



DOCUMENTATION

Plugin Creadis
GIRDER

CREATIS

Table of contents

Context (en)

I. CNRS & CREATIS

- a. The national center for scientific research p.05
- b. A laboratory among many others p.05

II. Issues

- a. Problematic p.06
- b. Goals p.06

III. Application

- a. Functional description p.07
- b. Virtual Imaging Platform p.08
- c. Girder p.09
- d. Launching an application p.10
 - i. *Api key of VIP* p.10
 - ii. *Selected files to be processed* p.10
 - iii. *List of applications* p.11
 - iv. *Initialize the launch of the application* p.12
- e. Fetch results p.13
 - i. *A history of status* p.13
 - ii. *History of executions* p.14

IV. Attachments

- a. Laboratories and financiers p.16
- b. About plugin p.16

Le plugin Creatis (fr)

I. Pré-requis

II. Client

- a. Cookbook

III. Serveur

- a. Cookbook

IV. Annexes

Context



In this first part, we will explain the role of the plugin and how to use it.

ISSUES

PROBLEMATIC

CREATIS provides a **free public web application** for researchers and doctors around the world. This application is called « **VIP** » for Virtual Imaging Platform. The Virtual Imaging Platform is a web portal for medical simulation and image data analysis. It leverages resources available in the biomed Virtual Organisation of the European Grid Infrastructure to offer an open and performant service to academic researchers worldwide.

Today, VIP users have an integrated distributed file system to the web portal. Nevertheless, there is a growing demand to **access to data** recorded in other platforms in a **transparent manner**. The laboratory wants to improve the user experience to use VIP in a simpler way, with better **data management**.

GOALS

On one side, we have the VIP application that offers image processing applications, and on the other side, we have Girder, who is a free and open source web-based data management platform.

The goal is to make an **easy-to-use** web application linking the **VIP** application and the **Girder** platform. To be more precise, we need to realize a **plugin** « CREATIS » integrating on Girder. This will enable a more transparent management of medical data, while using the image processing applications offered by VIP.

APPLICATION

FUNCTIONAL DESCRIPTION

Name : GIRDER – Plugin Creadis

Denominated : Data management web tool to use VIP applications

Technologies used : Backbone, Underscore.js, jQuery, Pug, Mongoddb, Python, Java

Girder is a data management web platform and VIP is a web platform who that provides applications as services.

This Girder plugin allows to interacts with VIP and is divided into 2 main parts. Here are their functionalities which are explained further in the document.

Launching an application on VIP from Girder

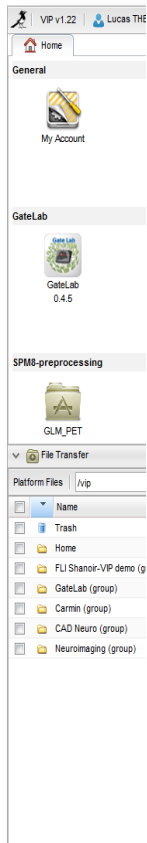
- Use the VIP API
- Select the application to use
- Send all the selected files to process
- Initialize the application with the arguments provided by the user

Fetch results

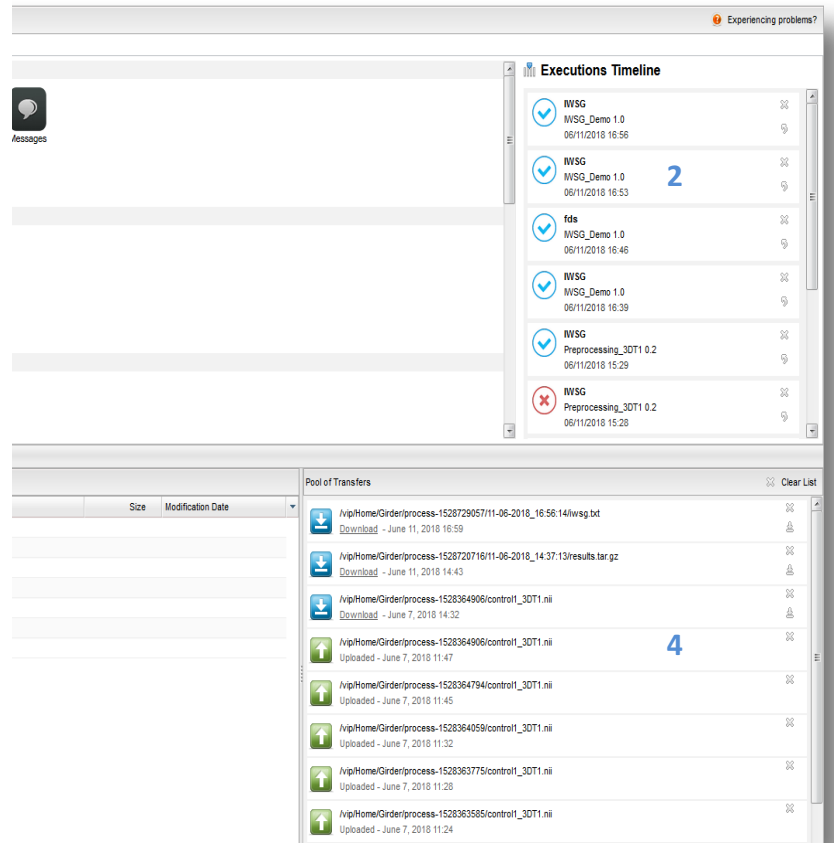
- Get a history of the processes launched by the user
- Update the status of launched executions
- Fetch the results of the finished executions
- Store the results on the Girder side

VIRTUAL IMAGING PLATFORM

The Virtual Imaging Platform (VIP) is a web portal freely accessible for researchers. It's a **medical imaging treatment application manager**.



VIP Home page screenshot



Main page contents :

1. Menu and list of applications available to the logged in user
2. All the executions launched by the user
3. The files stored on EGI Infrastructure
4. Upload and download timeline

EGI delivers advanced computing services to support scientists, multinational projects and research infrastructures. Without the EGI, VIP couldn't have the necessary computing power and storage capacities for the different images processing applications it provides.

