LABINVENT - INSTALLATION

URL officielle de ce doc : https://tinyurl.com/labinvent-install
Auteurs: E. Pallier, E. Bourrec
Version: 30/3/2021
⇒ *Pour faire une copie Word, Libre/Open Office, ou PDF de ce doc : menu "Fichier/Télécharger…"*

Cette documentation décrit la phase d'installation du logiciel.
Une fois l'installation terminée, consultez la ***documentation technique*** ***générale*** *pour savoir comment configurer le logiciel et l'adapter selon vos besoins.*

Un gros effort est fait continuellement pour que cette doc soit le plus à jour et pertinente possible,

merci de bien vouloir la lire ATTENTIVEMENT.

Les auteurs de cette doc eux-mêmes la suivent à la lettre, pourquoi donc feriez-vous autrement ?

N'hésitez pas aussi à la mettre à jour vous-mêmes quand c'est nécessaire, ou bien à soumettre vos suggestions à :

epallier AT irap POINT omp POINT eu

(ANCIENNE DOC qui n'est plus à jour: lien vers la doc d'origine)

Pour la documentation (non technique) décrivant l'utilisation du logiciel, cliquer sur le lien ci-dessous :

⇒ **LIEN VERS LA DOC UTILISATEURS**

HISTORIQUE DES MISES A JOUR DE CETTE DOCUMENTATION

- ...
- 17/2/21 Ajout nouveau chapître “installation avec docker”
- 20/1/21 **Simplification** : Migration de certains chapitres (hors contexte) dans la doc technique générale
- 25/11/20 Mise à jour du chapitre “Etiquettes” pour ajouter la nouvelle étiqueteuse Dymo LabelManager 420P
- 19/10/20 Ajout nouveau chapître “Astuces (howto)”
- 29/9/20 Mise à jour du chapitre “Installations existantes” pour ajouter IP2I
- 28/9/20 Ajout d’un nouveau chapitre expliquant la gestion des notifications
- 8/9/20 Ajout d’un exemple dans le howto “Ajouter une table dans la BD” (ajout de la nouvelle table “**projets**”)
- 4/9/20 Mise à jour du chapitre “Etiquettes” suite à la mise en place de la nouvelle étiqueteuse Dymo MobileLabeler
- 24/7/20 Mise à jour du chapitre “Installations existantes” pour IAS
- 23/7/20 Ajout de CETTE page-ci
- 23/7/20 Modification/Ajout des chapitres suivants :
    - Configuration des autorisations
    - Configuration des logos
    - Configuration des étiquettes
    - Configuration niveau DEV only

TABLE DES MATIÈRES

1. LICENCE

Auteurs : Etienne Pallier (epallier@irap.omp.eu), Elodie Bourrec (ebourrec@irap.omp.eu)

Le logiciel **LabInvent** (InventIrap pour l'IRAP) est sous licence libre copyleft **AGPL** (**GNU Affero General Public License**) v3.0 dont le détail est donné sur la page web https://spdx.org/licenses/AGPL-3.0.html#licenseText, mais aussi dans le fichier texte "LICENSE AGPL" à la racine du projet.

(voir aussi https://choosealicense.com/licenses/agpl-3.0, https://www.diatem.net/les-licences-open-source, et https://www.gnu.org/licenses/why-affero-gpl.fr.html, ainsi que https://fr.wikipedia.org/wiki/GNU_Affero_General_Public_License)

Ce logiciel est développé depuis 2012, à l'origine pour les besoins du laboratoire IRAP de Toulouse, sous le nom dédié "Inventirap". Depuis, il a été diffusé dans d'autres laboratoires, avec l'appellation plus générique de "LabInvent". Il est construit sur un framework Php orienté objets nommé "CakePhp", dans sa version 3.x (http://cakephp.org) qui n'est pas inclut mais récupéré automatiquement au moment de l'installation. Le framework CakePHP est sous licence MIT (licence sans copyleft). Il fonctionne avec Php 7 (mais reste encore compatible avec Php 5.6+).

Bien qu'il soit développé avec une attention particulière portée sur la qualité, ce logiciel est mis à disposition "en l'état" ("as is"), sans garantie aucune.

Toute modification n'altérant pas la finalité principale du logiciel qui est d'inventorier les matériels, est autorisée (et même encouragée) à deux conditions :

- Elle doit être partagée et ré-injectée dans le logiciel
- Elle doit être activable ou désactivable depuis la page web de configuration générale, et doit être désactivée par défaut

C'est afin que toute la communauté des utilisateurs puisse en profiter, et aussi pour que le logiciel LabInvent reste une entité unique et bien définie.

2. Améliorations possibles de la phase d'installation

(EP updated 20/1/21)

On met ici quelques pistes de réflexion sur la manière dont l'étape d'installation du logiciel pourrait être améliorée (ébauche en cours d'élaboration) - Vous pouvez bien sûr ignorer cette section si votre seule préoccupation est d'installer LabInvent

- **Utiliser le script oven.php au lieu du seul composer.json**
  Une autre manière rapide d'installer CakePHP est d'utiliser **Oven**. Il s'agit d'un **simple script PHP qui vérifie si vous respectez les recommandations systèmes**, **installe le squelette d'application CakePHP et met en place l'environnement de développement.** This is not a deployment script, it's aimed to help developers installing CakePHP for the first time and get a working development environment up and running in seconds. Production environments should consider several other factors, like file permissions, virtualhost configuration, etc.

