Une image contenant personne

Description générée automatiquement

|  |
| --- |
| RÉfÉrentiels  **Ressources et outils FHIR (FSH)**  04/01/2023 |

Sommaire

[Introduction 3](#_Toc123823709)

[Prérequis 3](#_Toc123823710)

[FSH 3](#_Toc123823711)

[Ressources et outils FSH 4](#_Toc123823712)

[SUSHI (SUSHI **Unshortens Short Hand Inputs**) 4](#_Toc123823713)

[Sortie FSH prétraitée (Preprocessed FSH Output) 4](#_Toc123823714)

[GoFSH 5](#_Toc123823715)

[Extension de VS code 6](#_Toc123823716)

[Caractéristiques linguistiques 6](#_Toc123823717)

[Extraits (Snippets) 6](#_Toc123823718)

[Autocomplétions améliorée 6](#_Toc123823719)

[Aller à la définition 7](#_Toc123823720)

[Ouvrir la documentation FHIR 7](#_Toc123823721)

[Tâches 7](#_Toc123823722)

[Compilation et exécution 8](#_Toc123823723)

[Tâches NPM 8](#_Toc123823724)

[Remarque sur les tests 8](#_Toc123823725)

[Mise à jour de la grammaire 8](#_Toc123823726)

[FSH en ligne (FSH Online) 9](#_Toc123823727)

[FSH finder 9](#_Toc123823728)

[FSH School 10](#_Toc123823729)

[GitHub 10](#_Toc123823730)

[Création d’un dépôt 10](#_Toc123823731)

[Création d’une branche 11](#_Toc123823732)

[Créer une branche 11](#_Toc123823733)

[Apporter et commiter des modifications 11](#_Toc123823734)

[Ouverture d’une demande de tirage (pull request) 12](#_Toc123823735)

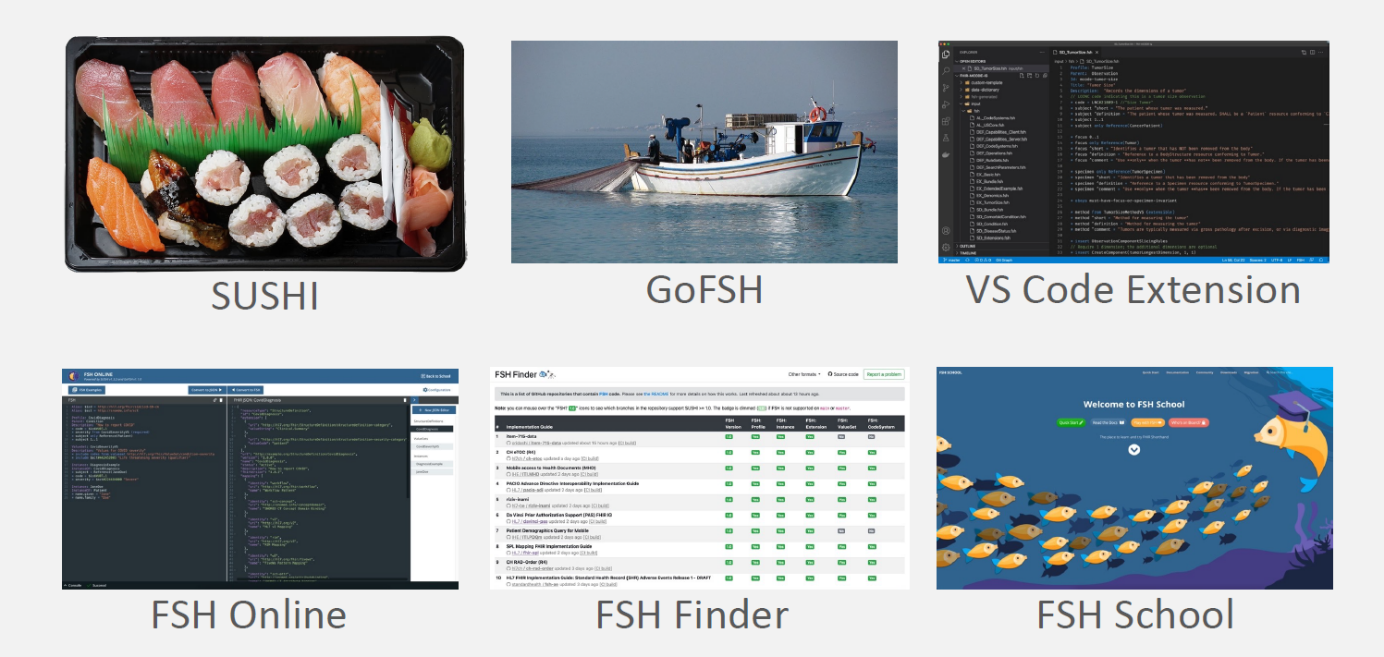
[Fusion de votre demande de tirage 12](#_Toc123823736)