GIRDER

Girder is a free and open source web-based **data management platform** developed by Kitware as part of the Resonant data and analytics ecosystem. What does that mean? Girder is both a standalone application and a platform for building new web services. It's meant to enable quick and easy construction of web applications that have some or all of the following requirements :



Data organization

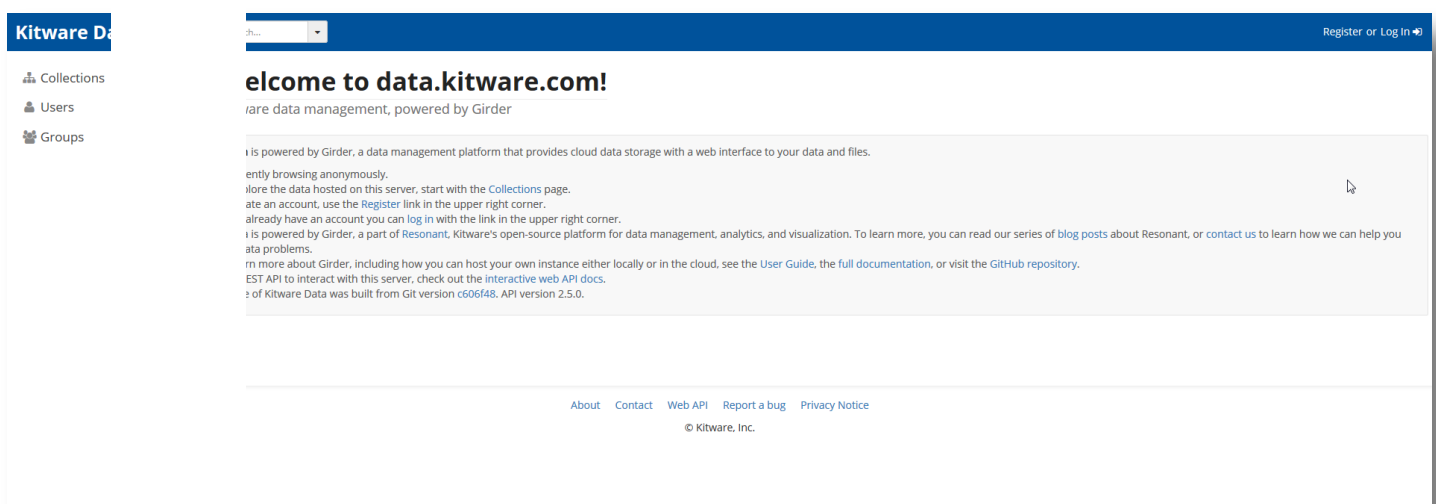


User management and authentication



Authorization management

Example of the Girder interface, without the Creatis plugin :



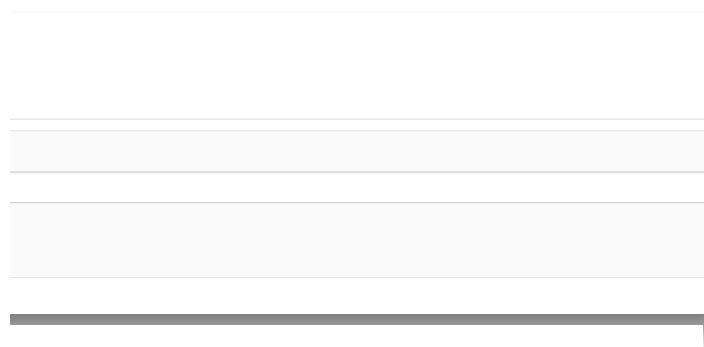
Girder classical home page

LAUNCHING AN APPLICATION

API KEY OF VIP

Communication with the VIP platform is done through a **REST API** called **CARMIN**. It's an interface to execute pipelines (applications) remotely, and also share data. This API provides all that we need to control VIP from the Girder plugin. For example, we can ask the API for the list of available applications, the status of an execution.

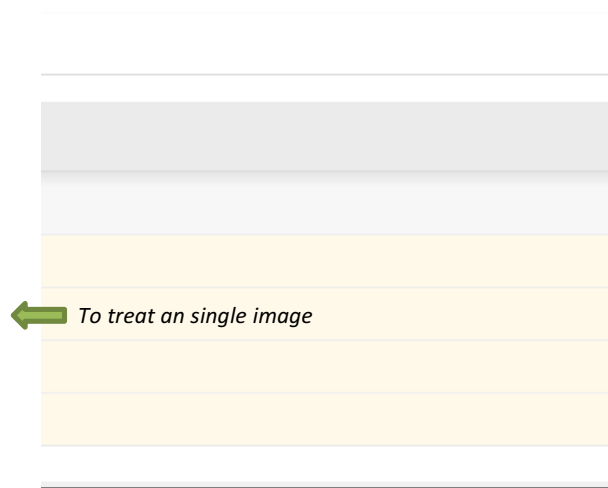
The API authentication is ensured with an API key. The user must enter his VIP api key in his Girder account settings. All the steps are explained on Girder's main page. The entered key is validated by the server to verify it is valid on VIP. **Without this key, the Creatis plugin is unusable.**



Account settings

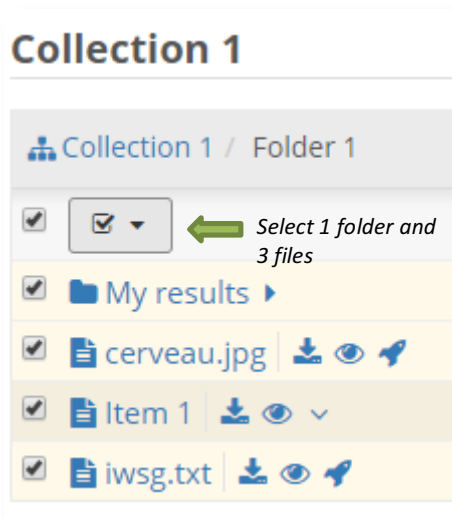
SELECT FILES TO BE PROCESSED

Now that we have filled in the API key, we can communicate with VIP. If the user wants **to process a single file**, he must click on the rocket icon.