  - Upload oven.php file to your host/server
  - Access oven.php on your host via your favourite browser such as http://localhost/oven.php
  - Click the big oven knob in the middle of the page
  - CakePHP will auto install and ready to go

- **Améliorer le script installation.sh**
  Phase 1 : supprimer les **setacl**, pas compatibles avec tous les OS
  Phase 2 : éviter de poser des questions, et mettre plutôt les paramètres demandés (login et pass mysql, login utilisateur labinvent, nom BD, …) dans un fichier à part style environment.txt (qu'on peut fournir avec des valeurs par défaut…)
  ⇒ cela suppose aussi de **détecter automatiquement si la BD existe** déjà ou pas (si elle n'existe pas, on la crée, et si elle existe on ne la crée pas, mais on crée seulement les tables, sauf si option "BD_RECREER_TABLES" est à NON)
- **Phase 3** : réécrire le script d'installation en php (install.php, au lieu d'un script shell bash) pour assurer une portabilité sur tout OS (pourvu que php, mysql, et apache soient présents)
- **Phase 4** : rendre le script install.php exécutable depuis le web pour qu'on puisse procéder à l'installation même si on n'a pas d'accès ssh (une fois qu'on a envoyé le site web sur le serveur web via ftp)
  ⇒ il faudra alors saisir les paramètres du fichier environment.txt sur cette page web
  ⇒ devrait éviter l'utilisation des "sudo" pour les chmod sur les répertoires, vu que le projet devrait déjà appartenir au serveur web

- Phase 5 : séparer l'instance de développement (pour le développeur, sur son pc local) et l'instance site web (sur le serveur web, qui peut aussi être local pour le développeur) : actuellement on mélange les 2, d'où le besoin compliqué de rendre l'instance modifiable à la fois par le développeur et le serveur web… (d'où les sudo et chmod…)
    - L'instance du développeur sera modifiable par lui seulement, et sera un dépot git (donc git pull et git push possibles)
    - L'instance web sera modifiable seulement par le serveur web, et ne sera pas un dépot git, mais mise à jour par ftp (ou rsync) depuis l'instance du développeur (phase de "déploiement") : on pourra d'ailleurs avoir 2 instances web, une de test, et une de prod ; Problème : comment faire les mises à jour de la BD (surtout les modifs structurelles) ? il faudra que ces mises à jour soient faites par des scripts php exécutables via le web...

- **DOCKER** :
  L'utilisation d'un conteneur docker permettra de supprimer les "sudo", et d'avoir des login et pass par défaut pour mysql (qu'on pourra modifier avant l'installation)
  Au lancement, on teste si l'appli est déjà installée ou pas : si pas installée, on lance la page web install.php pour procéder à l'installation (en récupérant la dernière version du projet sur le gitlab) ; si déjà installée, on lance le site web
  Il faudra prévoir une option permettant de mettre à jour le projet à la dernière version du gitlab
  Image serveur web nginx = Alpine (minimaliste)
  Image php : fastcgi (démarre php7-fpm et écoute sur port 9000), au lieu de Zend OPcache

- **GITLAB** : nouveau groupe irap et sous-groupe labinvent (irap/labinvent)

3. Installations existantes (liste des labos)

(EP updated 11/2/21)

Ce chapitre recense (dans l'ordre chronologique) les installations faites de **LabInvent** dans différents laboratoires CNRS, ainsi que les paramétrages particuliers de ces installations

	Laboratoire	**Nom donné au logiciel** (et aux groupes thématiques/ métiers)	**Versions et dates**		**LDAP**	**Titreuse**
			LabInvent	**Composants**		
2012	**IRAP version développement sur Mac OS**	**InventIrap**	Dernière version (branche dev)	**Mysql** 15.1 Distrib 10.3.12-Maria DB **PHPUnit** 5.7.27 **PHP** 7.3.11	fake	oui
2012	**IRAP** (Institut de Recherche en Astrophysique et Planétologie) - **Toulouse** https://www.irap.omp.eu (laboratoire d'origine du logiciel) Laurence Lavergne Etienne Pallier, Elodie Bourrec (branche dédiée dev-irap)	**InventIrap**	Dernière version prod (branche master) v4.105.X **Première installation** 2012 (v1.x)	(2020 07) **MySql** Server 5.1.73 **Php** 5.6	Oui OpenLdap (anonyme)	oui

2014	**IAS** (Institut d'Astrophysique Spatiale) - **Paris** https://www.ias.u-psud.fr Sandrine Couturier Stéphane Caminade **Particularités** : Numéro d'inventaire **SANS année** (ex: IAS-750 au lieu de IAS-2016-750) Chez eux les agents créent la fiche inventaire AVANT que la commande ne soit passée et de plus pour l'inventaire de matériel plus ancien qui rentre dans la partie « suivi technique » ils ont rarement la date de commande. Le champ **date commande doit donc être facultatif**. C'est la gestionnaire qui complète la fiche par la suite avec des droits « administration » (notamment pour cette date) Superadmin doit pouvoir voir la partie réservée à l'administration (ex: n° inventaire organisme...) Un équipement peut appartenir à un **groupe thématique (= projet**) sans appartenir à un **groupe métier (= service**) même si à la fin du projet il est censé être mis à disposition du labo et donc rattaché à un service (groupe métier). Du coup le responsable ne voit pas les équipements de son groupe thématique (quand ils ne sont pas rattachés à un groupe métier)... Il faut donc que sur la page d'accueil du "responsable", il voit tous les matériels	**LabInvent** Groupes thématiques = "Projets" Groupes métiers = "Service - Equipe"	Dernière version prod (branche master) v4.105.X **Première installation** Fin 2014 (v1.x)	**MySql** ? PHP 7.3.19-1~deb10u1 **PHPUnit 5.7.27**	Oui OpenLdap (anonymei	oui même que IRAP, mais peu utilisée (utilisent plutot une dymo manuelle)