[Conclusion 13](#_Toc123823737)

# Introduction

Dans ce document nous allons expliquer les différents outillage dont un opérateur FHIR de l'ANS a besoin.

Parmi ces outillages on trouve SUSHI, GoFSH, VSCode et ces différents extensions et plugins a installer, GitHub, FSH Online, FSH Finder et FSH School.



Nous allons commencer dans le reste du document par donner un descriptif sur le langage FSH

et expliquer les différents ressources et outils.

# Prérequis

* Installer [Visual Studio Code](https://code.visualstudio.com/) et l'[extension de langage FSH](https://marketplace.visualstudio.com/items?itemName=MITRE-Health.vscode-language-fsh).
* Installer un runtime Java.
* Installer Ruby et Jekyll en utilisant [ces instructions spécifiques au système d'exploitation](https://jekyllrb.com/docs/installation/#guides).
* Installer SUSHI en suivant [ces instructions](https://fshschool.org/docs/sushi/installation/).
* Création d’un compte GitHub

# FSH

FHIR Shorthand (FSH) est un langage spécialement conçu pour définir le contenu des guides de mise en œuvre FHIR (IG). Il est simple et compact, avec des outils pour produire des profils, des extensions et des guides de mise en œuvre (IG) Fast Healthcare Interoperability Resources (FHIR ). Parce qu'il s'agit d'un langage écrit dans des instructions textuelles, FSH encourage le développement distribué en équipe à l'aide d'outils de contrôle de code source conventionnels tels que GitHub. FSH fournit des outils qui permettent de définir un modèle une seule fois et de publier ce modèle sur plusieurs versions de FHIR.

SUSHI (alias "SUSHI Unshortens Short Hand Inputs") est le nom proposé pour l'interpréteur/compilateur de ligne de commande pour FSH.

# Ressources et outils FSH

* Spécification du langage FSH (FSH Language Specification) : norme HL7 FHIR
* SUSHI : compiler FSH en artefacts FHIR
* FSH School : site Web avec documentation, outils, exemples
* FSH en ligne (FSH Online) : raccourci FHIR interactif avec exemples
* GoFSH : conversion des guides de mise en œuvre existants en FSH
* FSH Finder : moteur de recherche Web pour rechercher des projets FSH
* Extension VS Code : surligneur de code pour l'éditeur de code VS
* GitHub : plateforme d’hébergement de code pour la gestion de versions et la collaboration

# SUSHI (SUSHI **Unshortens Short Hand Inputs**)

Le projet SUSHI existe sur GitHub sur l’adresse suivante : <https://github.com/FHIR/sushi>

Prenons l’exemple de la version 2.0.0-beta.1 de SUSHI.