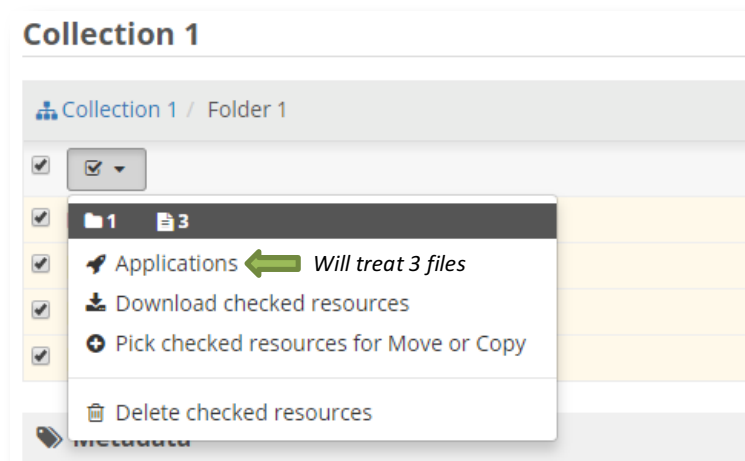


Content of folder in Girder

Some applications expect multiple input files. This is the case for example, with the « Preprocessing FLAIR » who segments and normalizes the first image and the second image. The user can therefore through Girder, **select as many files** as he wants to process.



Selected many files



Actions to selected files

LIST OF APPLICATIONS

Once the user has chosen the files that he wants to treat. The list of **applications it has access to** is displayed.

Creatis			
Quick search...			
<ul style="list-style-type: none"> Collections Users Groups Admin console 	Applications		
	3 selected files		
	#	Application	Version
	1	AdditionTest	0.9
	2	Demo_pipeline	0.1
	3	Freesurfer (recon-all)	0.3.6
	4	Freesurfer (recon-all, longitudinal)	0.3.5
	5	FSL - BEDPOSTX	0.2
	6	FSL - bet	0.2
	7	FSL - fast	0.2
	8	FSL - FIRST	0.2
	9	FSL - FLIRT	0.2
	10	FSL - slena	0.2
	11	FSL - slenax	0.2
	12	GateLab	0.4.6
	13	GLM_PET	0.1
	14	GrepTest	1.2
Execution			
			Run it
			Run it
			Run it
			Run it
			Run it
			Run it
			Run it
			Run it
			Run it
			Run it
			Run it

List of applications

INITIALIZE THE LAUNCH OF THE APPLICATION

The user **chooses the version** of the application that he wants to use and clicks on the button « Run it ».



Modal to fill the application parameters

He fills the parameters asked and clicks on the button « Execute ».

Noted that the tab « **General** » corresponds to the **arguments Girder side**. All other tabs are the arguments requested by the application, VIP side.

The process is now launched on VIP.

FETCH RESULTS

A HISTORY OF STATUS

Queries are made regularly to ask VIP **the status of an execution**. As soon as a status is changed, the status of the execution in Girder is directly updated. There can be **8 different status** (VIP side) :

- *Initializing*
- *Ready*
- *Running*
- *Finished*
- *InitializationFailed*
- *ExecutionFailed*
- *Unknown*
- *Killed*

If the execution is in the 'Finished' state, it means that the execution **generated the results** in VIP. In this case, we make a request to VIP to **download the results** on Girder and store them in the folder that the user choose in one of the arguments when launching the application (tab « General »).

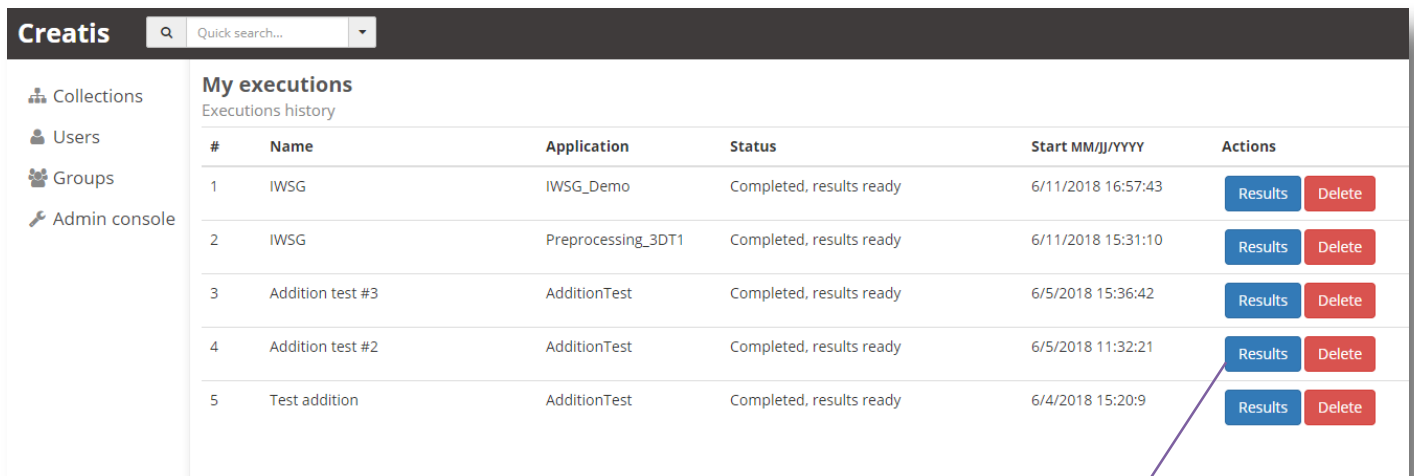
In addition to 8 status given by VIP, 2 status are added in Girder :

- *Fetches*
- *NotFetches*

To indicate whether the results have been retrieved or not.