	rattachés à son groupe thématique (OU à son groupe métier) intéressés par le module MÉTROLOGIE du LATMOS (mais non activé)					
2017	**IPSL-LATMOS** (Laboratoire Atmosphères, Milieux, Observations Spatiales) - **Paris** https://www3.latmos.ipsl.fr Marie-Sophie Clerc Christophe Dufour, Yann Delcambre (branche dédiée dev-latmos) **Contribution importante** : ajout d'un plugin "métrologie" activable depuis la page de configuration (désactivé par défaut) ; malheureusement encore inachevé	**LabInvent**	**Première installation** juin 2017 => v2.7.0 (stagiaire LATMOS Alexis Proust et stagiaire IRAP Thibaud Ajas 3 mois)	**MySql** ? **Php** 7.?	? je sais plus...	? je sais plus...
2018	**CRAL** (Centre de Recherche Astrophysique de Lyon) - **Lyon** https://cral.univ-lyon1.fr initiateur : Lionel Capoani (parti à l'IP2I fin 2020) Informaticienne : Marie-Claire Marthinet Responsable Administrative : Bérengère Chamoret Superadmin : Matthieu Guibert (ingénieur mécanicien) (stagiaire CRAL Jeanne Prugniel 6 semaines - janvier-février 2020) **Contribution importante** : aide à la transformation de l'application en version	**CRALinvent**	Mise à jour le 23/6/20 (v3.7.9.42) Mise à jour le 12/3/**20** (v3.7.9.0) => mobile Mise à jour juin 2019 (v2.13.x < 2.13.12) => pdf + ldap optim (cache) **Première installation** Fin 2018	**MySql** ? **Php** 7.?	Oui ActiveDirectory (authentifié)	oui

	compatible mobile (avec support important de l'IRAP quand même)					
2021 (01)	**IP2I (Institut Physique 2 Infinis) (IPNL et LMA, 250 pers) - IN2P3** Lionel Capoani (dir. technique) Yoan Giraud et Bruno CARLUS (informaticiens) **Contribution importante** : **dockerisation** de l'application (en cours)	**LabInvent**	2020-01	Conteneur **docker** avec Php 7, Mysql 5.6 (?), et Nginx	Oui Active Directory (authentifié)	Oui (19mm)

4. Installation via Docker (solution la plus simple)

(17/2/21 - EP)

La façon la plus simple d'installer LabInvent est de le faire avec Docker. Ceci est désormais possible grâce à un image docker (linux) réalisée par le laboratoire IP2I en février 2021. En outre, cette image linux permet une installation multi-plateformes du logiciel car il est ainsi possible de l'installer sur Linux, Mac, ou Windows. L'image docker installera pour vous un environnement linux autonome qui héberge LabInvent et permet de le configurer. Toutefois, si vous préférez une installation classique, veuillez passer à la section suivante "installation des pré-requis".

4.1. Installation

La première chose à faire bien sûr est d'installer le logiciel Docker : https://docs.docker.com/get-docker/

Ensuite, voici la procédure à suivre pour l'installation :

1 - Cloner le dépot :

$ git **clone https**://gitlab.in2p3.fr/ip2i/docker/labinvent.git

*(ou bien si vous préférez **ssh**, après avoir enregistré votre clé ssh sur le gitlab https://gitlab.in2p3.fr/-/profile/keys : "git clone git@gitlab.in2p3.fr:ip2i/docker/labinvent.git")*

2 - Aller dans le dossier du logiciel cloné :

$ cd labinvent/

3 - (optionnel) Changer la configuration par défaut :

Attention, par défaut l'installation prendra en compte les variables d'environnement du fichier **docker-compose.yml** (qui correspondent à peu près à celles du fichier .env-sample). Si ces valeurs par défaut ne vous conviennent pas, vous devez faire une copie du fichier .**env.sample** et le nommer .**env** :

- cp .env.sample .env

C'est ce fichier qui sera lu désormais pour les variables d'environnement utilisées par docker-compose.yml.

Ce fichier .env est votre copie “privée” de .env.sample, il est ignoré par git (donc non versionné)

- **MODE_BATCH=ON**

  Ce fichier contient une variable **MODE_BATCH** activée par défaut (=ON), ce qui signifie que **par défaut l’installation se fera directement SANS poser aucune question**. Si vous préférez le mode interactif (questions/réponses), alors désactiver le mode batch en mettant à OFF :

  MODE_BATCH=OFF

- **MYSQL_ROOT_LOGIN=root**

  Par défaut, on considère que vous avez le mot de passe de l’administrateur mysql. Si ce n’est pas le cas, désactiver la variable (attention, cela suppose que la BD est déjà créée pour vous et que vous avez un login défini pour y accéder)

  MYSQL_ROOT_LOGIN=OFF

- HTTP_PORT=80

  Si vous ne voulez pas que le serveur web réponde sur le port 80, changez le port, prenez par exemple 8081 (car phpmyadmin si vous l’utilisez est servi par défaut sur le 8080)

  HTTP_PORT=8081

4 - Démarrer le container :

Avant de démarrer le container docker, vous devez choisir parmi 2 options très différentes :

- (Option 1) (par défaut) soit le dossier source du logiciel LabInvent est caché car il est dans le conteneur : c’est l’option la plus simple et la plus adaptée en mode “**production**”
- (Option 2) soit vous voulez que le dossier source soit accessible localement sous le nom “./data”, ce qui est une option plus pratique en mode “**développement**” (le dossier source est toujours dans le conteneur, mais il est aussi accessible depuis un dossier local, qui n’est qu’une “vue” sur le dossier du conteneur, et permet donc de modifier le code source plus facilement ; en tant que développeur, c’est l’option que je choisis)
  Pour activer cette option (2), il suffit d’écraser le fichier docker-compose.yml actuel en le remplaçant par le fichier docker-compose.localsource.yml :
  $ cp docker-compose.localsource.yml docker-compose.yml

Maintenant que votre choix est fait, on peut démarrer le conteneur docker :

$ ./DOCKER_LABINVENT_START

(équivalent à docker-compose up -d)

Creating network "labinvent_labinvent-network" with driver "bridge"

