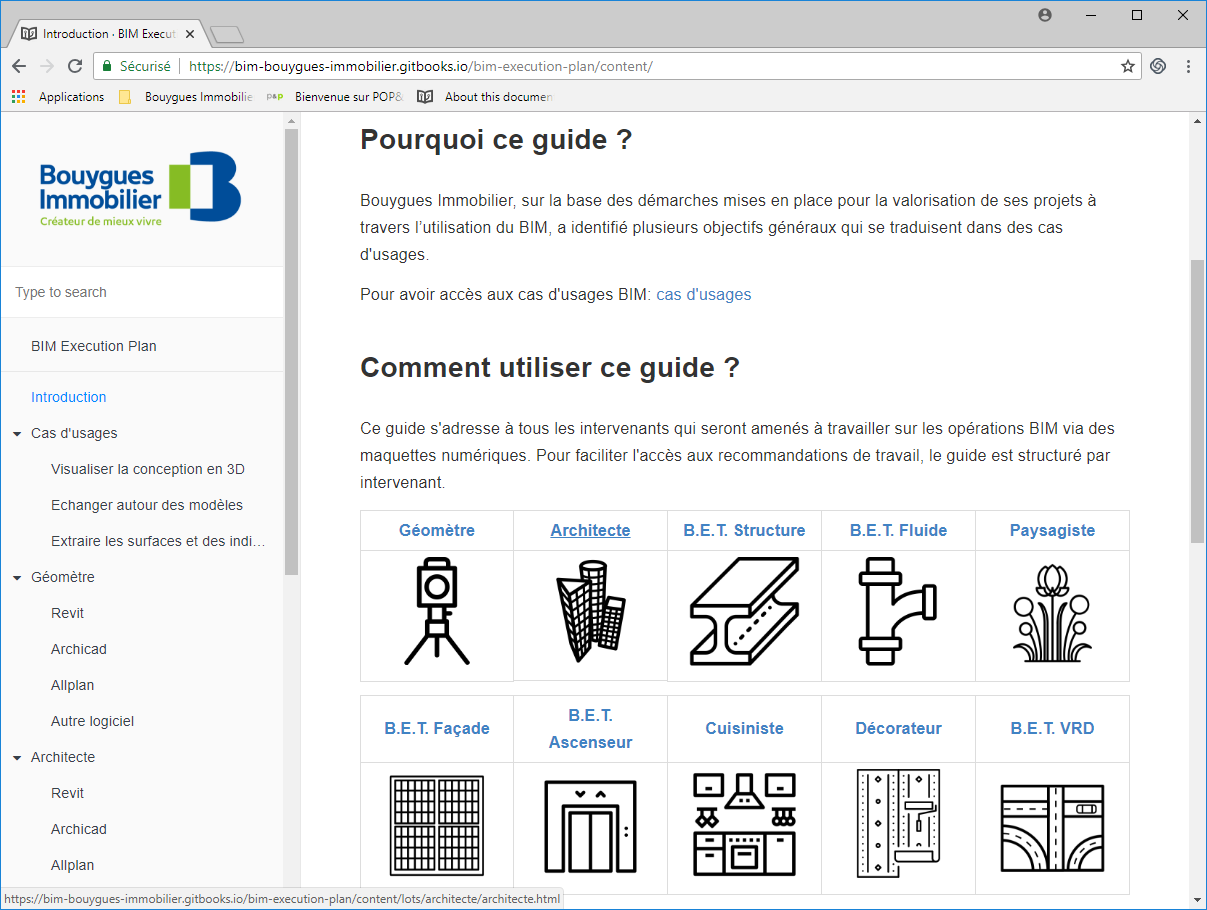
Vous devez également télécharger et installer Git. Pour cela, il faut aller sur l’adresse <https://git-scm.com/downloads>, puis télécharger la version qui correspond à votre PC. Une fois le fichier exe téléchargé, contacter le service informatique de BI (3333) pour l’installer, car il faut une autorisation administrateur.

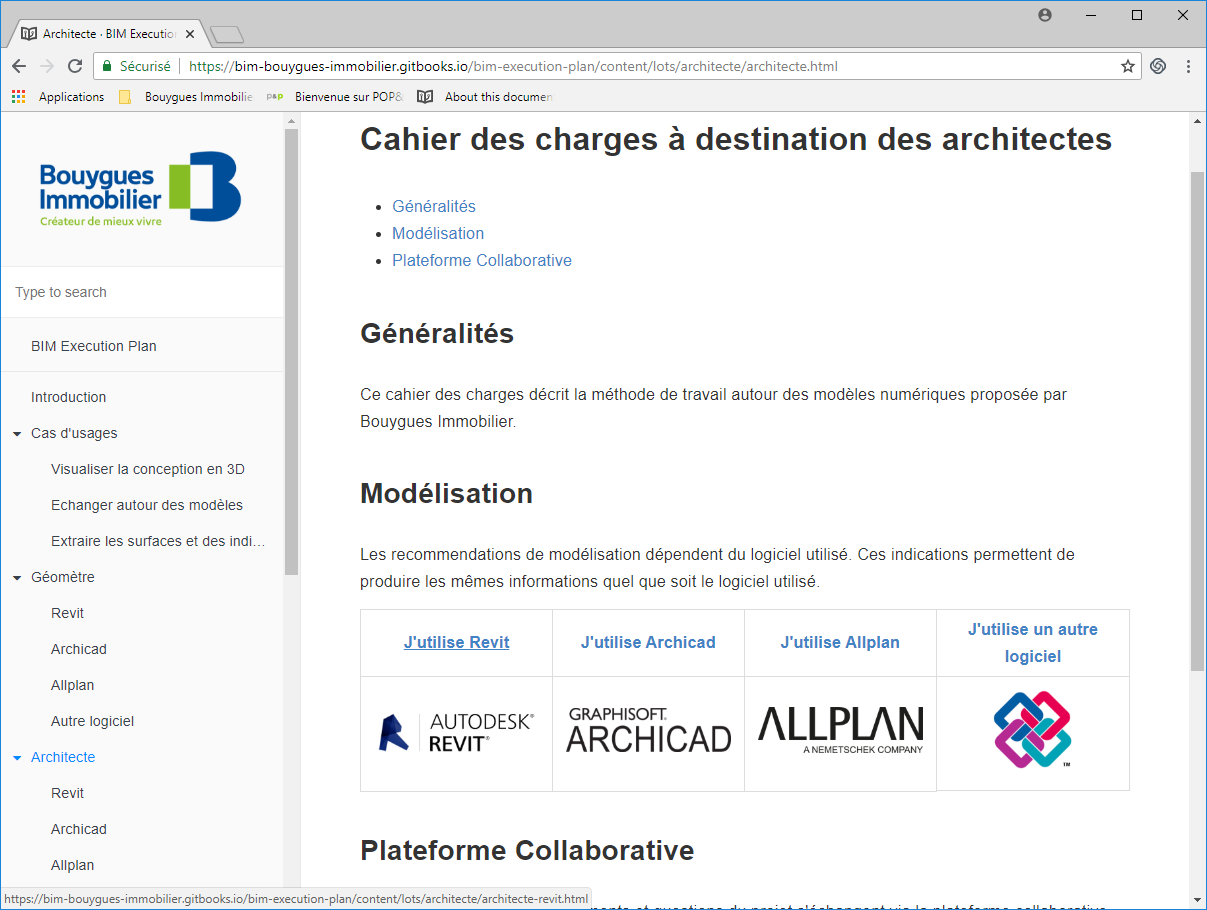
## Structure point de vue utilisateur

L’adresse du Cahier des charges BIM est : <https://bim-bouygues-immobilier.gitbooks.io/bim-execution-plan/content/>

Il est en consultation libre, Open source. Il est constitué de 3 grandes parties : une partie décrivant les raisons de sa création (appelée « Cas d’usages »), une partie expliquant la façon de modéliser attendue, et une partie concernant la plateforme collaborative bimsync, utilisée pour tous les projet en BIM chez BI. Sur la page d’introduction, on retrouve les explications concernant le but du cahier des charges, et quelques indices sur la façon de l’utiliser (comment naviguer entre les différentes pages…).  
Il y a deux façons de naviguer dans le GitBook : en suivant les instructions écrites au fur et à mesure et en cliquant sur les différents liens que vous allez rencontrer, ou en se servant du menu de navigation sur la partie gauche de l’écran.