L’installation se fait en ligne de commande par l’exécution de la ligne suivante : npm install -g [fsh-sushi@2.0.0-beta.1](mailto:fsh-sushi@2.0.0-beta.1)

Les nouveautés apportées par SUSHI sont :

* Les modèles logiques et ressources (Logical Models & Resources)
* Les chemins d'indentation (Indent paths)
* Hiérarchie des concepts et règles du caret (Concept Hierarchy & Caret Rules)
* Sortie FSH prétraitée (Preprocessed FSH Output)
* Changements :
* L'espace avant \* est maintenant significatif
* Syntaxe obsolète non prise en charge
* Format de projet hérité non pris en charge

## Sortie FSH prétraitée (Preprocessed FSH Output)

Dans cette section il faut lancer la commande : sushi -p myproject pour avoir la sortie FSH (preprocessed).

Ci-après un exemple qui montre les différences entre le fichier en entrée et celui en sortie :

|  |
| --- |
|  |

Le document SUSHI existe sur GitHub sur le lien suivant :

<https://github.com/nizarbs2508/SUSHI-FHIR> .

Ce document s’appel «FHIR\_WITH\_SUSHI.docx»

# GoFSH

Le projet GoFSH existe sur GitHub sur l’adresse suivante : <https://github.com/FHIR/GoFSH>

La version actuelle de GoFSH est 1.x.

L’installation se fait en ligne de commande par l’exécution de la ligne de commande suivante :

* npm install -g gofsh

Les nouveautés apportées par GoFSH sont :

* Interface de ligne de commande
* Conversion FHIR JSON/XML en FSH
* Support de plusieurs format en sortie
* Prise en charge FSH STU1 et indexation douce
* Permet le développement des :
* Modèles logiques et ressources (Logical Models & Resources)
* Systèmes de code hiérarchiques (Hierarchical CodeSystems)
* Règles du caret conceptuel (Concept Caret Rules)
* Chemins d'indentation (Indent paths)

Le document GoFSH existe sur GitHub sur le lien suivant : <https://github.com/nizarbs2508/GoFSH-FHIR>. Ce document s’appel «GoFSH.docx»

# Extension de VS code

IL faut tout d’abord commencer par installer Visual Studio Code Editor.

Pour installer l’extension « vscode-language-fsh » il faut aller sur la page <https://marketplace.visualstudio.com/items?itemName=MITRE-Health.vscode-language-fsh>

et cliquer sur « Install ».

Il s’agit d’une extension de prise en charge linguistique pour le langage FHIR Shorthand (FSH).

Une fois activées, les fonctionnalités de cette extension devraient être automatiquement implémentées.

## Caractéristiques linguistiques

Les fichiers FSH ont automatiquement la syntaxe appliquée. Cela permet une lecture et une écriture plus faciles de la sténographie FHIR.

## Extraits (Snippets)

Les extraits FSH facilitent la création de nouveaux éléments FSH ! Les extraits ajoutent automatiquement des mots clés et des espaces réservés pertinents afin que vous puissiez facilement saisir toutes les métadonnées recommandées pour une définition FSH. Les extraits créeront même automatiquement l’**Id** etle **Title** pour nous en fonction du nom.

Pour utiliser des extraits, nous saisissons l'une des phrases de déclenchement, appuyons sur la touche <TAB> ou <ENTRÉE>, saisissons la première valeur de champ et appuyons sur la touche <TAB> pour passer à l'espace réservé suivant. Tous les extraits FSH s'arrêtent toujours à la première règle de la définition.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Trigger** | **FSH Item** | **Keywords** | | pro | Profile | Profile, Parent, Id (auto), Title (auto), Description | | ext | Extension | Extension, Id (auto), Title (auto), Description | | log | Logical | Logical, Parent, Id (auto), Title (auto), Description | | res | Resource | Resource, Parent (choice), Id (auto), Title (auto), Description | | vs | ValueSet | ValueSet, Id (auto), Title (auto), Description | | cs | CodeSystem | CodeSystem, Id (auto), Title (auto), Description | | inst | Instance | Instance, InstanceOf, Usage (choice), Title (auto), Description | | inv | Invariant | Invariant, Description, Expression, Severity (choice), XPath | | map | Mapping | Mapping, Source, Target, Id, Title (auto), Description | | rs | RuleSet | RuleSet | |

De plus, un extrait est fourni pour aider lors de l'écriture des règles de découpage. Lorsque le début de la phrase **^slicing** est détecté, un bloc de règles peut être inséré pour les chemins couramment définis lors de la définition du découpage sur un élément.

