CAhier Numérique de LAboratoire de Recherche (CANULAR)

Christelle Aluome, Christophe Chipeaux, Nicolas Devert, Catherine Lambrot, Tovo Rabemanantsoa, Stéphane Thunot


CANULAR
Cahier numérique pour laboratoire de recherche
INRAE

SOMMAIRE

Contexte

La parution en 2021 du rapport du groupe de travail «Cahier de laboratoire électronique» (ELN)[1] sur le site internet «Ouvrir la science» a permis de prendre connaissance d'un ensemble de recommandations sur les critères de choix d'un outil et d'intégrer une liste comparative d'outils existants. La participation de deux membres de ce projet à ce groupe de travail a été déterminant sur la volonté de faire suite en proposant une solution mobile appliquée d'ELN basée sur le couple logiciel et matériel.
Un des freins à l'adoption des ELN est la crainte d'une perte de praticité et de souplesse par rapport à son homologue papier. Le personnel de la recherche, habitué à l'utilisation du cahier papier, souhaite pouvoir emmener son cahier là où l'expérimentation se déroule (serre, terrain, laboratoire,...) et «gribouiller» ses notes à main levée.
Pour lever ce frein, nous proposons de prendre 2 éléments, existants et basés sur des technologies ouvertes et matures et de les faire fonctionner en symbiose. Ces 2 éléments sont eLabFTW [2]et une liseuse.
Nous avons choisi eLabFTW comme socle d'ELN car c'est un outil libre (Open Source), développé par une communauté issue de la recherche dont les fonctionnalités sont agnostiques par rapport aux domaines de recherche. Son usage permet la maîtrise complète des données produites et gérées. Autre raison est le fait qu'il soit basé sur une interface web standard, ce qui lui permet d'être multi-plateforme. Enfin, il offre une interface de programmation applicative (API) bien fournie qui confère une interopérabilité accrue.
Nous avons choisi la liseuse parce que son coût est raisonnable (autour de 300€), sa technologie à base d'encre électronique lui donne une autonomie considérable (plus d'une semaine), sa capacité de stockage est généralement confortable et sa connectivité exemplaire (wifi, bluetooth). Enfin, son système d'exploitation Android® permet aisément l'installation, voir le développement d'applications compatibles afin d'étendre ses fonctionnalités en gardant l'ergonomie de ces dernières. Avec une liseuse dotée d'un stylet pour l'écriture, l'utilisateur retrouve finalement la souplesse de l'écriture manuelle associée à la puissance d'un objet connecté. Une fois l'application Android® développée, on pourra la réutiliser sur n'importe quel autre support (Mobile, tablette) compatible.
L'idée est d'arriver rapidement à une adoption généralisée de cet ELN et de se substituer au cahier de laboratoire papier.
Le projet CANULAR a reçu un soutien financier de la pépinière numérique d'INRAE.

Objectif

L'objectif du projet CANULAR est d'établir une preuve de concept (PoC) pour l'interfaçage d'une liseuse électronique avec un logiciel de cahier électronique existant. Cet interfaçage est possible par le développement d'une application Android® sur la liseuse . Le PoC ne pourra être validé que si l'application permet de récupérer des données du cahier électronique sur la liseuse et de pouvoir effectuer des actions (recherche, lecture, écriture) dans l'autre sens.

Participants et rôles

Nom	Unité	CATI	Rôle
Aluome Christelle	ISPA	PROSODIe	Conception / Développement
Chipeaux Christophe	ISPA	DIISCICO	Conception
Devert Nicolas	ISPA	Sans CATI	Testeur
Lambrot Cathy	ISPA	Sans CATI	Testeur
Rabemanantsoa Tovo	ISPA	PROSODIe	Conception
Thunot Stéphane	ISPA	GEDEOP	**Chef de projet** / Designer

Matériels et méthodes

Choix de la solution de cahier électronique de laboratoire

Parmi la grande variété de cahiers électroniques existants, nous avons choisi pour ce projet, elabFTW. Ce choix a été motivé par le rapport du Groupe de Travail ELN. Ce rapport a dressé une liste de 23 critères de choix (fonctionnalités, utilisation, interopérabilité, sécurité, sauvegarde,...) et a évalué 16 solutions d'ELN en fonction de ces critères. ElabFTW fait partie des solutions testées et remporte une note de 16/23. De plus, un des grands avantages de ElabFTW, est la présence d'une interface de programmation (API [3]), permettant d'interagir avec le contenu de ELN à distance. Le projet est porté par Nicolas Carpi (développeur principal Elabftw) avec qui nous avons pu échanger.