### Suivre les instructions et cliquer sur les liens rencontrés:

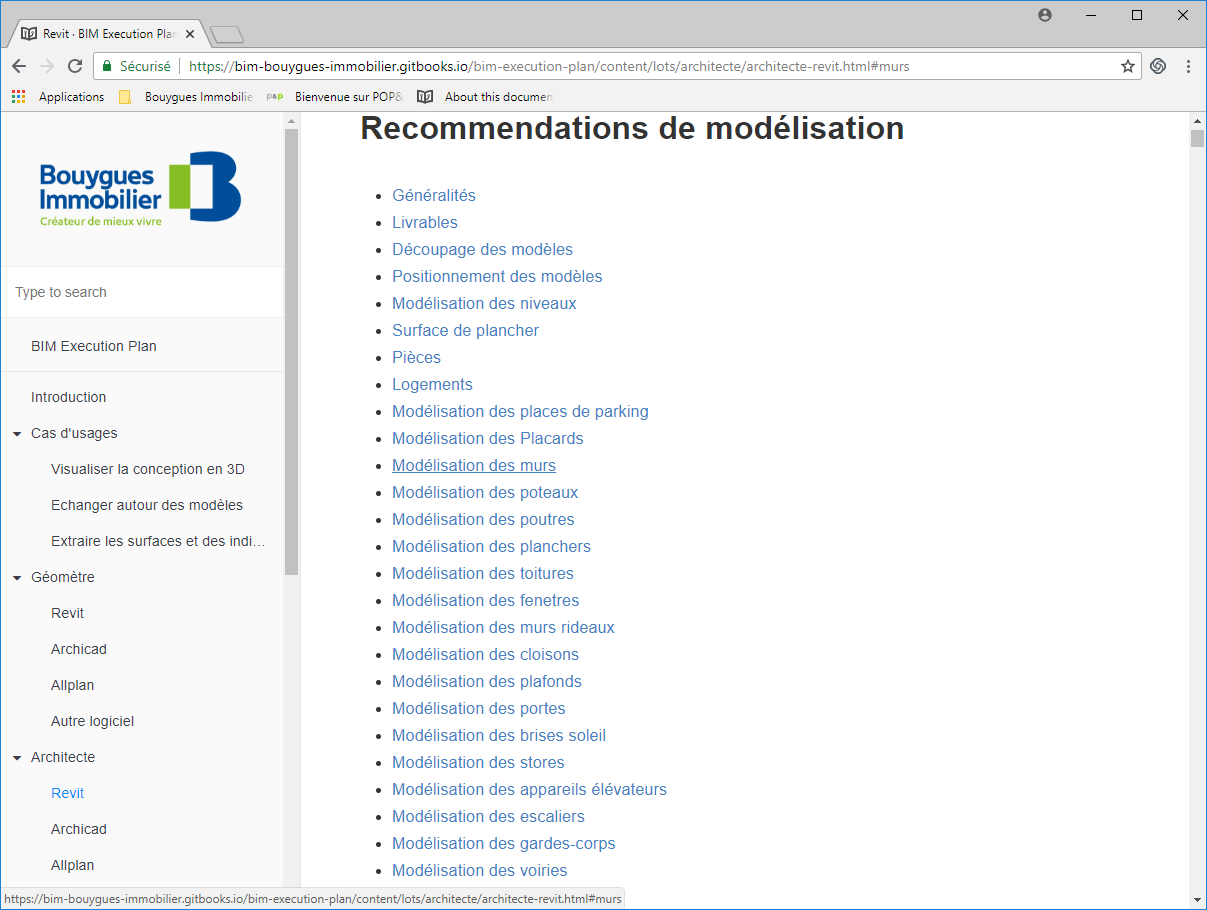
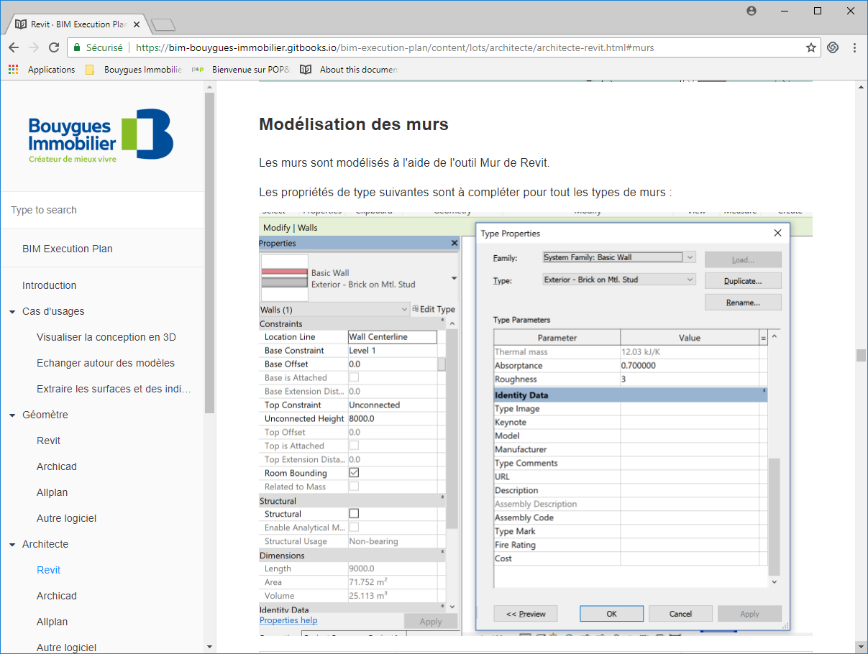
  
  
  
  
  
  
  
Par exemple, je suis architecte, donc je clique sur l’image correspondante (1), ou sur le nom juste au-dessus de l’image.



J’ai ainsi accédé à la page architecte. Ensuite, je choisis le logiciel que j’utilise pour modéliser, par exemple Revit : je clique sur l’image correspondante (2), ou le nom.