### Autocomplétions améliorée

Les définitions d'entités FSH ont leurs noms fournis en tant que résultats de saisie semi-automatique dans des scénarios contextuellement appropriés.

* Après le mot clé - **Parent**, les résultats incluront les noms **Profile**, **Extension**, **Logical** et **Resource** selon le type d'entité en cours de définition.
* Après le mot clé - **InstanceOf**, les résultats incluront les noms **Profile**, **Extension** et **Resource**.
* Lors de l'écriture d'une règle **obeys**, les résultats incluront des noms **Invariant**.

Lors de l'écriture des règles qui s'appliquent à un élément, des chemins d'accès aux éléments seront suggérés. Il existe certaines limitations connues à cette fonctionnalité :

* Les éléments d'achèvement ne sont fournis que pour les règles sur un **Profile**, **Extension**, **Logical**, **Resource** ou **Instance**.
* Les noms de tranche ne sont pas inclus dans les éléments d'achèvement.
* Les éléments de complétion ne sont pas fournis pour les règles en retrait.
* Si un type est supprimé d'un élément de choix avec une **only** règle, les éléments d'achèvement seront toujours proposés pour les types supprimés.
* Les éléments ajoutés à une règle FSH **Logical** ou **Resource** par une règle FSH ne seront pas disponibles en tant qu'éléments de complétion pour d'autres règles.
* Seul le chemin de l'élément principal aura des éléments d'achèvement fournis. Une règle de caret n'aura pas d'éléments d'achèvement fournis pour le chemin du caret.

## Aller à la définition

Les définitions d'entités FSH dans l’espace de travail peuvent être trouvées partout où leur nom est utilisé. Pour accéder à la définition de l'entité, faites un clic droit sur le nom de l'entité et sélectionner "Aller à la définition" dans le menu contextuel. Ou, nous pouvons appuyer sur <F12> lorsque le curseur de texte se trouve sur le nom de l'entité.

## Ouvrir la documentation FHIR

Les pages de documentation peuvent être ouvertes directement à partir des fichiers FSH. Cliquer avec le bouton droit sur le nom d'une ressource FHIR ou d'un mot-clé FSH, puis sélectionner "Ouvrir la documentation FHIR" dans le menu contextuel.

## Tâches

L'extension fournit une tâche personnalisée pour exécuter SUSHI sur l'espace de travail actuel. La tâche exécutera SUSHI sur l'espace de travail, consignera les messages dans l'onglet Terminal intégré de VS Code, signalera les erreurs ou les avertissements dans l'onglet Problèmes de VS Code et mettra en évidence les erreurs et les avertissements en ligne dans le fichier FSH. La sélection d'une erreur ou d'un avertissement dans l'onglet Problèmes ouvrira le fichier dans lequel se trouve l'erreur.

Il faut notez qu'après la résolution des erreurs ou des avertissements dans le FSH, SUSHI doit être exécuté à travers la tâche pour les résoudre dans l'onglet Problèmes et en ligne dans le fichier FSH.

Pour exécuter la tâche Build SUSHI, il faut utiliser la fonction Exécuter la tâche de VS Code. La fonction "Exécuter la tâche" est accessible dans le menu "Terminal" en cliquant sur "Exécuter la tâche". Lorsque le menu s'ouvre dans VS Code, sélectionner 'fsh' puis sélectionner 'sushi'. La tâche peut également être exécutée à l'aide du raccourci clavier pour exécuter les tâches de construction, qui est ⇧⌘B sur Mac et Ctrl+Maj+B sur Windows.