Choix de la liseuse

En ce qui concerne le choix de la liseuse, notre attention s'est portée très vite sur la liseuse Notéa [4] de l'entreprise française Bookeen.

Plusieurs arguments ont joué en sa faveur :

- c'est une liseuse haut de gamme, avec un écran de bonne qualité et la présence d'un stylet,
- Elle possède une autonomie d'une semaine,
- Elle fonctionne sur un système d'exploitation rooté ouvert : Android
- Bookeen est une entreprise française.

Malgré tout, il est à noter plusieurs points négatifs :

- Réactivité bien moindre qu'une tablette,
- Système Android vieillissant (Android 8.2 alors que la dernière version disponible est Android 11),
- Absence de module logiciel de reconnaissance de caractères (OCR) intégré.
- Absence de modules matériels : caméra et GPS

L'aide de la pépinière numérique nous a permis d'acheter deux liseuses pour notre PoC.

Personne de notre groupe n'avait de compétence en développement Android®. Le seul environnement que nous connaissions pour développer des applications était Android Studio®. C'est donc par cet outil que nous avons commencé. Assez vite, les limites de ce choix ce sont imposées. Nos compétences en développement sous le langage Java étaient limitées et assez lointaines et cela a très vite été un soucis pour avancer dans le développement, sans compter l'architecture d'ensemble d'une application Android® qui nous était méconnue. Notre groupe a alors lancé un appel à assistance auprès de l'ensemble des CATI 3G. Très vite, le framework IONIC [5] a été mentionné. Il est en effet utilisé par plusieurs groupes de développement au sein des CATI 3G. Un avantage certain de ce framework est que les langages de programmation qu'il faut maîtriser sont le HTML/CSS et JavaScript pour lequel notre groupe possédait des compétences. De plus, le framework possède un générateur de vue simple qui permet de rapidement déployer de nouvelles pages. Les compétences du groupe et la facilité de prise en main de l'outil par comparaison avec Android studio® ont motivé le choix de travailler sur ce framework, d'autant plus que nous savions pouvoir faire appel à la communauté Android® des CATI 3G en cas de blocage Associé à IONIC, nous avons fait le choix d'utiliser le framework ANGULAR [6] pour le côté applicatif. Enfin, certaines fonctionnalités comme la gestion des accès au matériel (lecture/écriture sur la liseuse) sont gérées par l'installation et l'utilisation de plugins CORDOVA [7].

En résumé, IONIC est utilisé pour le frontend, ANGULAR pour le backend (manipulation des données) et CORDOVA pour le backend (interaction matérielle)

Développement

Installation de ElabFTW sur un serveur

La première étape du travail a consisté dans le déploiement de la solution ElabFTW sur une machine sous Debian 10. Même si la documentation d'ElabFTW recommande l'installation avec Docker, nous avons choisi de l'installer sans, en suivant l'installation dite «old-school» : https://web.archive.org/web/20201112010121/https://doc.elabftw.net/install-oldschool.html (page supprimée depuis - seule l'installation sous docker est maintenant proposée).

Il y a deux raisons à ce choix :

- l'absence de maîtrise de la technologie Docker

- un besoin de comprendre comment les différents éléments de l'applicatif fonctionnent et s'articulent ensemble (ce qui n'est pas aisé pour nous avec Docker).

Notre procédure d'installation ainsi que la configuration de notre serveur web (apache2) est disponible en annexe.