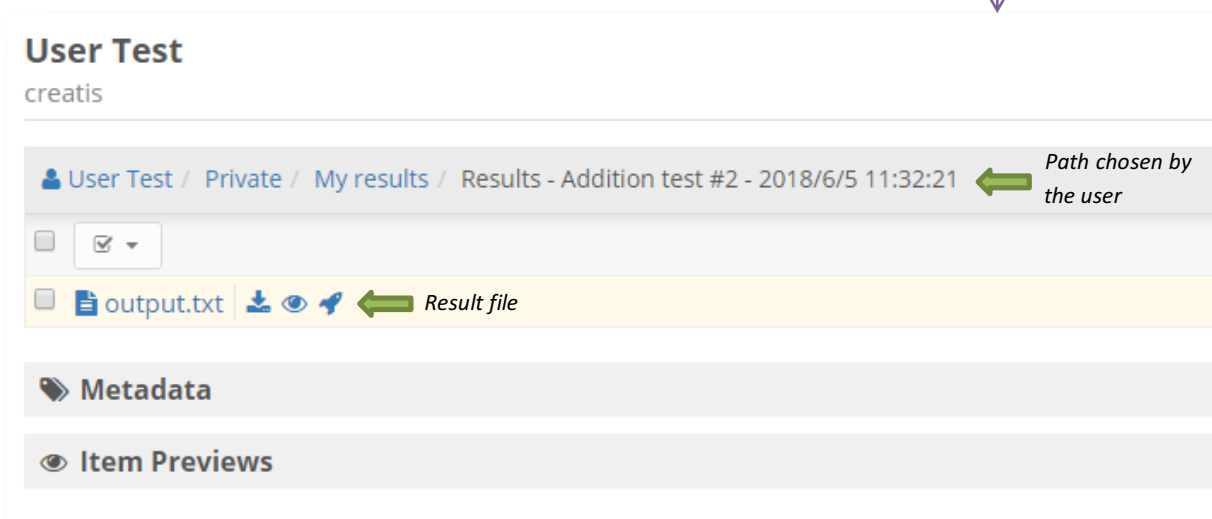
HISTORY OF EXECUTIONS

All **executions initiated** by the user are **saved** in the **database**. So we can give him a history of all these executions. In addition to having the list of these executions, actions are proposed by execution. It can delete an execution, or go to the results folder (if they are generated).



#	Name	Application	Status	Start MM/JJ/YYYY	Actions
1	IWSG	IWSG_Demo	Completed, results ready	6/11/2018 16:57:43	Results Delete
2	IWSG	Preprocessing_3DT1	Completed, results ready	6/11/2018 15:31:10	Results Delete
3	Addition test #3	AdditionTest	Completed, results ready	6/5/2018 15:36:42	Results Delete
4	Addition test #2	AdditionTest	Completed, results ready	6/5/2018 11:32:21	Results Delete
5	Test addition	AdditionTest	Completed, results ready	6/4/2018 15:20:9	Results Delete

History of executions



User Test
creatis

[User Test](#) / [Private](#) / [My results](#) / [Results - Addition test #2 - 2018/6/5 11:32:21](#) ← *Path chosen by the user*

☐ ☒ ☐

☐ output.txt ← *Result file*

Metadata

Item Previews

Folder containing the results generated by the execution

Le plugin Creatis



L'objectif de ce plugin est d'utiliser les applications de VIP depuis la plateforme Girder. L'utilisateur n'aura donc plus besoin de passer par VIP pour lancer une application de traitement d'images.

Dans cette partie nous allons voir comment se déroule le développement de Girder et plus particulièrement du plugin VIP. Vous aurez à partir de là un minimum d'outils pour faire évoluer le plugin.

PRE-REQUIS

[Installer les pré-requis](#)

[Installer Girder](#)

[Plugin Creadis](#)

Lancer mongodb en tache de fond :

```
mongod &
```

Lancer le serveur Girder en tache de fond :

```
girder serve &
```

Accès par défaut :

<http://localhost:8080>

Installer le plugin sur Girder :

- `git clone https://github.com/Saloukai/VIP-girder-plugin.git vip`
- `girder-install -s plugin /path/to/vip/plugin`

/! Le nom du plugin ne doit pas contenir de tiret : « - »

Si le repo existe déjà et qu'il y a eu un changement sur un tag, penser à faire :

```
pull --tags
```

Pour activer le plugin sur Girder, il faut se connecter en admin sur la plateforme Girder puis se rendre dans *Admin > Plugin*. Passer en 'on' le plugin VIP et cliquer sur le bouton '*Rebuild web code and restart the server*'. Si le rebuild génère une erreur, c'est sûrement dû au npm du rebuild, il faut donc faire la commande manuellement. Pour cela aller voir les logs qui se trouvent dans `~/girder/error.log`

Pour mettre à jour automatiquement le client à chaque modification d'un fichier du plugin :

```
girder-install web --watch-plugin vip
```

Pour lancer le serveur Girder et déployer le plugin sur le serveur *brain-perfusion*

Accéder au virtual-env

- `~/girder_env/bin/activate`

Connecter Girder à la db

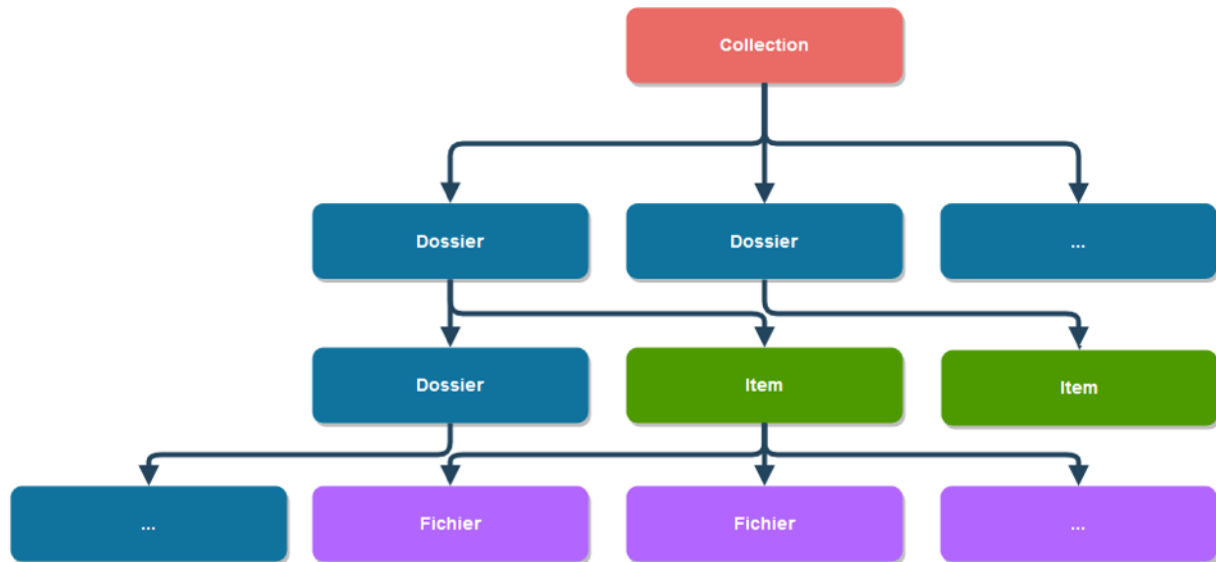
- `mongod --dbpath ~/data/`

Lancer Girder

- `python -m girder`

A savoir...