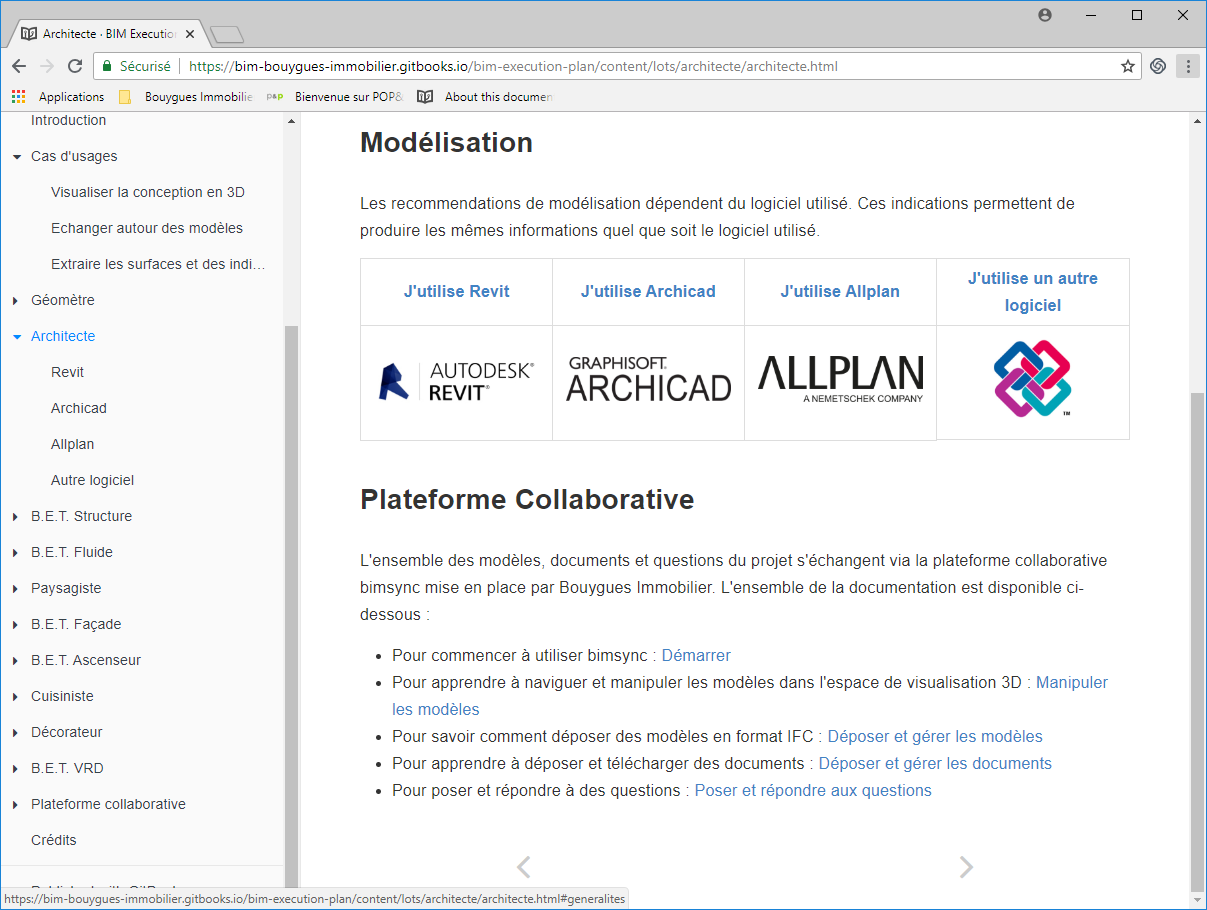
J’ai enfin accédé à la page guide de modélisation (ici, dédiée aux architectes utilisant Revit). Sur cette page, sont présentes toutes les instructions nécessaires à un architecte pour qu’il puisse modéliser ses maquettes conformément aux attentes de Bouygues Immobilier. En haut de page, sont indiqués tous les différents points qui vont être abordés dans la page ; c’est une sorte de sommaire de la page. Chaque point est également un lien vers la partie concernée. Il suffit de cliquer dessus pour être automatiquement dirigé vers la partie en question plus bas dans la page.

Par exemple, si on clique sur «  \* Modélisation des murs » :



La page des recommandations de modélisations est composée d’une succession de textes, de tableaux et d’images. Les images sont là pour appuyer le texte, et en général des numéros sont indiqués sur l’image et dans le texte pour faciliter la compréhension du lecteur. On retrouve également des liens dans la partie texte, parfois pour télécharger des fichiers (des fichiers natifs Revit/Archicad/Allplan par exemple), ou accéder à une autre page du Gitbook.