Notre instance de cahier électronique de laboratoire est accessible à : https://canular.inrae.fr/

- Architecture générale de l'application

	**Home** : Page d'accueil / Présentation du projet CANULAR **Database** : Moteur de recherche, permettant la consultation des fiches de la base de connaissance. Fonctionne en fournissant l'identifiant de la fiche. **Expériences** : Moteur de recherche, permettant la consultation du cahier électronique. Fonctionne en fournissant l'identifiant de la fiche. **Fichiers** : Affichage de la liste des notes enregistrées, possibilité de les visualiser et de les supprimer. **Synchronisation** : Permet d'envoyer sur le serveur ElabFTW les notes en les liant à une fiche expérience déjà existante ou à une nouvelle fiche à créer également sur cette page. **Configuration** : section vide à dédier à la gestion de son compte. **A propos** : présentation de l'équipe de développement. **Fermer** : fermeture de l'application.

➢ Fonctionnalités pour valider le PoC

Pour valider le PoC, nous sommes partis sur une application intégrant ces deux fonctionnalités :

- Lire le contenu d'ElabFTW depuis l'application CANULAR
- Ajouter des notes prises depuis la liseuse dans ElabFTW

- Lire le contenu d'ElabFTW depuis l'application CANULAR

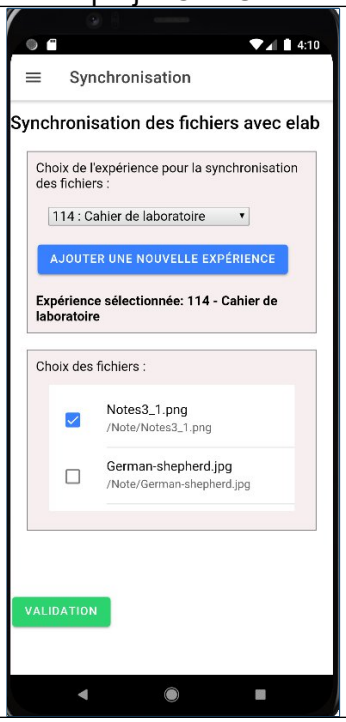
Visuel de la section «base de données» de l'application ElabFTW.

Recherche et affichage depuis l'application Android canular.

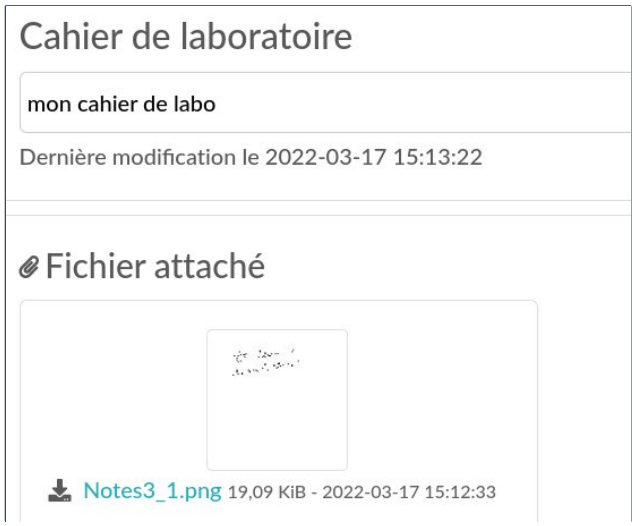
- Ajouter des notes prises depuis la liseuse dans ElabFTW

Les notes sont prises depuis la liseuse Notéa et enregistrées soit au format PNG (une note) ou au format PDF (une compilation de note). Les fichiers «notes» sont visibles dans l'application CANULAR depuis la rubrique «fichier» ou la rubrique «synchronisation».