Creating labinvent-db ... done

Creating labinvent ... done
Creating labinvent-webserver ... done

… A partir d'ici, **attendre un peu** pour que le clonage du dépôt soit complètement terminé …
(notamment, si vous avez choisi l'option (2) plus haut, attendez que le dossier local ./data/ apparaisse)

Chaque fois que vous modifierez le fichier .env (ou docker-compose.yml), il faudra relancer le conteneur docker :

$ ./DOCKER_LABINVENT_RESTART
(équivalent à docker-compose down && docker-compose up -d)

*Si on veut **voir les conteneurs** qui tournent :*
*$ docker **ps -a***

*Si on veut **voir les logs** du conteneur :*
$ docker-compose logs -f labinvent

5 - Lancer l'installation et l'exécution du logiciel

Attention, **par défaut l'installation se fera en mode NON interactif**. Si on veut un mode interactif, il faut désactiver la variable MODE_BATCH dans votre fichier .env (voir plus haut comment changer la config).

Installation :
$ ./DOCKER_LABINVENT_INSTALL
*(équivalent à docker exec -it -w /var/www/install labinvent ./**installation**.sh)*
Si vous êtes en mode interactif, laisser les réponses par défaut à quasiment toutes les questions posées, sauf :

- Nom du serveur BD : **db** (et non pas "localhost")
- Admin mysql ? oui

6 - (optionnel) Déclarer un compte ldap comme superadmin :

Si vous projetez d'utiliser l'authentification via un annuaire LDAP, vous devez d'abord déclarer un compte utilisateur ldap (le vôtre par exemple) comme étant Super Administrateur :

$ docker exec -it labinvent ./ADD_NEW_SUPERADMIN_LDAP_USER

D'autres infos sur la procédure à suivre, ici.

4.2. Exécution & Arrêt

Si vous venez juste de faire l'installation, le site est déjà disponible à l'adresse http://localhost:8081
(souvenez-vous que j'ai changé la configuration et que j'ai remplacé le port 80 par 8081, mais pour vous c'est peut-être toujours 80 ?)

De manière générale, se placer dans le dossier du projet docker, puis :

$ ./DOCKER_LABINVENT_START **-> lancement des conteneurs**
$ ./DOCKER_LABINVENT_STOP **-> arrêt des conteneurs**
$ ./DOCKER_LABINVENT_RESTART **-> Redémarrage des conteneurs**

Exemple pour arrêter le conteneur docker :

$ ./DOCKER_LABINVENT_STOP
(équivalent à docker-compose down)

Stopping labinvent-webserver ... done
Stopping labinvent ... done
Stopping labinvent-db ... done
Removing network labinventdock_labinvent-network

4.3. Installation de l'environnement de DEV

Par défaut c'est l'environnement de PROD qui est installé (**docker-compose**.yml)
L'environnement de dev est le même que PROD mais il AJOUTE seulement quelques outils tels que : phpmyadmin, ldap, et éventuellement un mapping mysql
Pour ajouter ces suppléments à l'installation par défaut (PROD), il suffit de faire ceci :

$ DOCKER_LABINVENT_STOP (**si les conteneurs sont déjà lancés**)
$ **cp docker-compose.dev**.yml **docker-compose.override**.yml
Adapter le script **docker-compose.override**.yml si besoin (*)
$ DOCKER_LABINVENT_START
(plus d'infos sur cette technique de "surcharge" de docker-compose ⇒ https://docs.docker.com/compose/extends/)

*(*) Dans ce fichier, le mapping mysql est proposé mais désactivé par défaut car il ne devrait pas être nécessaire puisque cette configuration de dev ajoute un **phpmyadmin** donnant accès au serveur mysql et que vous pouvez aussi utiliser les script DOCKER_LABINVENT_DB_USER ou DOCKER_LABINVENT_DB_ROOT pour accéder au serveur en mode commande via le terminal du host.*
Le mapping ne serait utile que si vous avez vraiment besoin d'accéder au serveur mysql docker directement depuis l'extérieur du conteneur (avec la commande "mysql" du host local ou avec un client externe style phpmyadmin…).
Une fois ce mapping activé, vous pourriez faire la chose suivante :

(en root login)
$ mysql -h 127.0.0.1 -u $MYSQL_ROOT_LOGIN -P3307 -p$MYSQL_PASSWORD
Ou
(en labinvent user login)
$mysql -h 127.0.0.1 -u $MYSQL_USER -P3307 -p$MYSQL_PASSWORD

4.4. Accès à la BD LabInvent

Vous pouvez accéder à la BD LabInvent de 2 façons :

- En mode commande (commande "mysql") via les scripts DOCKER_LABINVENT_DB_USER (accès seulement à la BD LabInvent) ou DOCKER_LABINVENT_DB_ROOT (accès à tout le serveur en root)
- Via un client PhpMyAdmin à l'adresse http://localhost:8080 (si vous avez installé l'environnement de dev, voir chapitre précédent) :
  - Accès seulement à la BD projet : user "labinvent_user", pass "labinvent" (sauf si vous avez changé ces valeurs dans le fichier .env)
  - Accès à tout le serveur MySql : user "root", pass "labinvent" (sauf si vous avez changé ces valeurs dans le fichier .env)

4.5. Autres instructions utiles par la suite

- **Pour exécuter une commande <cmd> dans le conteneur** :

  ```
  $ docker exec -it labinvent <cmd>
  ```

- Pour **mettre à jour le logiciel** à la dernière version disponible sur le dépot :

  ```
  $ ./DOCKER_LABINVENT_UPDATE
  ```

(équivalent à docker exec -it labinvent ./UPDATE*)*

- Pour **se connecter directement sur le conteneur** (comme si on faisait un ssh sur le serveur):
  $ ./DOCKER_LABINVENT_SHELL
  (équivalent à docker exec -it labinvent /bin/bash*)*