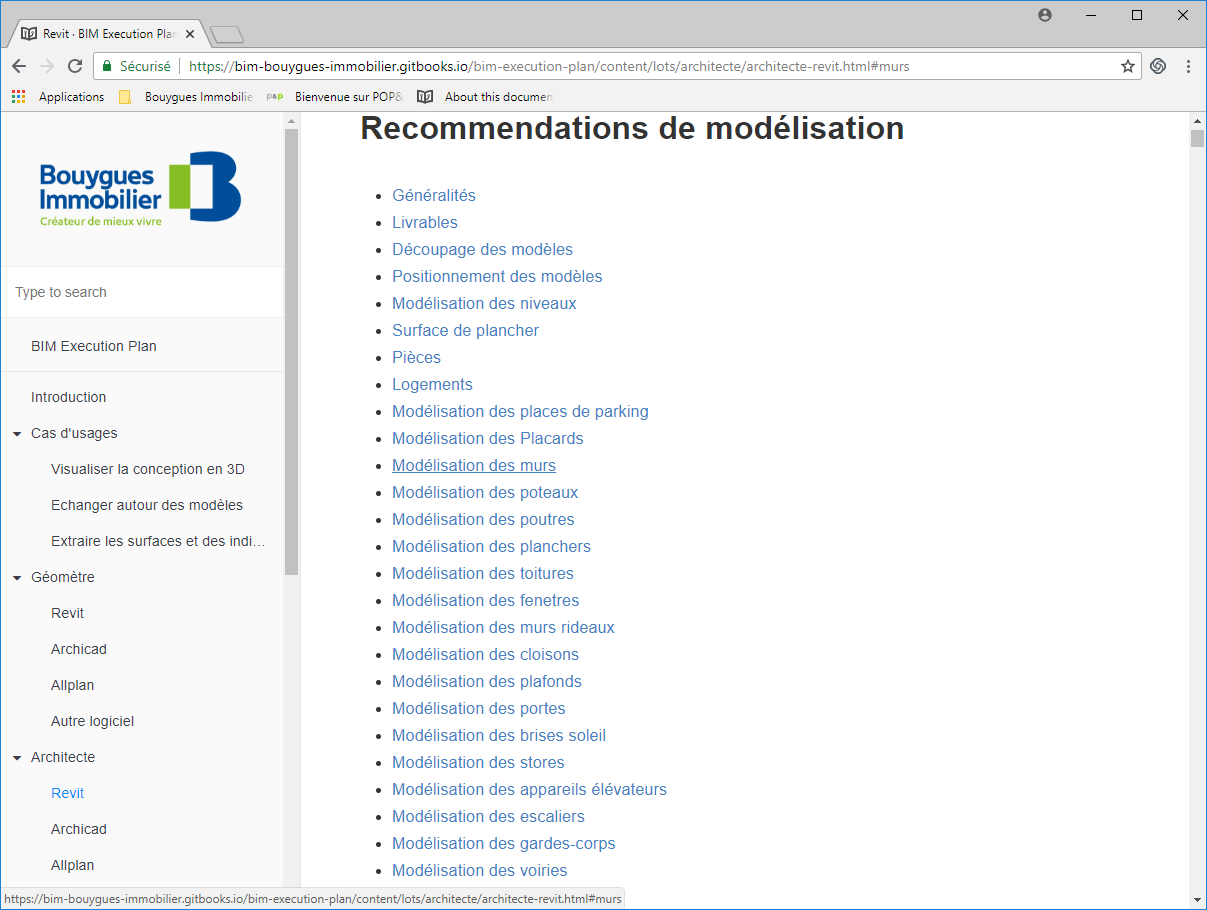
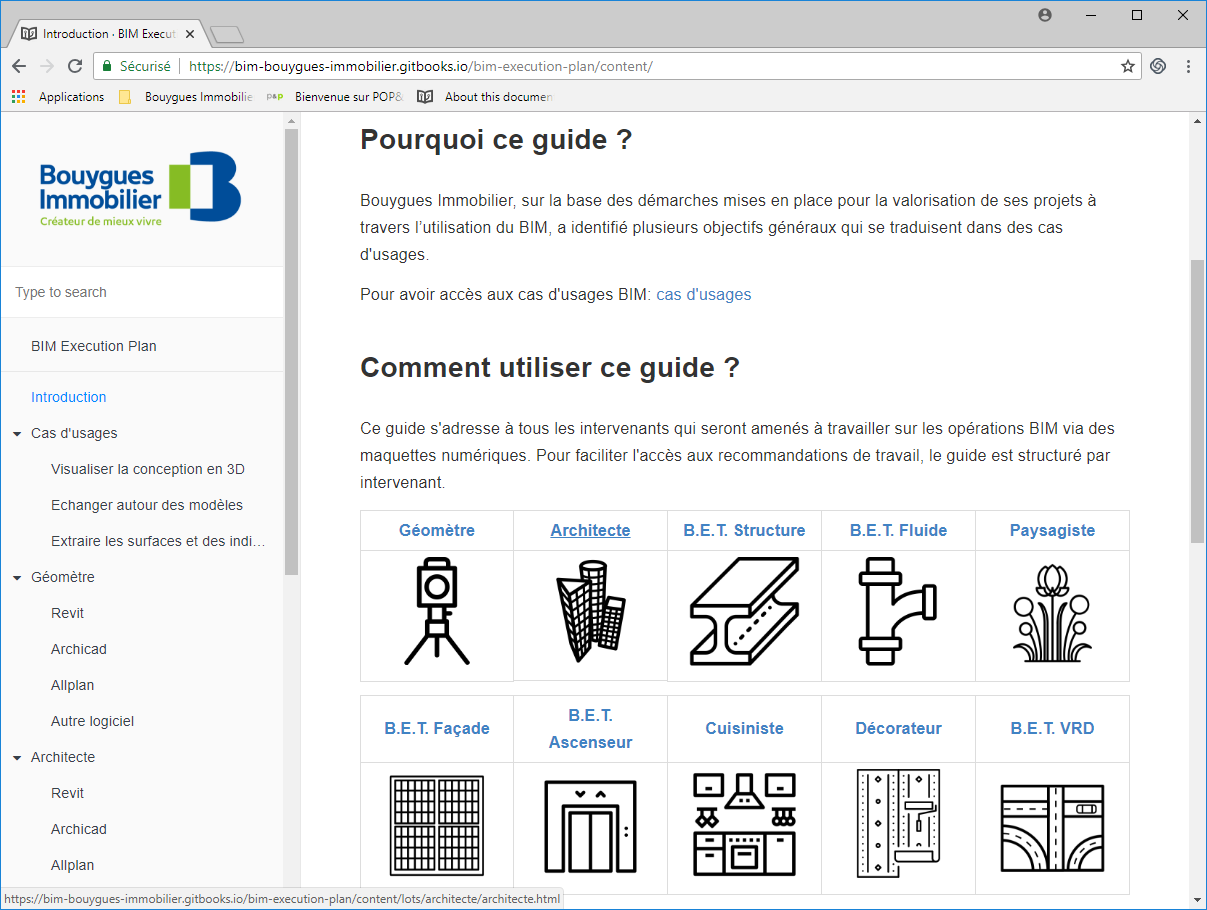
Pour se rendre à la partie concernant bimsync, il faut passer par la page Architecte (avant d’avoir sélectionné un logiciel de modélisation). En bas de la page, se trouvent plusieurs liens pour accéder aux différents points abordés dans cette partie.



La partie bimsync est en fait rédigée et mise à jour sur un autre GitBook, dont l’auteur est également Bouygues Immobilier. Les fichiers ont été copiés collés depuis cet autre Gitbook, et donc doivent être modifiés manuellement si des changements surviennent dans la documentation.

### Utiliser le menu de navigation

La partie gauche de la page du GitBook est réservée pour le menu de navigation, contenant chacune des pages du livre. Vous pouvez développer ou réduire l’arborescence en cliquant sur les petits triangles devant les titres des pages principales. Ainsi, vous pouvez vous rendre directement à n’importe quelle page en cliquant sur son titre, sans passer par des pages intermédiaires (contrairement à la première façon de naviguer).

Par exemple, en déroulant le menu « Architecte », et en cliquant sur Revit (1), on se rend directement à la page guide de modélisation dédiée aux architectes utilisant Revit :

## Structure point de vue rédacteur

Le Cahier des Charge BIM BI est généré par GitBook, avec comme données d’entrée des fichiers textes. Ces fichiers textes sont hébergés sur GitHub. La façon dont est organisée l’arborescence des dossiers et des fichiers du livre n’a aucune importance, tant que les liens sont établis en respectant cette arborescence. C’est-à-dire, lorsque des liens sont établis entre certains des fichiers textes, il faut s’assurer que le chemin d’appel soit correct.