- Girder utilise le framework **Backbone.js**
- La base de données utilisée est **MongoDB**. MongoDB est un système de gestion de base de données orientée documents (format **JSON**) qui ne nécessite pas de schéma prédéfini des données.
- L'arborescence des données dans Girder s'organise comme décrit ci-dessous :



- Comme on peut le voir, il y a une notion d'item. Un item est composé d'un ou plusieurs fichiers. Il faut retenir que :
 - 1 collection peut contenir seulement des dossiers
 - 1 dossier peut contenir des dossiers, des items et des fichiers
 - 1 fichier = 1 item
 - 1 item = 1 ou + fichiers
- API Girder : /api/v1 (par défaut)
- [CARMIN](#) : Common API for Research Medical Imaging Network
 - API RESTful qui communique avec VIP
- Pipeline = application VIP
- Si le développeur **travaille** en **local**, le développeur doit désactiver les sécurités des navigateur car **l'authentification CORS** est impossible en localhost. Pour cela, pendant le développement et **seulement pour le développement**, pour ne pas avoir le problème du cross-origin resource sharing (**CORS**), ouvrir un terminal et faire la commande pour ouvrir google chrome sans sécurité:

- **Windows :** 'Program Files
(x86)\Google\Chrome\Application\chrome.exe' --user-data-dir -
disable-web-security
- **Linux :** google-chrome-stable -user-data-dir --disable-web-
secutiry

CLIENT

Arborescence vip/

Dossier / Fichiers		Description	Technologie	Extension
./		-	-	-
./	.gitignore	Fichier github pour ignorer certains type de dossiers et/ou fichiers lors d'un commit.	-	-
./	REAMDE.md	Fichier github qui décrit un projet git et dans notre cas qui décrit le plugin.	Markdown	.md
./	package.json	Fichier contenant la liste des dépendances du plugin. Lors d'un ' <i>npm install</i> ', c'est grâce à ce fichier que npm sait quel paquet installer.	JSON	.json
./	plugin.js	Fichier qui décrit le plugin, ce fichier est nécessaire pour le fonctionnement du plugin sur Girder. C'est grâce à ce fichier que Girder sait s'il y a un nouveau plugin.	JSON	.json
./	webpack.helper.js	Ce fichier règle le problème suivant : npm n'arrivait pas installer le paquet ' <i>compare-versions</i> '.	Javascript	.js
/example		Contient les fichiers d'exemples cités dans la documentation.	-	-
/server		Contient les fichiers concernant le côté serveur du plugin.	Python	.py
/web_client		Contient les fichiers concernant le côté client du plugin.	Javascript	.js

CLIENT

Arborescence /web_client

Dossier / Fichiers		Description	Technologie	Extension
./		Contient les dossiers et fichiers du client.	-	-
./	constants.js	Fichier contenant des variables constantes.	Javascript	.js
./	main.js	Fichier d'entrée du plugin. Contient tous les imports de nos fichiers.	Javascript	.js
./	routes.js	Fichiers qui décrivent nos routes (nouvelles pages du plugin).	Javascript	.js
./	utilities.js	Fichier contenant des fonctions utiles qui sont utilisées dans plusieurs vues.		
/collections		Décrit pour chaque fichier une table dans la db. Collection = table pour MongoDB.	Javascript	.js
/models		La couche modèle est responsable de l'interaction avec la db. C'est une entité de la collection, elle est complémentaire à la collection (table).	Javascript	.js
/stylesheets		Préprocesseur pour le css. Décrit le style de la page.	<u>Stylus</u>	.styl
/templates		Préprocesseur pour l'HTML. Décrit la page HTML qui sera appelée par la vue.	<u>Pug</u>	.pug
/views		Chaque fichier correspond à une vue dans l'application. Il y a 2 façons de l'utiliser : <ul style="list-style-type: none">- En utilisant la fonction 'wrap' dans le cas où l'on souhaite modifier une vue déjà existante.- En créant une extension de View pour créer une nouvelle vue.	Javascript	.js
/vendor		Contient des dépendances qui ne sont pas gérées par npm.	-	-

Description /web_client/views

Fichier	Wrap d'une vue existante ?	Description	Templates rattaché	Vue rattachée	Styles rattaché
CheckedMenuWidget.js	✓	Vue qui implémente le bouton 'Applications' dans le menu d'actions multifiiles. Redirige vers ListPipelinesMultiFiles quand on clique sur le bouton 'Applications'.	checkedActions Menu.pug	CheckedMenuWidget.js (/girder/views/widgets) ListePipelinesMultifiiles.js (vue du plugin)	-
ConfirmPipelineDialog.js	✗	Modal qui s'ouvre quand on lance une pipeline pour renseigner les arguments demandés (nom de l'exécution, le result-directory...). Seulement dans le cas où l'utilisateur choisit de traiter une seule image.	confirmPipeline Dialog.pug	View.js (/girder/views) <i>Pour créer une nouvelle vue, il faut qu'elle hérite de la vue View.js</i>	-
ConfirmPipelineDialogMultiFiles.js	✗	Même rôle que la ligne précédente, cependant c'est dans le cas où l'utilisateur a sélectionné plusieurs fichiers à traiter.	confirmPipeline Dialog.pug	View.js (/girder/views)	-
FolderView.js	✓	Dans la liste des fichiers se trouvant dans un dossier, ajoute l'icône 'Rocket' (fusée) pour rediriger vers la page des applications. Si l'item est composé d'un seul fichier, l'icône apparaît devant le nom du fichier. Si l'item est composé de plusieurs fichiers, une liste déroulante est créée pour pouvoir lister les fichiers et ajouter l'icône 'Rocket'.	buttonListPipeline.pug selectImage.pug collapseImage.pug	ItemListWidget.js (/girder/views/widgets)	collapseImage.styl
FrontPageView.js	✓	Affiche un bloc d'information expliquant à l'utilisateur les démarches pour utiliser les applications VIP sur Girder (Créer un compte vip, rentrer la clé api de	frontPage.pug	FrontPageView.js (/girder/views/body)	-