- Pour **réinstaller** à partir de la dernière version disponible :
  $ ./DOCKER_LABINVENT_UPDATE
  $ ./DOCKER_LABINVENT_RESTART
  A partir de là, si on veut réinstaller :
  $ ./DOCKER_LABINVENT_INSTALL

- Pour **refaire une installation 'propre'** (à partir de l'étape 4) :
  $ ./DOCKER_LABINVENT_STOP
  $ docker **volume** rm labinvent_data
  $ docker **volume** rm labinvent_db
  A partir de là, on peut reprendre à l'étape 4 ci-dessus.
  (au prochain DOCKER_LABINVENT_START *il va repartir de 0 et donc faire un clone du dépôt du logiciel labinvent)*
  Explication :
  - **labinvent_data** c'est le volume docker qui contient /var/www
  - **labinvent_db** c'est le volume avec la base de données

  (remplacer "labinvent_data" par "labinventdock_data" si le dossier du projet s'appelle labinventdock par exemple)

- Pour **supprimer les données (les volumes, par exemple labinvent_data ou labinvent_db) :**
  $ docker volume rm <volume>

- **Pour remplacer php7.2 (par défaut) par php7.4** :

  $ cd php/

  remplacer "FROM php:7.2-fpm" par "FROM php:7.4-fpm" dans php/Dockerfile avant

  $ docker build -t gitlab-registry.in2p3.fr/ip2i/docker/labinvent .

- **Pour monter un volume local (au lieu d'un volume docker)** :
  ⇒ voir le sujet suivant (transférer un fichier local)

- **Pour transférer un fichier LOCAL (de mon disque local) vers un volume docker (par exemple le volume labinvent_data) :**

  Il suffit de trouver où sont stockés les fichiers du volume labinvent_data sur mon disque avec la commande

  - $ docker volume inspect labinvent_data

    (normalement c'est indiqué dans les lignes Mountoint)

  Problème sur Mac : docker tourne dans une VM et donc le volume n'est pas accessible par défaut.
  Donc, il faut **monter un répertoire local** (et non pas un volume docker) et ensuite se connecter dans le conteneur pour copier les fichiers.
  Pour monter un répertoire local, dans **docker-compose.yml** on ajoute une ligne aux volumes, par exemple :

```
    #PHP Service

    labinvent:
        image: gitlab-registry.in2p3.fr/ip2i/docker/labinvent
        …
        volumes:
            - data:/var/www

            - ./php/local.ini:/usr/local/etc/php/conf.d/local.ini

            - ./local:/mnt/local
        networks:
        ...
```

Cela va monter le répertoire **“local”** (au même niveau que le fichier docker-compose.yml) dans le répertoire /mnt/local du conteneur. Ca peut servir de dossier tampon. Avec cette manip, on se connecte sur le conteneur (docker exec -it labinvent /bin/bash) et on peut alors copier les fichiers dans /mnt/local On les retrouvera dans le dossier “local” (par contre il y aura peut etre un pb de droits)

4.6. Informations diverses utiles à la compréhension

- Décryptage de l’instruction “docker exec -it -w /var/www/install labinvent ./**installation**.sh” :
  - docker exec
    - permet de lancer une commande dans le conteneur
  - -it
    - pour avoir un truc interactif (genre quand on lance un script qui pose des questions, c'est bien de pouvoir répondre)
  - -w /var/www/install
    - (working dir) ça lui dit de se mettre dans ce répertoire
  - labinvent
    - le nom du conteneur dans lequel il faut exécuter la commande
  - et enfin, la commande à lancer

5. Installation SANS Docker

5.1. Installation des pré-requis

(created 25/1/19 EP - updated 13/1/20 EP)

Pour information, voir les dates de support prévues pour chaque pré-requis

5.1.1. Plateformes testées

Ce logiciel est **multiplateforme** (Linux, MacOS, et presque possible sur Windows), mais la plateforme de prédilection est Linux car c'est ce qui est utilisé pour la production dans tous les laboratoires où ce logiciel a été diffusé. D'autre part, la distribution linux conseillée est CentOS (et ses dérivés).
Pour le développement, on utilise plutôt Mac OS X, mais ça peut très bien se faire aussi sur Linux. On pourrait aussi le faire sur Windows 10, mais il faudrait "juste" pour cela convertir le script d'installation bash en PHP. Quelqu'un veut s'y coller ?

- **LINUX :**
  - **Fedora 20** (version test Thibault Ajas, IRAP, avril 2017)
  - **Centos 6.6** (version de "production", SI IRAP) :
    - PHP : 5.6.22
    - Mysql : 5.1.73-3.el6_5
    - Apache : 2.2.15-39.el6.centos
  - **Debian GNU/Linux 8.5** (jessie) (version de "production", IAS) :
    - PHP : 5.6.22
    - Mysql : MariaDB 10.0.25
    - Apache : 2.4.10
  - **Scientific Linux (=Centos) 6.4** (version dev/test Etienne Pallier linux, IRAP) :
    - PHP : 5.6.30
    - MYSQL : 5.5.56
    - APACHE : 2.2.15
  - **UBuntu 14.04.4** (Ancienne version dev/test Alexandre Cases, IRAP) :
    - PHP : 5.5.9 (ne suffit plus)
    - MYSQL : 5.5.47