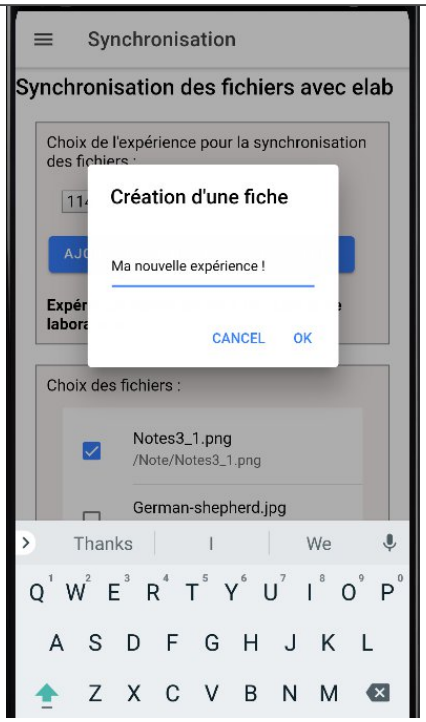
Pour pouvoir synchroniser les notes avec ElabFTW, il convient depuis la rubrique synchronisation de sélectionner l'expérience à laquelle on veut rajouter les notes et de cocher le ou les fichiers à synchroniser avant de valider. Le/les fichier(s) sont alors attaché(s) à l'expérience correspondante. A noter que nous avons également implémenté la possibilité de créer à la volée une nouvelle expérience vide à laquelle il est possible par la suite de lier les notes.



Page de synchronisation : choix de l'expérience et des fichiers à lier.



Fichier de note attaché à la fiche après synchronisation.



Page de synchronisation : création d'une nouvelle expérience pour y lier les notes.

La nouvelle expérience a été créée sur ElabFTW !

- Tests effectués

  Deux séries de tests ont été effectuées

  – Des tests «locaux» sur une machine virtuelle Android® générée par Android Studio® sous système Android® 8.1, avec l'appareil Pixel 5 API 27. En effet, la librairie Capacitor [8] permet de faire facilement le lien entre les frameworks IONIC/ANGULAR et Android Studio® en faisant communiquer les codes développés sous Javascript (IONIC/ANGULAR) et le Java sous Android Studio®. Ces premiers tests ont permis de s'assurer que l'application fonctionne comme attendu.
  – Des tests en condition réelle ont été réalisés sur la liseuse Notéa après compilation et installation de l'application afin de valider le PoC.

Difficultés rencontrées

L'authentification

L'authentification sur l'ELN à un compte utilisateur se fait via une clé API. Une clé API est une suite de 80 caractères alphanumériques. Il n'est pas envisageable de demander à un utilisateur qui veut se connecter de remplir sa clé dans son application Android®. Il convient donc d'imaginer une solution pour qu'une authentification soit par login, mail ou LDAP puisse être possible, et que l'authentification permette de récupérer la clé.

La recherche d'information

La recherche actuellement ne peut se faire sur l'API que via le numéro d'expérience ou le numéro de la fiche de base de données. Une recherche via le titre ou via le nom de l'auteur n'est pas possible de manière directe. Il y a bien une manière indirecte de :

- récupérer l'intégralité des entrées de la base de données.
- faire une recherche de similarité sur le titre et/ou l'auteur
- récupérer l'identifiant de l'entrée qui matche avec la recherche
- refaire une requête sur l'API avec cet identifiant (en effet, une recherche sur l'intégralité des données permet de récupérer toutes les informations sauf le «body» de la fiche, ce qui nécessite une seconde requête pour récupérer ces informations)

La méthode décrite qui a été testée, n'est pas viable, car elle va consommer beaucoup de bandes passantes, la meilleure solution serait que l'API soit modifiée pour intégrer une recherche textuelle, en attente d'une réponse du développeur.

Conclusion

Ce premier développement a mis en évidence la faisabilité du projet , il reste cependant encore de nombreux modules logiciel à développer après évolution de l'API. Un retour d'expérience par différents types d'utilisateurs permettra également de finaliser l'outil et de juger de son potentiel d'adoption.

Modules logiciel à développer :

- gestion de session utilisateur
- authentification login /mdp
- recherche textuelle (titre et auteur)
- module OCR à intégrer (côté serveur ?)

Bien que nous soyons persuadés de l'utilité d'un tel outil, nous n'avons pas les ressources nécessaires pour continuer le développement jusqu'à une application complète dans un délai raisonnable. Une solution envisageable serait de faire appel à un prestataire pour développer l'application.