		<p>vip...).</p> <p>Si l'utilisateur n'est pas connecté, on précise en plus qu'il faut créer un compte sur Girder.</p>			
HeaderUserView.js	✓	Ajoute le bouton 'My pipelines' dans le menu déroulant de l'utilisateur.	headerUser.pug	HeaderUserView.js (/girder/views/layout)	-
ItemView.js	✓	Sur la page d'un item, ajoute l'icône 'Rocket' qui redirige vers les applications, derrière chaque fichier d'un item.	buttonListPipeline.pug	ItemView.js (/girder/views/body)	-
ListPipelines.js	✗	Liste les applications de l'utilisateur lorsqu'il traite un seul fichier.	listePipelines.pug	View.js (/girder/views) ConfirmPipelineDialog.js (vue du plugin)	-
ListPipelinesMultiFiles.js	✗	Liste les applications de l'utilisateur lorsqu'il traite plusieurs fichiers.	listPipelinesMultiFiles.pug	View.js (/girder/views) ConfirmPipelineDialogMultiFiles.js (vue du plugin)	-
MyPipelines.js	✗	Liste l'historique des exécutions de l'utilisateur.	myPipelines.pug	View.js (/girder/views)	myPipelines.styl
UserAccountView.js	✓	Dans les paramètres de l'utilisateur, ajoute un nouvel onglet pour ajouter l'apikey de VIP.	userAccountTab.pug userAccountVIP.pug	UserAccountView.js (/girder/views/body)	-

Cookbook

Faire une requête sur l'API de Girder

```
1 import { restRequest } from 'girder/rest';
2
3 restRequest({
4   method: 'GET',
5   url: 'item/',
6   data: {folderId: '5ac63fea37121ca25cfa1d00a'}
7 }).then(function (resp) {
8   // Success function, do something
9 }).catch(function (e) {
10  // Failed function
11 });
12
```

Créer une nouvelle page et une nouvelle route

Pour créer une nouvelle page, nous avons besoin de créer :

- Une route
- Un template
- Une vue

Exemple : Créons la page « hello » accessible depuis <http://localhost:8080/#hello>.

Route

```
1 // creatis/web_client/route.js
2
3 import router from 'girder/router';
4 import router from 'girder/events';
5
6 // View
7 import Hello from './views/Hello';
8
9 router.route('hello', 'hello', function () {
10   events.trigger('g:navigateTo', Hello);
11 });
```

Path (/#hello)

Evènement JS

Vue (Hello.js)

Template

Peut s'écrire aussi :

`p(id='message')= message`

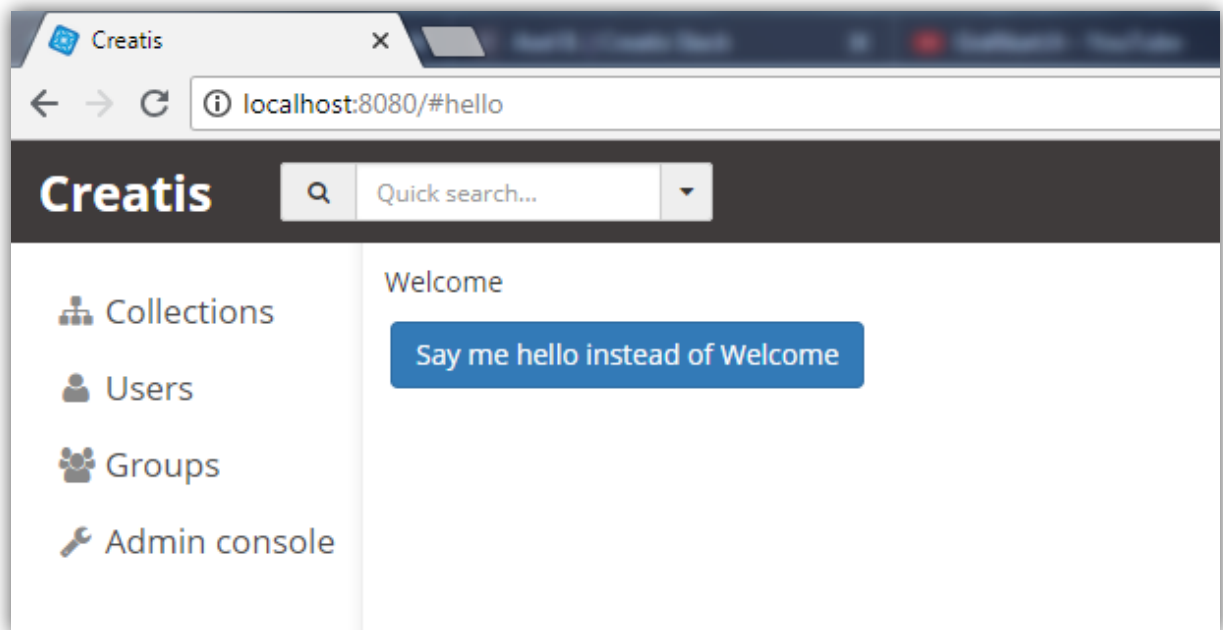
```
1 /* creatis/templates/hello.pug */
2
3 p#message= message
4 button.btn.btn-large.btn-primary#buttonHello Say me hello instead of Welcome
5
```