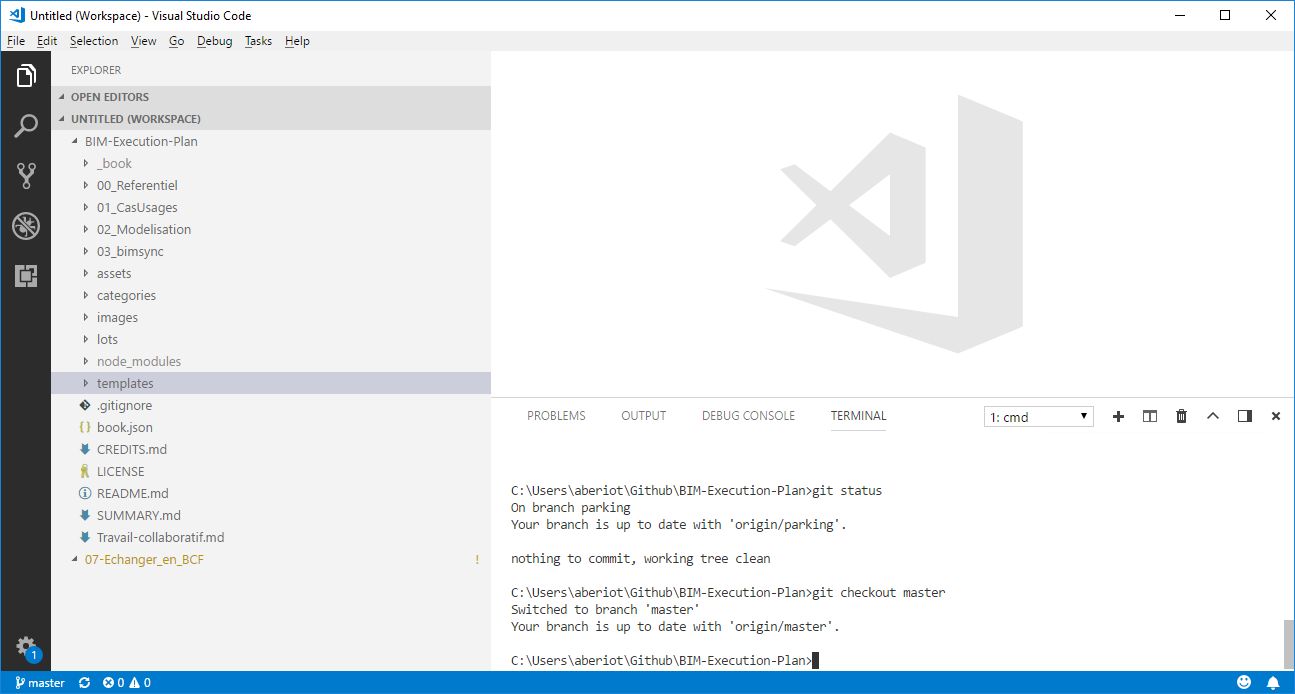
### Comment éditer le GitBook

La première chose à faire est de récupérer les fichiers textes sur votre PC, depuis GitHub. Pour cela, aller à l’adresse <https://github.com/BIM-Bouygues-Immobilier/BIM-Execution-Plan> (assurez-vous d’être connecté sur votre compte GitHub), cliquer sur le bouton vert Clone or download, puis copier le lien qui s’affiche (1).

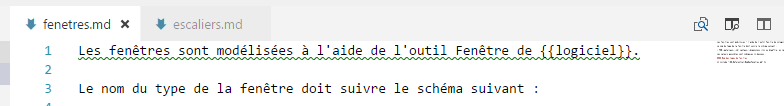
Aller ensuite dans Visual Studio Code, puis rentrer la commande suivante :

git clone git@https://github.com/BIM-Bouygues-Immobilier/BIM-Execution-Plan.git

cette commande permet d’obtenir une copie du dépôt Git existant du cahier des charges. Vous devez donc avoir l’arborescence du cahier des charges qui apparait sur la gauche, comme sur l’image suivante. Pour l’afficher, ou la cacher cliquer sur l’icône (2).

Vous pouvez ouvrir et éditer les fichiers textes composant le cahier des charges, en utilisant cette arborescence pour accéder aux fichiers qui vous intéressent. Ces fichiers s’ouvrent et se modifient dans la partie haute droite de l’écran (3).

Plusieurs fichiers textes peuvent être ouverts et modifiés en même temps, ils seront empilés en plusieurs onglets (4).



Vous pouvez les modifier librement, car les changements seront uniquement faits en local sur votre machine.

Pour proposer vos modifications à l’auteur du GitBook, plusieurs manipulations sont nécessaires.

Tout d’abord, il est important de connaître et comprendre le fonctionnement de Git.

C’est est un logiciel de gestion de versions, qui permets à plusieurs utilisateurs de modifier un document, en proposant des nouvelles versions de celui-ci. Ces différentes versions peuvent être gardées, refusées, fusionnées…

En prenant l’exemple du cahier des charges BIM, vous pouvez librement modifier les fichiers copies que vous avez récupérés sur GitHub, ce qui donnera une version alternative du cahier des charges. Lorsque vous proposer votre nouvelle version à l’auteur du GitBook, celui-ci va comparer les 2 versions, et choisir ce qu’il veut garder parmi les morceaux qui diffèrent.

Git permet également à un utilisateur de travailler en simultané sur 2 versions d’un document : c’est le principe des branches. Le document (ici le GitBook), à une branche principale (appelée Master), et des branches secondaires (le choix du nom est libre). La branche principale est la version du document qui est publiée, visible pour les lecteurs. Les branches secondaires servent aux rédacteurs pour travailler sur les modifications à apporter au document, sans qu’elles soient publiées, c’est-à-dire visible par les lecteurs.

Pour apporter des modifications au Cahier des charges BIM BI, il est demandé de créer une nouvelle branche avant de commencer à modifier les documents. En aucun cas ne travailler sur la branche Master. Pour créer une nouvelle branche, entrer dans l’espace de commande de Visual Studio Code la formule suivante :

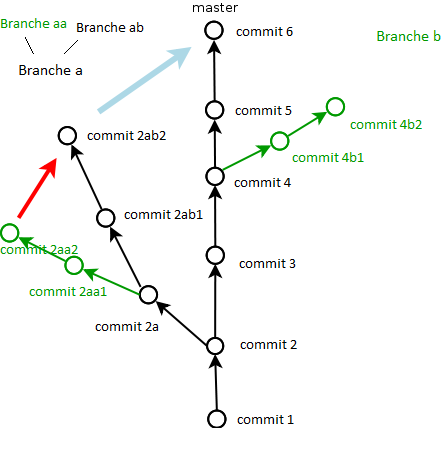
git checkout -b NomDeLaBranche

pour s’assurer d’être sur la bonne branche, entrer la formule : git status

Visual Studio Code indiquera sur quelle branche les modifications seront apportées.

En plus des branches, une autre notion fondamentale du logiciel est le commit. Pour faire simple, un commit correspond à un enregistrement, une sauvegarde d’une version. C’est une sorte de check point de la branche, qui permet de figer une version dans le temps. L’idée est de pouvoir revenir en arrière si les changements opérés ne s’avère pas judicieux. Ainsi, chaque branche est une succession de commits.

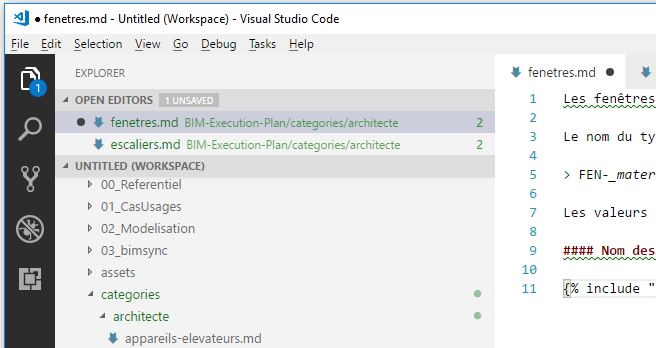
Le schéma suivant illustre assez bien le concept git :

La branche master est la branche principale, les branches a et b sont des branches secondaires. Les branches secondaires peuvent également avoir des sous-branches, comme illustré sur les schéma avec les branches aa et ab.

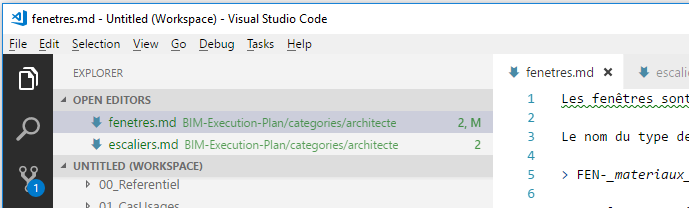
On remarque que les branches sont un enchainement de nœuds (les commits), les bifurcations de branche s’opèrent au niveau de ces nœuds.