Liens / références

[1] - https://www.ouvrirlascience.fr/wp-content/uploads/2022/01/Rapport_GT_ELN_v3.2-2022-01-04FINAL.pdf

[2] - https://www.elabftw.net/

[3] - https://doc.elabftw.net/api/

[4] - https://bookeen.com/pages/notea-bloc-notes-numerique

[5] - https://ionicframework.com/

[6] - https://ionicframework.com/docs/angular/overview

[7] - https://cordova.apache.org/docs/en/10.x/guide/hybrid/plugins/

[8] - https://capacitorjs.com/docs/android

Annexe : Installation de eLabFTW sans Docker

Préambule

- documentation : https://doc.elabftw.net/

- installation suivie : https://web.archive.org/web/20201112010121/https://doc.elabftw.net/install-oldschool.html

- os : debian 10

Installation des paquets sur la VM :

! PHP - attention par défaut sur debian10, php est en version 7, et elab nécessite la version 8.

```
sudo apt install -y lsb-release apt-transport-https ca-certificates wget
sudo wget -O /etc/apt/trusted.gpg.d/php.gpg https://packages.sury.org/php/apt.gpg
sudo apt update
sudo apt install php8.0 -y
sudo apt install php8.0-fpm -y
sudo apt install php-curl
sudo apt install php-ldap
sudo apt install php-mbstring
sudo apt install php-gd
sudo apt install php-zip
sudo apt install php-mysql
```

Installation de la base de données.

```
sudo apt install mariadb-server
```

Importation de ElabFTW

```
sudo apt install git
cd /var/www/html/
sudo git clone -b 4.1.0 --depth 1 https://github.com/elabftw/elabftw.git
```

Installation de composer pour gérer les dépendances sous PHP

```
sudo apt install wget php-cli php-zip unzip
wget -O composer-setup.php https://getcomposer.org/installer
sudo php composer-setup.php --install-dir=/usr/local/bin --filename=composer
```

```
su -s /bin/bash www-data

composer install --no-dev
```

Installation de Yarn

!! Ne pas utiliser la commande "apt install yarn" qui installe le mauvais paquet (paquet cmdtest)

```
sudo apt install curl

curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -

echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee /etc/apt/sources.list.d/yarn.list

sudo apt update

sudo apt install yarn

yarn install --prod
```

Pour résoudre ce message d'erreur :

error css-loader@6.5.0: The engine "node" is incompatible with this module. Expected version ">= 12.13.0". Got "10.24.0"

error Found incompatible module.

il faut mettre à jour la version de node.js

```
sudo apt-get install curl software-properties-common

curl -sL https://deb.nodesource.com/setup_16.x | sudo bash -

sudo apt-get install nodejs

node --version #pour vérifier qu'on a une version suffisante.
```

Compiler avec yarn le js + css.

- supprimer le fichier "yarn.lock", si présent d'abord, sinon ça ne recompile pas.

```
yarn install --prod

yarn buildall
```

Quelques soucis de compilation

/bin/sh: 1: brotli: not found

/bin/sh: 1: zopfli: not found

```
sudo apt-get install brotli

sudo apt-get install zopfli
```

Création des répertoires

```
mkdir cache uploads

chown www-data:www-data cache uploads

chmod 700 cache uploads
```

Base de données

Installation sécurisée de la bd

```
sudo mysqlsecureinstallation
```

notamment mot de passe sur compte root

Création de la base de données

```
mysql -uroot -p

create database elabftw character set utf8mb4 collate utf8mb4generalci;

grant usage on . to elabftw@localhost identified by 'testelabcanular';
```