Vue

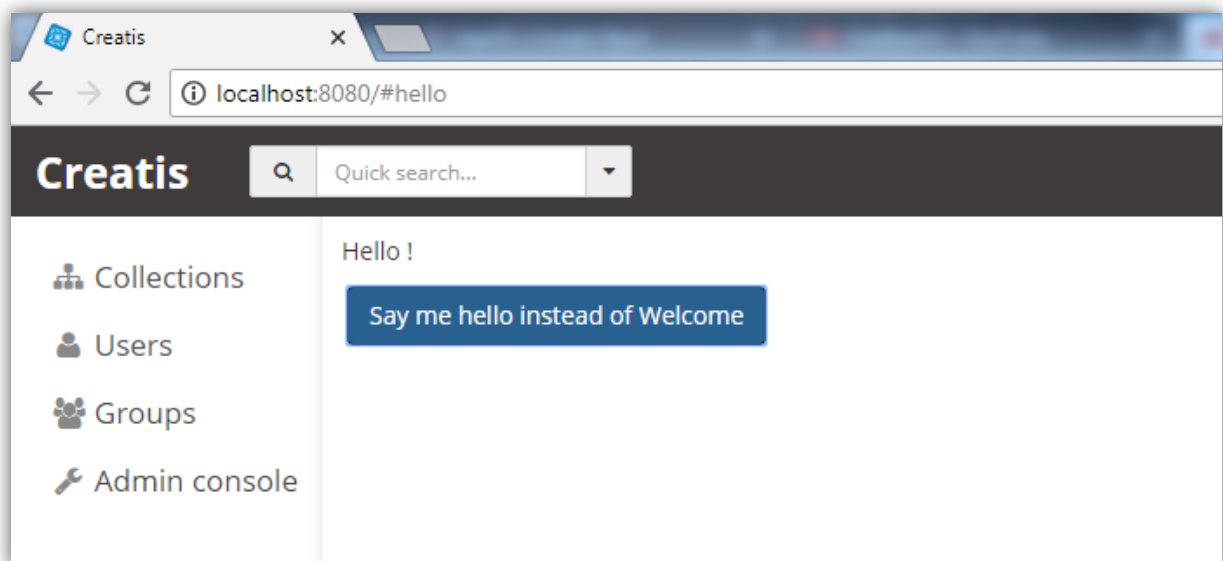
```
1 // creatis/web_client/views/Hello.js
2
3 // View
4 import View from 'girder/views/View';
5
6 // Template
7 import HelloTemplate from '../templates/hello.pug';
8
9 var Hello = View.extend({
10   // Constructor, set variables...
11   initialize: function (settings) {
12     this.message = "Welcome";
13     this.render();
14   },
15
16   // Lier un évènement à une fonction
17   events: {
18     'click #buttonHello' : 'sayHello'
19   },
20
21   render: function () {
22     this.$el.html(HelloTemplate({
23       // Variables envoyées dans le template, ex:
24       message: this.message
25     }));
26   },
27
28   // Cette fonction est appelée quand l'utilisateur clique sur le bouton qui a comme id
29   // 'buttonHello'
30   sayHello: function (e) {
31     // Récupérer l'élément de la page avec un id 'message'
32     var a = e.view.$.find('#message');
33
34     // Le transformer en objet jquery et remplacer le texte
35     $(a).text('Hello !');
36   }
37 });
38 export default Hello;
```

Note du développeur : *N'oubliez pas l'export, de longues heures perdues à cause de ça.. (l.38)*

Voici le resultat de notre nouvelle page



Quand on clique sur le bouton « Say me hello instead of Welcome »



Je vous l'admet, l'intérêt de cette page n'est pas super.. Mais c'est justement à vous de créer vos propres actions dans la vue :D

SERVEUR

Description /serveur

Le dossier « models » contient 2 fichiers :

- `__init__.py`
 - Fichier vide mais indispensable pour le fonctionnement du plugin
- `Pipeline.py`
 - Décrit le modèle Pipeline qui sera dans la bdd

Si vous voulez créer une nouvelle table, il faudra donc créer un nouveau modèle décrivant votre objet.

Fichier	Description
<code>__init__.py</code>	Fichier principal qui inclut les fichiers des nouvelles routes de l'API. Met à jour la description de l'API (/api/v1)
<code>pipeline_rest.py</code>	Les routes communiquant avec la table (collection) 'Pipeline_execution' n'existent pas, on crée donc à partir du modèle toutes les routes dont on a besoin. C'est par exemple ce fichier qui sera inclus dans le fichier <code>__init__.py</code> .
<code>user_rest.py</code>	Les routes communiquant avec la table (collection) 'User' existent déjà, on a juste besoin d'étendre ces routes. Ce fichier est aussi inclus dans le fichier <code>__init__.py</code> .

Le fichier 'main' est le fichier appelé « `__init__.py` », c'est le point d'entrée de l'extension de notre API. Tous les autres fichiers nommés par exemple `xxxxxx_rest.py` décrivent les différentes routes de l'API et sont inclus dans le fichier principal « `__init__.py` ».

Une route est reliée à une fonction, par exemple :

```
1  # creatis/server/myExtend_rest.py
2
3  ...
4
5  self.route('GET', (':param1', ':param2', ...), self.myFunction)
6
7  ...
8
9  @access.public
10 @autoDescribeRoute(
11     Description("Description de ma nouvelle route")
12     .param('param1', 'Description du paramètre 1')
13     .param('param2', 'Description du paramètre 2')
14     .errorResponse('Message d\'erreur personnalisé')
15 )
16 def myFunction(self, params):
17     # do something
18
```

Méthode HTTP

Nouvelle route

ANNEXES

Laboratories and financiers

CREATIS : <https://www.creatis.insa-lyon.fr/site7/fr>

Team 2 – Images et modèles : <https://www.creatis.insa-lyon.fr/site7/fr/immod>

CNRS : <http://www.cnrs.fr/>

INSERM : <https://www.inserm.fr/>

INSA Lyon : <https://www.insa-lyon.fr/>

About plugin

Github VIP : <https://github.com/virtual-imaging-platform>

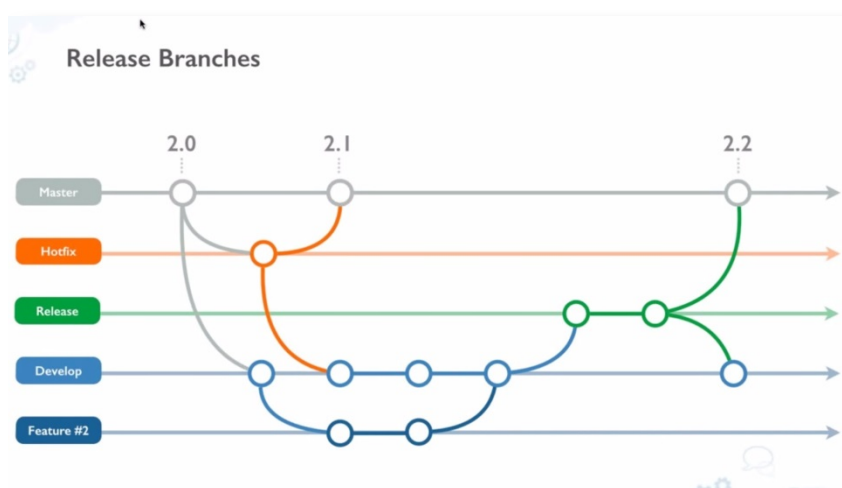
Github girder : <https://github.com/girder/girder>

Github CARMIN : <https://github.com/CARMIN-org/CARMIN-API>

Swagger CARMIN : https://app.swaggerhub.com/apis/CARMIN/carmin-common_api_for_research_medical_imaging_network/0.3