A tout moment, le rédacteur peut choisir de revenir à n’importe quel commit.

Lorsque vous être sur votre branche, et que vous avez apporté des modifications à des fichiers, la première action à effectuer est d’enregistrer ces modifications en local. Le nombre de fichiers modifiés est indiqué en haut à gauche.



Une fois les fichiers enregistrés, il faut effectuer 3 opérations pour créer un commit en ligne de ces modifications. Premièrement, il faut ajouter les fichiers à la liste des fichiers à commit. Pour cela, entrer la commande suivante :   
  
git add .

Puis, il faut commit les fichiers ajoutés à cette liste. Le nombre de fichiers à commit est indiqué en haut à gauche : 

Pour créer un commit regroupant les modifications apportées aux fichiers, entrer la commande :

git commit -m «commentaires»

Mettre en commentaire un bref résumé du commit.

Enfin, il faut mettre en ligne le commit, pour cela entrer la commande :

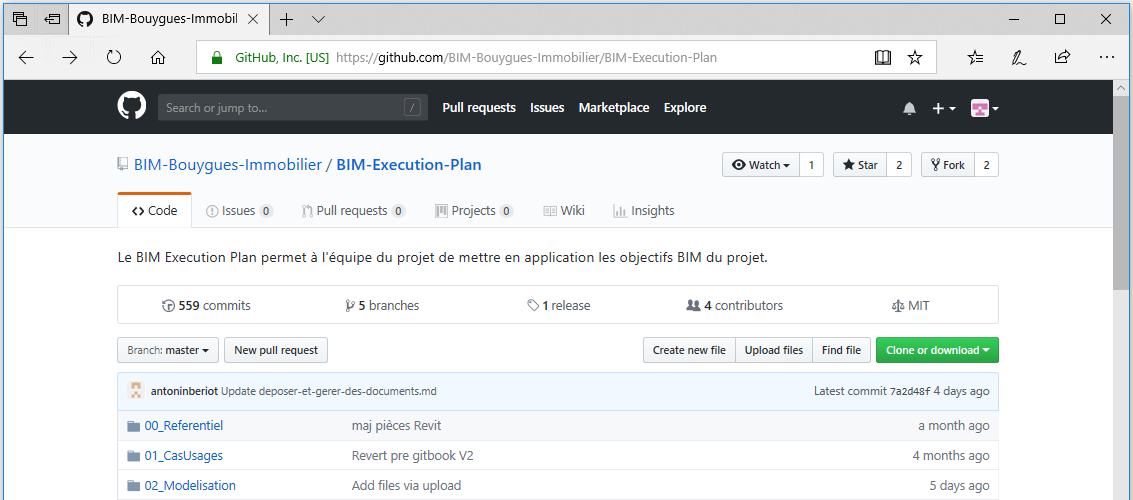
Git push -u origin NomDeLaBranche

Visual Studio Code vous demandera à ce moment de confirmer votre identité, il faudra donc rentrer votre nom d’utilisateur GitHub ainsi que votre mot de passe. Une fois ces information renseignées, votre commit sera disponible sur GitHub, et vous recevrez un mail de confirmation que votre push à bien été effectué.

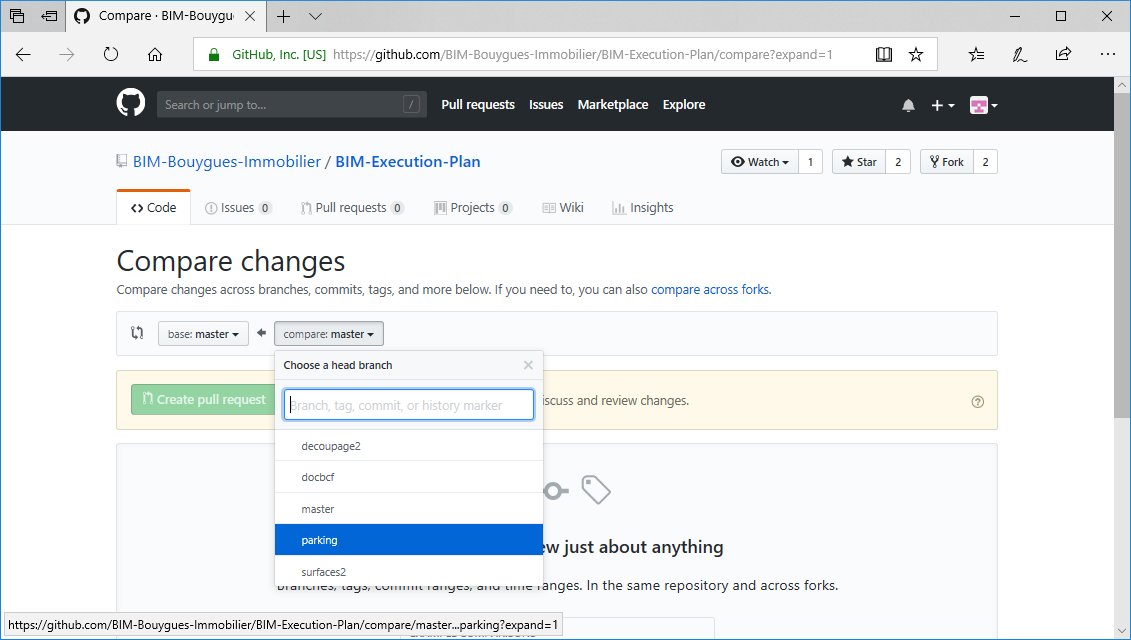
Une autre commande est importante à connaitre, le pull. Elle permet de mettre à jour vos fichiers locaux, s’il y a eu des modifications effectuées par d’autres contributeurs.