        - Apache : 2.4.7

- **<u>MAC OS X</u> :**
    - (20/1/21) **Mac OS 11.1 (Big Sur) avec brew** (version dev/test Etienne Pallier, IRAP)
    - (25/1/19) **Mac OS 10.14 (Mojave) avec brew** (version dev/test Etienne Pallier, IRAP)
        - OS Darwin Kernel Version 18.2.0: Mon Nov 12 20:24:46 PST 2018; root:xnu-4903.231.4~2/RELEASE_X86_64 x86_64
        - Apache/2.4.37 (Unix)
        - Mysql (MariaDB) Ver 15.1 Distrib 10.3.12-MariaDB, for osx10.14 (x86_64) using readline 5.1
        - PHP 5.6.40 (cli) (built: Jan 16 2019 14:53:29) ET php 7.3.1, avec Zend Engine v2.6.0, Copyright (c) 1998-2016 Zend Technologies with Zend OPcache v7.0.6-dev, Copyright (c) 1999-2016, by Zend Technologies
    - **(17/1/18) Mac OS 10.13.2 avec brew** (version dev/test Etienne Pallier, IRAP) :
        - PHP 7.2.0 + MySQL 5.7.20 + Apache 2.4.28
    - **Mac OS 10.12.5 avec XAMPP** 5.6.3 et 7.1 (version dev/test Etienne Pallier, IRAP) :
        - PHP 5.6.3 + MySQL 5.6.21 + Apache 2.4.10
        - PHP7.1.6 + MariaDB 10.1.24 + Apache 2.4.25

- **<u>WINDOWS 10</u>** (TODO)

5.1.2. Installation (pré-requis)

Le logiciel nécessite une combinaison "AMP" pour fonctionner, soit les 3 pré-requis suivants :

- *un serveur web **Apache** (récent), ou Nginx*
- *un serveur de base de données **Mysql** (récent)*
- *le langage **Php en version 7.x** (php 5.6 est encore supporté mais plus pour longtemps)*
- *Optionnel mais très recommandé : **PhpMyAdmin***
- *le gestionnaire de versions **git***
- *(N'installez PAS le framework CakePhp, il sera installé automatiquement pour vous !)*

Si ces logiciels sont déjà présents sur votre OS, vous pouvez passer à l'étape suivante (Installation du logiciel), et revenir ici seulement en cas de problème de configuration.

Sur Windows, vous pouvez utiliser Wampserver ou XAMPP qui regroupent les 3 premiers éléments (il n'y aura rien d'autre à faire ensuite).

Sur Mac, vous pouvez installer chacun des 3 premiers éléments séparément via HomeBrew (ou MacPort), ou plus simplement utiliser XAMPP ou MAMP, ou encore télécharger le paquet binaire Mac correspondant à chaque élément. Personnellement, j'ai utilisé HomeBrew.

5.1.2.1. Installation sur Mac OS (avec HomeBrew)

⇒ Aller à l'annexe correspondante : Installation des pré-requis sur Mac OS X (avec HomeBrew)

5.1.2.2. Installation sur Windows 10

⇒ Aller à l'annexe correspondante : Installation des pré-requis sur Windows 10

5.1.2.3. Installation sur Linux

5.1.2.3.1. Exemple pour une distribution UBuntu 18.04.3 LTS

(fait en **janvier 2020**, par E. Pallier, lors de l'encadrement de la stagiaire Jeanne Prugniel)

Version des composants LAMP :

- **Linux Ubuntu 18.04.3** : utilisateur "labinv" (avec sudo) avec password = **'labinv'**
- **Apache 2.4.29**
- **Mysql** 14.14 (**5.7.28**) : password root utilisé = **'PassWord,0'** *(chiffre zéro)*
- **Php 7.2.24**
- **PhpMyAdmin :** password utilisé (le même que mysql root)
- **Git 2.17.1**

Choix faits pour la BD inventaire sur cette VM :

- nom BD = **labinvent**
- nom utilisateur labinvent du logiciel = **labinventuser**
- password de cet utilisateur = '**PassWord,0'** *(chiffre zéro)*
- *Autres utilisateurs définis :*
    - *superadmin, pass=login*
    - *… (pass=login)*

Installation AMP (Apache, Mysql, Php) :
https://www.digitalocean.com/community/tutorials/comment-installer-la-pile-linux-apache-mysql-php-lamp-sur-un-serveur-ubuntu-18-04-fr
(voir aussi https://doc.ubuntu-fr.org/lamp)

Installation PhpMyAdmin :
https://www.digitalocean.com/community/tutorials/how-to-install-and-secure-phpmyadmin-on-ubuntu-18-04

(mdp mysql et phpmyadmin vm jeanne : PassWord,0)

En résumé :

- **GIT**
  - **Installation:**

    $ sudo apt install git

    $ git --version

- **APACHE**
  - **Installation :**

    $ sudo apt update

    $ sudo apt install apache2
  - **Ajuster votre pare-feu afin d'autoriser le trafic web :**

    $ sudo ufw app list

    Si vous regardez sur le profil Apache Full, il devrait y être indiqué qu'il permet le trafic aux ports 80 et 443 :

    $ sudo ufw app info "Apache Full"

    Autoriser le trafic HTTP et HTTPS entrant pour ce profil :

    $ sudo ufw allow in "Apache Full"

    Vous pouvez immédiatement effectuer une vérification :

    Avec firefox, se connecter à "http://localhost"

    ⇒ On devrait voir une page web avec "It works !", ce qui indique que votre serveur web est maintenant bien installé et qu'il est accessible à travers votre pare-feu.