Paramétrer le serveur web

Choix de Apache2

-création du 001-canular.conf dans /etc/apache2/sites-avalaible

```
<VirtualHost *:80>

 ServerName canular.inrae.fr
 ServerAdmin webmaster@localhost
 DocumentRoot /var/www/html/elabftw/web

 Header set Access-Control-Allow-Methods "GET, POST, OPTIONS"


 <Directory />
 Require all denied
 Options -Indexes -Includes -ExecCGI -FollowSymlinks
 </Directory>

 <Directory /var/www/html/elabftw/web>
 Require all granted
 Options -Indexes -Includes -ExecCGI +FollowSymlinks
 Header add Access-Control-Allow-Methods "GET, POST, OPTIONS"
```

```
</Directory>
# Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
# error, crit, alert, emerg.
# It is also possible to configure the loglevel for particular
# modules, e.g.
#LogLevel info ssl:warn

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

# For most configuration files from conf-available/, which are
# enabled or disabled at a global level, it is possible to
# include a line for only one particular virtual host. For example the
# following line enables the CGI configuration for this host only
# after it has been globally disabled with "a2disconf".
#Include conf-available/serve-cgi-bin.conf

RewriteEngine on
RewriteCond %{HTTP:Authorization} ^(.*)
RewriteRule .* - [e=HTTP_AUTHORIZATION:%1]
RewriteRule ^ - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
RewriteCond %{SERVER_NAME} =canular.inrae.fr
RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]


</VirtualHost>
```

Elab fonctionne en https seulement, besoin d'un certificat

```
apt install certbot

apt install python3-certbot-apache

certbot --apache -d canular.inrae.fr --post-hook "/usr/sbin/service apache2 restart"
```

Si tout se passe bien

- Congratulations! Your certificate and chain have been saved at:

/etc/letsencrypt/live/canular.inrae.fr/fullchain.pem

Your key file has been saved at:

/etc/letsencrypt/live/canular.inrae.fr/privkey.pem

Your cert will expire on 2022-01-31. To obtain a new or tweaked

version of this certificate in the future, simply run certbot again

with the "certonly" option. To non-interactively renew allof

your certificates, run "certbot renew"

- Your account credentials have been saved in your Certbot
- configuration directory at /etc/letsencrypt. You should make a
- secure backup of this folder now. This configuration directory will

- also contain certificates and private keys obtained by Certbot so
- making regular backups of this folder is ideal.
- If you like Certbot, please consider supporting our work by:
- Donating to ISRG / Let's Encrypt: https://letsencrypt.org/donate
- Donating to EFF:https://eff.org/donate-le

Génère automatiquement le fichier 001.canular-le-ssl.conf

il faut un peu modifier ce fichier, notamment pour éviter les problèmes de Cross-origin resource sharing (CORS) avec les développement sous IONIC, en ajoutant des règles sur les HEADERS.

L'ensemble des règles utilisées sont détaillées ci-dessous.

attention "Access-Control-Allow-Origin" est à "*" pour les tests, mais ça sera à modifier en production.

```
<IfModule mod_ssl.c>
<VirtualHost *:443>

 ServerName canular.inrae.fr
 ServerAdmin webmaster@localhost
 DocumentRoot /var/www/html/elabftw/web
 Header set Access-Control-Allow-Methods "GET, POST, OPTIONS"
 Header set Access-Control-Allow-Origin *



 <Directory />
 Require all denied
 Options -Indexes -Includes -ExecCGI -FollowSymlinks
 Header always set Access-Control-Max-Age "3600"
 Header always set Access-Control-Allow-Headers "access-control-allow-origin, x-requested-with,
Content-Type, Origin, authorization, ac$
 RewriteEngine On
 RewriteCond %{REQUEST_METHOD} OPTIONS
 RewriteRule ^(.*)$ $1 [R=200,L]


 </Directory>

 <Directory /var/www/elabftw/web>
 Require all granted
 Options -Indexes -Includes -ExecCGI +FollowSymlinks
 Header set Access-Control-Allow-Origin "*"
 Header set Access-Control-Max-Age "3600"
 Header set Access-Control-Allow-Headers "access-control-allow-origin, x-requested-with, Content-Type,
Origin, Authorization, accept, c$
 Header always set Access-Control-Expose-Headers "Authorization, *"
 RewriteEngine On
 RewriteCond %{REQUEST_METHOD} OPTIONS
 RewriteRule ^(.*)$ $1 [R=200,L]

 </Directory>
# Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
 # error, crit, alert, emerg.
```

```
# It is also possible to configure the loglevel for particular
# modules, e.g.
#LogLevel info ssl:warn

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

# For most configuration files from conf-available/, which are
# enabled or disabled at a global level, it is possible to
# include a line for only one particular virtual host. For example the
# following line enables the CGI configuration for this host only
# after it has been globally disabled with "a2disconf".
#Include conf-available/serve-cgi-bin.conf

SSLCertificateFile /etc/letsencrypt/live/canular.inrae.fr/fullchain.pem
SSLCertificateKeyFile /etc/letsencrypt/live/canular.inrae.fr/privkey.pem
Include /etc/letsencrypt/options-ssl-apache.conf

SSLEngine on
SSLProxyEngine on
RewriteEngine On
RewriteCond %{HTTP:Authorization} ^(.*)
RewriteRule .* - [e=HTTP_AUTHORIZATION:%1]
RewriteRule ^/api/v1/(.*)$ /app/controllers/ApiController.php?req=%{REQUEST_URI}&args=$args [P,L]
Header add Access-Control-Allow-Methods "GET, POST, OPTIONS"


</VirtualHost>
</IfModule>
```