Cette commande s’écrit : git pull

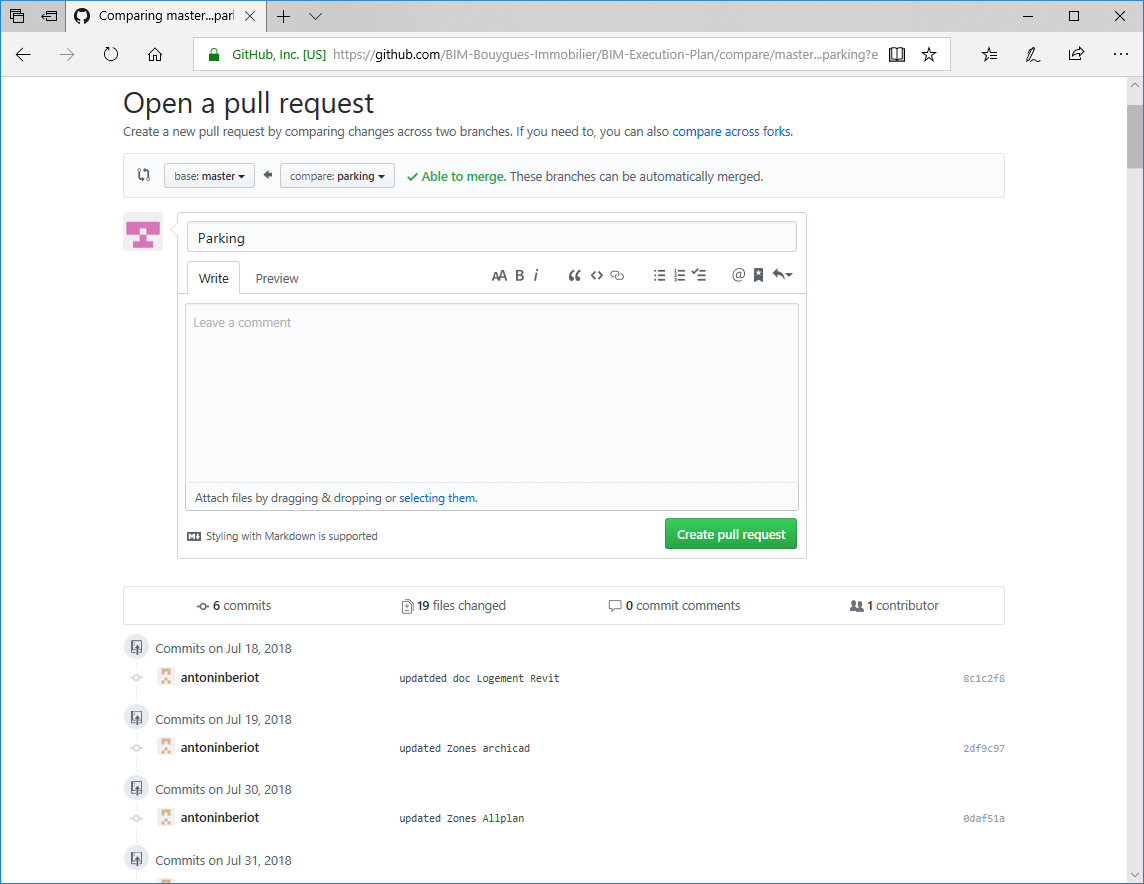
Lorsque vous être satisfait des modifications apportées à la branche, vous devez la faire converger vers la branche master ; vous allez en quelque sorte greffer votre branche sur la branche master. Le terme technique est Merge la nouvelle branche vers la branche master. C’est à ce moment où intervient la comparaison des branches. Si des parties diffèrent, Visual Studio Code va demander à l’auteur lesquelles doivent être gardées. Comme vous n’êtes pas l’auteur du Cahier des charges, vous ne pouvez pas merge directement, il faut proposer votre branche à celui-ci. Il faut donc faire un pull request. Ce pull request se fait sur Github, à l’aide de l’option « New pull request » (5).



Il faut ensuite choisir les branches à comparer, ici par exemple, parking avec master.

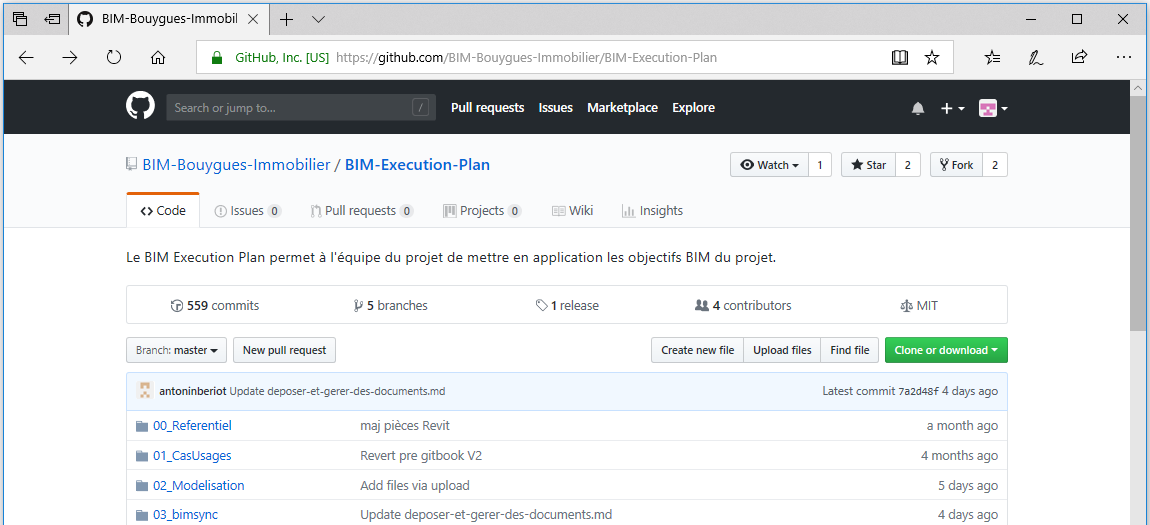


Dans la nouvelle page qui s’affiche, on retrouve la liste des commits (6) et des fichiers qui ont subis des modifications (7). Lorsque vous avez vérifié que tout est bien en ordre, vous pouvez créer la pull request, en cliquant sur Create pull request (8). Ne pas hésiter à laisser un commentaire dans la zone prévue à cet effet, pour résumer les raisons du pull request.



Une fois la pull request envoyée, il faut attendre que l’auteur du GitBook la consulte, la valide, puis la merge sur la branche master. Dès que c’est fait, la branche master contient vos modifications, vous pouvez donc supprimer votre branche secondaire, et en créer une nouvelle, pour des nouvelles modifications.

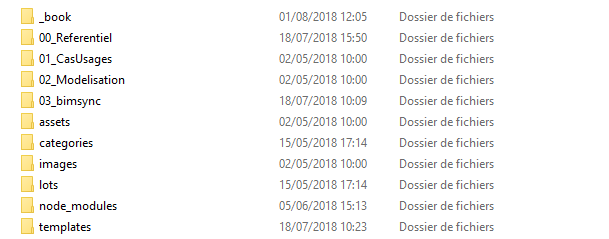
Pour supprimer la branche, aller dans GitHub, cliquer sur X Branches (9), X étant le nombre de branche actuelles du GitBook.



Enfin, dans la page qui s’affiche, cliquer sur Delete branche à droite de la branche qui vient d’être merged.

### Arborescence des fichiers textes du Cahier des charges BIM BI

L’arborescence du cahier des charges BIM s’organise dans un premier temps autour de ces quelque dossiers :

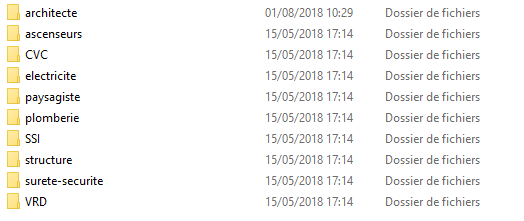


Notons que tous ces dossiers ne sont pas nécessairement utilisés (ou très peu), car le cahier des charges est en restructuration perpétuelle ; le but étant de lui trouver une structure la plus optimale.

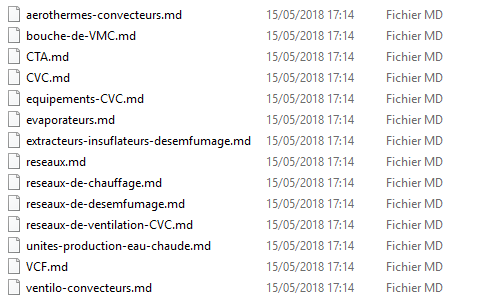
* Le dossier *00\_Referentiel* contient tous les tableaux présents dans le cahier des charges.
* Le dossier *01\_CasUsages* contient les images que l’on retrouve dans la partie décrivant les cas d’usages.
* Le dossier *02\_Modelisation* contenait précédemment les différentes méthodes de modélisation pour chaque lot et logiciel. Suite à une restructuration de l’arborescence, il ne contient plus que des images, en particulier celle concernant ces méthodes de modélisation.
* Le dossier *03\_bimsync* regroupe toute la documentation concernant la plateforme collaborative bimsync. Il s’agit d’une copie du dossier du GitBook dédié à bimsync, dont l’adresse est la suivante : <https://github.com/antoninberiot/bimsync-documentation>

Le Gitbook bimsync-documentation est également en open source, et peut être modifié par n’importe quel utilisateur GitHub (sous condition de validation par les auteurs).