Il faut noter que SUSHI doit être installé localement pour que la tâche s'exécute correctement.

### Compilation et exécution

Exécuter npm install dans ce dossier. Cela installe tous les modules npm nécessaires dans le dossier client et serveur ouvrez VS Code sur ce dossier.

Basculer vers la viewlet de débogage.

Sélectionner Extension dans le menu déroulant.

## Tâches NPM

Les tâches NPM suivantes peuvent être utiles pour le développement :

| **Tâche** | **Description** |
| --- | --- |
| **Construire** | compile src/\*\*/\*.tsles fichiers en out/\*\*/\*.jsfichiers à l'aide du compilateur TypeScript (tsc) |
| **construire:regarder** | similaire à *build* mais construit automatiquement lorsque des modifications sont détectées dans les fichiers src |
| **Peluche** | vérifie tous les fichiers src pour s'assurer qu'ils respectent les styles et les règles du code du projet |
| **peluche:réparer** | corrige les erreurs de charpie lorsque des correctifs automatiques sont disponibles pour eux |
| **plus jolie** | vérifie tous les fichiers src pour s'assurer qu'ils respectent les conventions de formatage du projet |
| **plus joli :réparer** | corrige de plus belles erreurs en réécrivant les fichiers en utilisant les conventions de formatage du projet |
| **Test** | exécute la suite de tests |
| **Chèque** | exécute toutes les vérifications effectuées dans le cadre de ci (peluche, plus joli, test) |

Pour exécuter l'une de ces tâches, utilisez npm run. Par exemple:

|  |
| --- |
| $ npm run build:watch |

### Remarque sur les tests

La tâche **test** produira souvent (mais pas toujours) une erreur si elle est exécutée alors que VS Code est ouvert. Si nous avons cette erreur, fermer VS Code et exécuter la tâche à partir de la ligne de commande.

## Mise à jour de la grammaire

Le **src/lang/** répertoire contient le lexer et parseur FSH. Lorsque la grammaire FSH change, les fichiers de ce répertoire devront être mis à jour. Ces fichiers sont générés dans le projet SUSHI en exécutant ANTLR sur les fichiers grammaire de définition. Les fichiers sont ensuite copiés du projet SUSHI vers ce projet. Les fichiers à copier sont :

* FSH.tokens
* FSHLexer.tokens
* FSHLexer.js
* FSHListener.js
* FSHParser.js
* FSHVisitor.js

# FSH en ligne (FSH Online)

Le lien pour exécuter le FSH en ligne : <https://fshschool.org/FSHOnline/#/>

C’est un outil utilisé pour :

* Convertir les fichier FSH en FHIR
* Convertir les fichiers FHIR en FSH
* Parcourir et copier les exemples FSH
* Partager des définitions FSH avec d’autres

Cet outil est recommandé pour

* Essayer rapidement certains FSH
* Convertir rapidement FHIR en FSH
* Déboguer les définitions FSH
* Signaler les erreurs

# FSH finder

Le lien pour le **finder** est <https://fshschool.org/fsh-finder>

Il s'agit d'une liste de référentiels GitHub contenant du code FSH. Seuls les référentiels qui utilisent la configuration FSH ou SUSHI sur la branche par défaut sont inclus. Consulter le README pour plus de détails sur la façon dont cela fonctionne.

Plus de 100 guides de mise en œuvre :

* Etats-Unis
* Nouvelle-Zélande
* Suisse
* Belgique
* Danemark
* Suède
* OMS
* Da Vinci
* Covid SANER, Logica
* Bouton bleu CARIN
* Certificat de vaccination SMART…

# FSH School

Le lien pour FSH school esi le suivant <https://fshschool.org/>

Il s’agit d’un portail qui ramène à de nombreux autres sous liens :

* FSH en ligne (FSH Online)
* FSH Finder
* La documentation SUSHI
* La documentation GoFSH
* Plusieurs autres tutoriels
* Présentations
* Plusieurs autres ressources

# GitHub

GitHub est une plateforme d’hébergement de code pour la gestion de versions et la collaboration. Cette plateforme vous permet de travailler avec d’autres personnes sur des projets depuis n’importe où.