Redémarrer le serveur

```
a2enmod proxy proxy_http

sudo service apache2 restart
```

test si ça fonctionne ici : http://canular.inrae.fr/elabftw

Erreur possible si plusieurs versions de php qui cohabite sur la machine.

désactiver celles antérieures à la 8 si besoin.

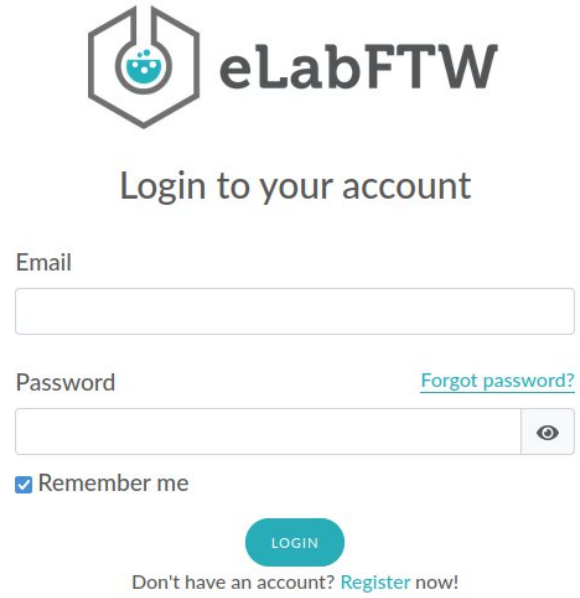
```
a2dismod php7.3

a2enmod php8.0

service apache2 restart
```

on teste si ça fonctionne ici : http://canular.inrae.fr/elabftw

si on arrive sur la page login.php et que ça ressemble à ça :



L'installation est réussie.

Configuration d'eLabFTW

Création du compte admin :

https://canular.inrae.fr/register.php

```
chmod 777 /var/www/html/elabftw/cache/purifier
chown -R www-data:www-data elabftw/
```

Configuration du Serveur mail

```
sudo apt install exim4
```

Configuration

```
/etc/exim4/update-exim4.conf.conf
```

Fichier de configuration

```
dceximconfigconfigtype='smarthost'
dcotherhostnames=''
dclocalinterfaces='127.0.0.1 ;147.100.202.22' # seconde adresse = ip du serveur
dc_readhost='bordeaux.inrae.fr'
dcrelaydomains=''
dc_minimaldns='false'
```

```
dcrelaynets=''
dc_smarthost='smtp.inrae.fr::587'
CFILEMODE='644'
dcusesplit_config='false'
dchidemailname='true'
dcmailnamein_oh='true'
dclocaldelivery='mailspool'
```

Compte mail

/etc/exim4/passwd.client

```
*:compte@inrae.fr:password
```

\## adresse

/etc/email-addresses

```
root:compte@inrae.fr
debian:compte@inrae.fr
www-data:compte@inrae.fr
```

Redémarrer le service

```
sudo update-exim4.conf
sudo systemctl restart exim4.service
```

Test si ça fonctionne

```
/usr/bin/mailx -n -s "test subject" -a "sdeeph06 - mail" blabla.blabla@inrae.fr
Configuration mail avec exim
exim4-base installé
```

Test envoi :

```
echo "Message de test" | mail blabla.blabla@inrae.fr -s "Test date"
```