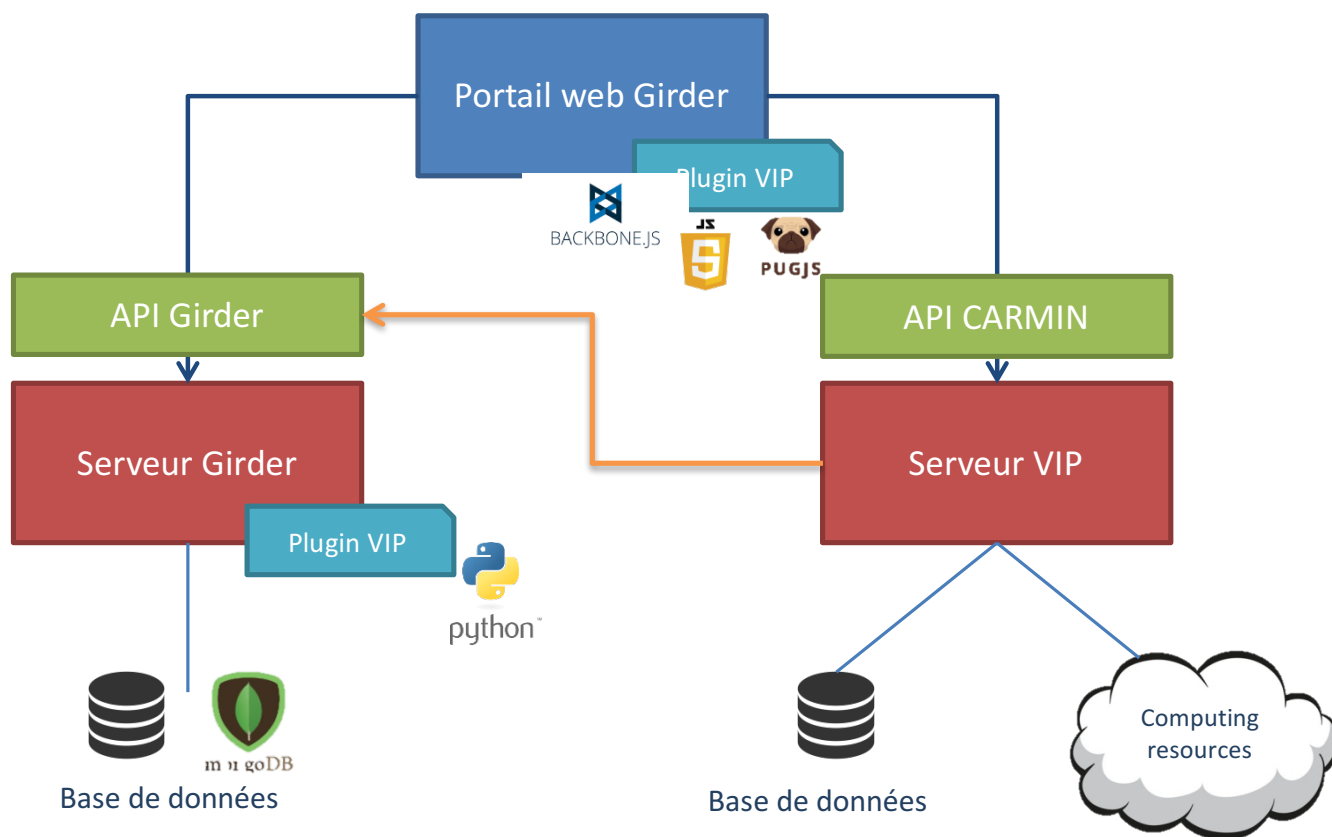
Github plugin : <https://github.com/Saloukai/VIP-girder-plugin>

Work on GitHub :

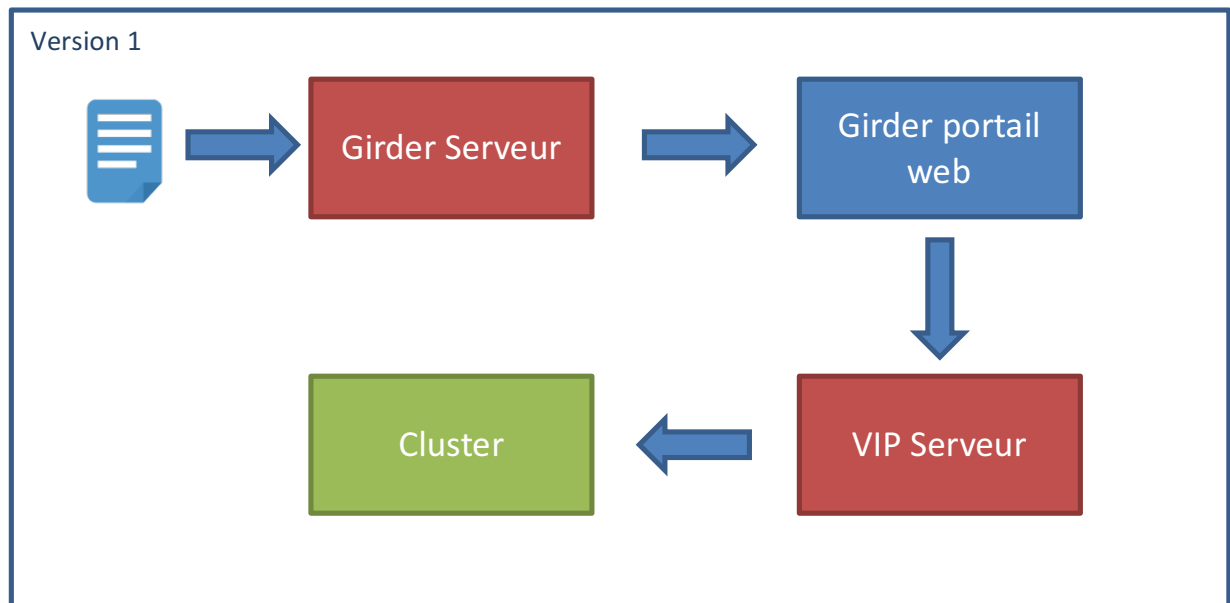


Work on GitHub – The plugin is on the branch « develop »

Schema – Architecture logicielle



Schema – Tranfert d'un fichier v1



Schema – Tranfert d'un fichier v2