Dans cette section nous allons présenter les bases de GitHub comme les dépôts, les branches, les commits et les demandes de tirage (pull requests).

Dans ce guide de démarrage rapide, nous allons :

* Créer et utiliser un dépôt
* Démarrer et gérer une nouvelle branche
* Apporter des modifications à un fichier et les pousser vers GitHub sous forme de commits
* Ouvrir et fusionner une demande de tirage (pull request)

## Création d’un dépôt

Un dépôt s’utilise généralement pour organiser un seul projet. Les dépôts peuvent contenir des dossiers et des fichiers, des images, des vidéos, des feuilles de calcul et des jeux de données, tout ce dont le projet a besoin.

Souvent, les dépôts incluent un fichier README, à savoir un fichier qui contient des informations sur le projet. Les fichiers README sont écrits en langage Markdown en texte brut.

Nous pouvons utiliser cet aide-mémoire pour bien démarrer avec la syntaxe Markdown.

GitHub permet d’ajouter un fichier README en même temps que nous créons notre dépôt.

1. Naviguer vers l’url [www.github.com](http://www.github.com) et s’identifier
2. En haut à droite d’une page, utiliser le menu déroulant et sélectionner Nouveau dépôt.
3. Dans la zone Nom du dépôt, entrer le nom du répositorie.
4. Dans la zone Description, écrivons une brève description.
5. Sélectionner Ajouter un fichier README.
6. Indiquer si notre dépôt sera public ou privé.
7. Cliquer sur Créer le dépôt.

## Création d’une branche

La création de branches permet d’avoir différentes versions d’un dépôt en même temps. Par défaut, un dépôt a une seule branche nommée « main » considérée comme la branche définitive. Nous pouvons créer des branches supplémentaires séparées de « main » dans notre dépôt. Nous pouvons utiliser des branches pour avoir différentes versions d’un projet en même temps. Cela s’avère utile quand nous voulons ajouter de nouvelles fonctionnalités à un projet sans modifier la source principale du code. Le travail effectué sur les différentes branches n’apparaît pas sur la branche « main » tant que nous ne l’avons pas fusionné, ce que nous allons aborder plus tard dans ce guide. Nous pouvons utiliser des branches pour expérimenter et apporter des modifications avant de les commiter dans « main ». Quand nous créons une branche en dehors de la branche « main », nous effectuons une copie, ou un instantané, de cette branche « main » telle qu’elle est à ce moment-là. Si une autre personne apporte des modifications à la branche « main » alors que nous sommes en train de travailler sur notre branche, nous pouvons tirer (pull) ces mises à jour.

### Créer une branche

1. Cliquer sur l’onglet Code de notre dépôt.
2. Cliquer sur la liste déroulante située en haut de la liste des fichiers qui indique « main ».
3. Taper un nom de branche dans la zone de texte.
4. Cliquer sur Créer une branche : à partir de « main ».

Maintenant, nous avons deux branches, « main » et une deuxième. Pour le moment, elles sont strictement identiques. Par la suite, nous allons ajouter des modifications à la nouvelle branche.

## Apporter et commiter des modifications

Quand nous avons créé une branche à l’étape précédente, GitHub nous a amené à la page de codes de notre nouvelle branche «xxx», qui est une copie de « main ».

Nous pouvons apporter et enregistrer des modifications dans les fichiers de notre dépôt. Sur GitHub, les modifications enregistrées sont appelées commits. Chaque commit est associé à un message de commit qui décrit pourquoi une modification particulière a été apportée. Les messages de commit capturent l’historique de nos modifications afin que d’autres contributeurs puissent comprendre ce que nous avons fait et pourquoi.