- **MYSQL**
  - **Installation :**

    $ sudo apt install mysql-server

    $ sudo mysql_secure_installation

  - **Configuration** :

Veuillez noter que pour les systèmes Ubuntu fonctionnant avec MySQL 5.7 (et les versions ultérieures), l'utilisateur **root** MySQL est configuré par défaut pour authentifier en utilisant le plugin auth_socket, plutôt qu'avec un mot de passe. Cela permet d'avoir une meilleure sécurité et ergonomie dans de nombreux cas, mais il peut également compliquer les choses lorsque vous devez autoriser l'ouverture d'un programme externe (ex : phpMyAdmin) afin d'accéder au serveur. Si vous préférez utiliser un mot de passe lorsque vous vous connectez au MySQL en tant que **root**, vous aurez besoin de changer le mode d'authentification de auth_socket à mysql_native_password. Pour y parvenir, ouvrez le prompt MySQL à partir de votre terminal :

```
$ sudo mysql

mysql> SELECT user,authentication_string,plugin,host FROM mysql.user;

mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'PassWord,0';

mysql> FLUSH PRIVILEGES;

mysql> SELECT user,authentication_string,plugin,host FROM mysql.user;

mysql> exit
```

- Tester la connexion en tant que root (sans utiliser "sudo") :

  ```
  $ mysql -u root -p
  ```

- **PHP**
  - **Installation** :

    ```
    $ sudo apt install php libapache2-mod-php php-mysql
    ```

  - **Configuration :**

    Actuellement, si un utilisateur demande un répertoire du serveur, Apache recherchera d'abord un fichier nommé index.html. Nous voulons dire au serveur web de donner priorité aux fichiers PHP, ainsi il faut exiger à Apache de regarder pour un fichier index.php en premier.

Editer /etc/apache2/mods-enabled/dir.conf

Cela va ressembler à cela :

/etc/apache2/mods-enabled/dir.conf

<IfModule mod_dir.c>

DirectoryIndex **index.html** index.cgi index.pl index.php index.xhtml index.htm

</IfModule>

Déplacer le fichier d'index PHP (surligner ci-dessous) à la première position après la spécification DirectoryIndex, de la manière suivante :

<IfModule mod_dir.c>

DirectoryIndex **index.php index.html** index.cgi index.pl index.xhtml index.htm

</IfModule>

Ensuite, redémarrer le serveur web Apache afin que vos modifications prennent effet.

$ sudo systemctl restart apache2

$ sudo systemctl status apache2

vous avez l'option d'installer de modules supplémentaires si besoin. Pour voir les options disponibles de modules PHP et de bibliothèques :

$ apt search php- | less

Pour en savoir plus sur la fonctionnalité d'un module :

$ apt show package_name

- **Test:**

  Afin de tester si votre système est configuré correctement pour PHP, créer un script PHP de base appelé **info.php**. Afin qu'Apache puisse localiser ce fichier et le desservir correctement, il devra être sauvegardé dans un répertoire bien spécifique, qui se nomme le "web root" : /var/www/html/. Mettre dans ce fichier le contenu suivant :

  ```
  <?php

  phpinfo();

  ?>
  ```

  Se connecter à "http://localhost/info.php" pour voir le résultat

- **PHPMYADMIN**
  - **Installation:**

    $ sudo apt install phpmyadmin php-mbstring php-gettext

    - For the server selection, choose apache2
    - Select Yes when asked whether to use dbconfig-common to set up the database
    - You will then be asked to choose and confirm a MySQL application password for phpMyAdmin

    The installation process adds the phpMyAdmin Apache configuration file into the **/etc/apache2/conf-enabled/** directory, where it is read automatically. The only thing you need to do is explicitly enable the mbstring PHP extension, which you can do by typing:

    $ sudo phpenmod mbstring

    Afterwards, restart Apache for your changes to be recognized:

    $ sudo systemctl restart apache2

When you installed phpMyAdmin onto your server, it automatically created a database user called **phpmyadmin** which performs certain underlying processes for the program. Rather than logging in as this user with the administrative password you set during installation, it's recommended that you log in as either your **root** MySQL user or as a user dedicated to managing databases through the phpMyAdmin interface.

You can now access the web interface by visiting your server's domain name or public IP address followed by /phpmyadmin:

http://localhost/phpmyadmin

Se connecter en tant que user "root" (c'est à dire le mysql root user)

(on peut aussi utiliser le user "phpmyadmin", mais il est moins privilégié)

- **BUGFIXES**:

  Cette version de phpmyadmin n'est pas assez récente pour php7.2, il faut donc corriger 2 lignes du code source selon ce qui est indiqué ici : https://doc.ubuntu-fr.org/phpmyadmin#incompatibilite_avec_php_72

5.1.2.3.2. Exemple pour ScientificLinux (CentOS) 6.4 (fait en juillet 2017)

ATTENTION, SUR CENTOS (ou dérivé), il vaut mieux désactiver selinux

Pour mettre à jour PHP de la 5.6 à la 7.1 :
--> https://blog.remirepo.net/post/2016/12/05/Install-PHP-7.1-on-CentOS-RHEL-or-Fedora

```
$ sudo yum update kernel
$ sudo yum update

$ sudo yum install yum-utils
$ sudo yum-config-manager --enable remi-php71
$ sudo yum update
```

=> mais il y a un conflit à cause de phpmyadmin, donc je supprime ce package :
$ sudo yum erase phpmyadmin
$ sudo yum update

$ php -v => 7.1

Redémarrage Apache:
$ sudo /etc/init.d/httpd restart

Tentative de réinstaller phpmyadmin
$ sudo yum install phpmyadmin
=> toujours un conflit, je laisse tomber, dommage...

Tentative d'accélérer php 7:
(https://community.1and1.com/php-7)
$ sudo yum install php71-php-opcache
Creer un repertoire .opcache/ dans le webroot/ du projet:
$ cd webroot/
$ mkdir .opcache/
$ chmod 777 .opcache/

Modifier php.ini :
$ sudo vi /etc/php.ini
Ajout des lignes suivantes :
; EP added this for opcache (Jul 2017):
zend_extension=opcache.so;
opcache.enable=1;opcache.memory_consumption=32;
opcache.interned_strings_buffer=8;
opcache.max_accelerated_files=3000;
opcache.revalidate_freq=180;
opcache.fast_shutdown=0;
opcache.enable_cli=0;

opcache.revalidate_path=0;
opcache.validate_timestamps=2;
opcache.max_file_size=0;
opcache.file_cache=/projects/labinvent/labinvent2/webroot/.opcache;
opcache.file_cache_only=1;

Faire un lien vers l'extension opcache.so
$ cd /usr/lib64/php/modules/
$ sudo ln -s /opt/remi/php71/root/usr/lib64/php/modules/opcache.so

Verifier que .opcache/ contient bien des données (du cache)

Mais je ne vois pas vraiment d'accélération...