* Le dossier *assets* ne contient qu’une image.
* Le dossier *catégories* contient tous les fichiers concernant les méthodes de modélisations des objets. Ce dossier est sous-divisé en lots, qui va regrouper les fichiers propres à chacun. Ces lots sont les suivants :



Chaque dossier lot contient les fichiers textes décrivant la modélisation   
 d’objets en accord avec sa discipline. Exemple du dossier CVC :



* Le dossier *images* contient quelque images
* Le dossier *lots* contient une partie des fichiers concernant les procédures accompagnant la modélisation de la maquette, comme par exemple les procédures de découpage de la maquette en lots, ou la procédure d’exportation en format IFC. Il va de pair avec le dossier *templates*.
* Le dossier *node\_modules* contient les plugins qui sont utilisés pour générer le site, comme par exemple le plugin qui affiche le logo Bouygues Immobilier en haut à gauche, ou celui qui permet d’avoir les vidéos YouTube directement visibles sur le site.
* Le dossier *templates* contient l’autre partie des fichiers concernant les procédures accompagnant la modélisation de la maquette. Il va de pair avec le dossier *lots*. Il contient également les dossiers de modélisation commune à tous les lots.

Conclusion : les dossiers les plus importants (qui sont amenés à être modifiés le plus) sont *templates, lots, catégories et 00\_Referentiel.* Bien évidemment, le dossier *03\_bimsync* devra être modifié dès que le GitBook bimsync-documentation subira des changements.

### Liens / appels entre les fichiers

1. Les images

Les images sont ajoutées dans le fichier texte, selon la syntaxe suivante :

![NomDeL’Image](/Dossier\_1/Dossier\_2/[…]/Dossier\_n/NomDuFichierImage.PNG)

Par exemple, ici l’image des combles est appelée par la formule suivante :

![Combles](/02\_Modelisation/02\_architecte/images/Combles.PNG)

1. Les includes

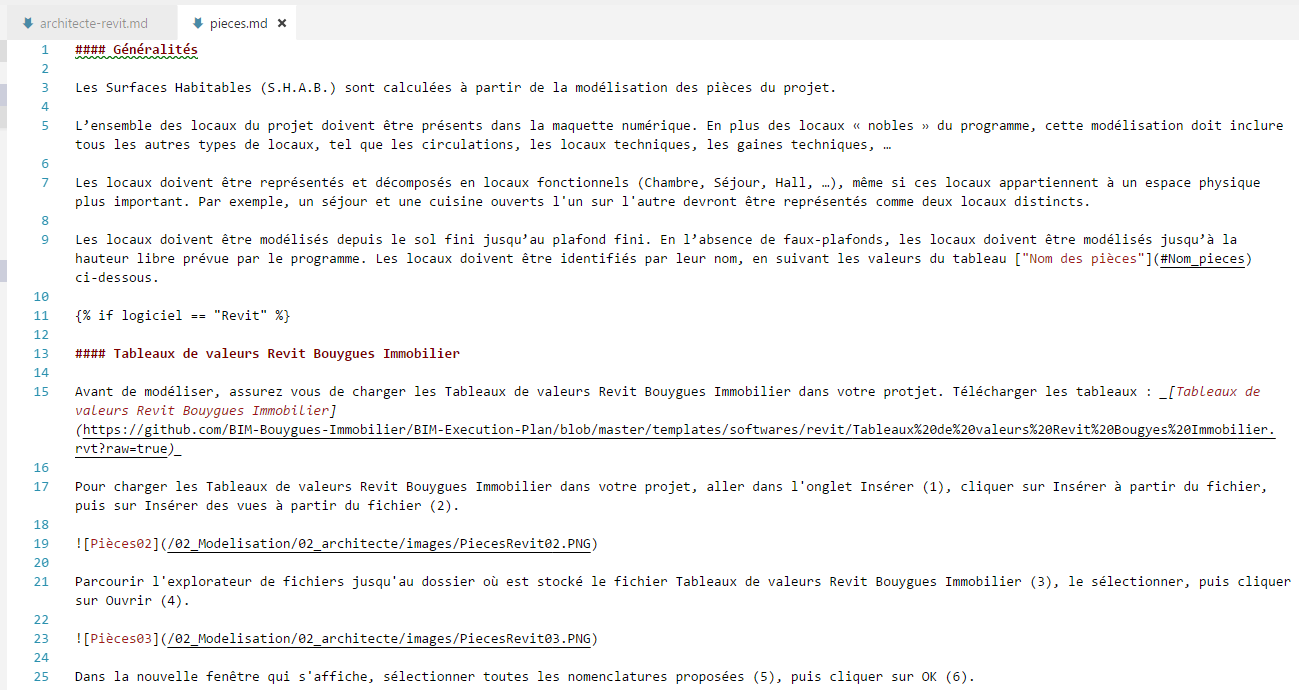
Les includes sont les appels d’un fichier entier, à un emplacement précis dans un autre fichier. Cela permet de factoriser un fichier Principal, tel qu’un fichier directement appelé dans le SUMMARY.md. L’idée est d’avoir un fichier Principal, faisant appel à plusieurs fichier secondaires, cela réduit ainsi sa taille, et permet de fractionner les modifications qu’on lui apporte. Voici un exemple de factorisation du fichier *architecte-revit.md*, faisant appel à la modélisation de chaque lot avec la fonction include :



1. La fonction « if »

La fonction if permet, comme en programmation, de mettre des conditions sur l’affichage de certaines parties du texte. Nous avons ici fait une documentation pour une modélisation avec 3 logiciels différents, mais ayant parfois de grandes similitudes. On peut ainsi parfois se permettre de faire un fichier commun, et d’utiliser la fonction if pour les quelques subtilités qui varient selon que l’on utilise un logiciel, ou un autre.

Cela permet par exemple lors de la modélisation d’un objet (Pièce par exemple), de faire un include vers un fichier commun (appelé piece.md), et qui va afficher une documentation adaptée en fonction du logiciel choisi. On a ainsi UN seul fichier qui renseigne sur la modélisation des pièces, et non 3.



Exemple ci-dessus du fichier pieces.md. Le début du fichier parle de généralités sur les pièces, informations communes aux trois logiciels de modélisation, puis commence une boucle if (voir ligne 11).

Pour la syntaxe de la fonction if, voir le lien : <https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet>

1. Les templates

Les templates sont une sorte de page modèle, où l’on peut laisser des blancs à remplir par la suite ; une sorte de texte à trous. L’idée est de créer un template pour une procédure qui va être similaire aux trois logiciels, et laisser des espaces libres pour y inscrire des spécificités propre à chaque logiciel. L’intérêt d’un template est de n’écrire qu’une seule fois la procédure, et de pouvoir l’écrire dans autant de pages qu’on le souhaite (ici 3 pages). Cette fonction est d’autant plus pertinente que la procédure est longue, et varie peu selon les différent logiciels.