1. Sous la branche «xxx »  que nous avons créée, cliquer sur le fichier README.md.
2. Cliquer sur le crayon pour modifier le fichier.
3. Dans l’éditeur, présentons-nous en quelques mots. Essayer d’utiliser différents éléments Markdown.
4. Dans la zone Commiter les modifications, écrire un message de commit qui décrit les modifications.
5. Cliquer sur Commiter les changements.

Ces modifications sont uniquement apportées au fichier README de notre branche « xxx ». Le contenu de cette branche est donc maintenant différent de celui de « main ».

## Ouverture d’une demande de tirage (pull request)

Maintenant que nous avons des modifications dans une branche distincte de « main », nous pouvons ouvrir une demande de tirage (pull request).

1. Cliquer sur l’onglet Demandes de tirage de le dépôt.
2. Cliquer sur Nouvelle demande de tirage.
3. Dans la zone Exemples de comparaisons, sélectionner la branche que nous avons créée pour la comparer au « main » (l’originale).
4. Examiner les modifications dans les différences figurant dans la page Comparer pour vérifier qu’elles correspondent bien à ce que nous voulons soumettre
5. Cliquer sur Create pull request (Créer une demande de tirage).
6. Donner un titre à notre demande de tirage et décrire brièvement les modifications. Nous pouvons inclure des emojis et faire glisser des images et des gifs.
7. Cliquer éventuellement, à droite du titre et de la description, sur située en regard de Réviseurs, Destinataires, Étiquettes, Projets ou Jalons pour ajouter l’une de ces options à la demande de tirage. Nous n’avons pas besoin d’en ajouter à ce stade, mais ces options nous offrent différentes façons de collaborer en utilisant des demandes de tirage.
8. Cliquer sur Create pull request (Créer une demande de tirage).

Les collaborateurs peuvent maintenant passer en revue vos modifications et faire des suggestions.

## Fusion de votre demande de tirage

Dans cette dernière étape, nous allons fusionner notre branche dans la branche main. Une fois que nous avons fusionné notre demande de tirage, les modifications apportées à notre branche sont incorporées dans « main ».

Parfois, une demande de tirage peut introduire dans le code des modifications qui entrent en conflit avec le code existant sur main. En cas de conflits, GitHub nous envoie une alerte sur le code en conflit et empêchent toute fusion tant que les conflits ne sont pas résolus. Nous pouvons effectuer un commit qui résout les conflits ou utiliser des commentaires dans la demande de tirage pour discuter des conflits avec les membres de notre équipe.

Dans cette procédure pas à pas, nous ne devons pas avoir de conflits. Nous sommes donc prêt à fusionner notre branche dans la branche main.

1. Cliquer sur Fusionner la demande de tirage pour fusionner les modifications dans main.
2. Cliquer sur Confirmer la fusion. Nous allons recevoir un message indiquant que la demande a été correctement fusionnée et fermée.
3. Cliquer sur Supprimer la branche. Maintenant que notre demande de tirage est fusionnée et que nos modifications sont dans main, nous pouvons supprimer en toute sécurité la branche. Si nous voulons apporter d’autres modifications à notre projet, nous pouvons toujours créer une branche et répéter ce processus.

# Conclusion

Dans ce document nous avons présenté l'outillage complet dont un opérateur FHIR de l'ANS a besoin: Sushi, GoFsh, VSCode, les plugins a installer, GitHub etc.. Ce tutoriel pourrait être un document chapeau par lequel un néophyte de l'ANS va commencer à apprendre la création d'un guide de mise en œuvre avec FHIR Shorthand FHIR (FSH) qui est un langage convivial pour définir le contenu des IG (implémentation guide) et, plus récemment, des modèles logiques et des ressources FHIR.

Ce document est publié sur GitHub en cliquant sur le lien suivant : <https://github.com/nizarbs2508/OUTILS-FHIR-FSH>