5.1.2.3.3. Exemple pour CentOS 6.7 (fait en 2016)

ATTENTION, SUR CENTOS, il vaut mieux désactiver selinux

Mettre à jour le serveur:
$ sudo yum kernel
(restart)
$ sudo yum update

Pour installer Apache, MySQL & PHP 5.3 :
--> https://www.zerostopbits.com/how-to-install-apache-mysql-and-php-on-centos-6-7/

Pour mettre à jour PHP de la 5.3 à la 5.6
--> https://www.zerostopbits.com/how-to-upgrade-php-5-3-to-php-5-6-on-centos-6-7/

Mettre à jour Mysql (version 5.1 à 5.5):
$ sudo yum update

5.1.2.3.4. Exemple pour une distribution UBuntu 14.04.4 (fait en 2016) et 14.04.5 (Mai 2017)

/!\ Par défaut, la version de php installée ici est php5.5 qui ne suffit plus. Si vous souhaitez installer la version 5.6, remplacez TOUS les "php5" par "php5.6", et si vous voulez la version 7.1, remplacez TOUS les "php5" par "php7.1" /!\

Pour commencer il faut mettre à jour les "repository" de apt :
$ sudo apt-get update && sudo apt-get upgrade

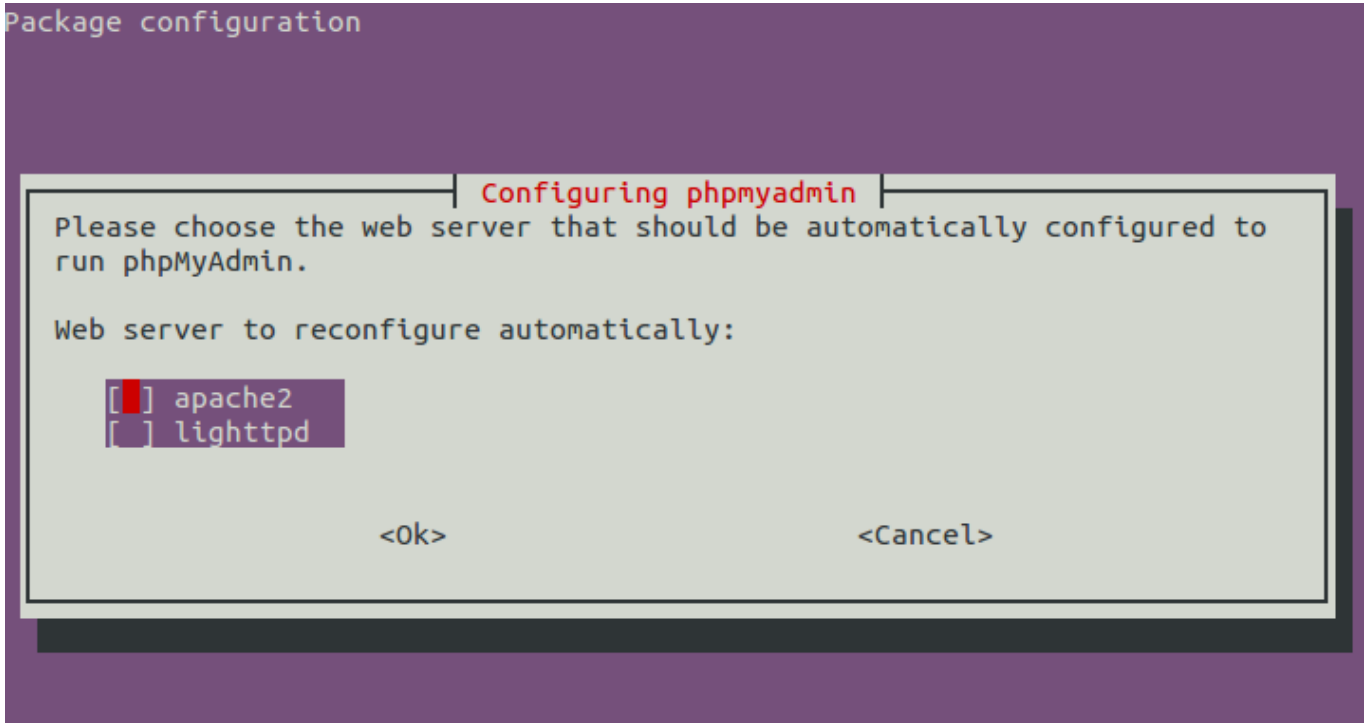
Installer un serveur web (Apache) :
$ sudo apt-get install apache2

Installer un serveur de base de données (MySQL):
$ sudo apt-get install mysql-server

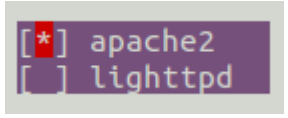
Installer le langage PHP en version 5.5.9 minimum (5.6 recommandé)
$ sudo apt-get install php5 php-pear
$ sudo apt-get install php5-mysql

Installer phpmyadmin et le configurer
$ sudo apt-get install phpmyadmin
$ sudo dpkg-reconfigure -plow phpmyadmin

/!\ Lorsque vous aurez l'écran suivant, n'oubliez pas d'appuyer sur la touche "espace" avant la touche "entrée" /!\



Afin d'avoir cela :



Si, en visitant http://localhost/phpmyadmin/ vous avez l'erreur "The mcrypt extension is missing. Please check your PHP configuration.", executez les commades suivantes :

$ sudo apt-get install php5-mcrypt
$ sudo ln -s /etc/php5/conf.d/mcrypt.ini /etc/php5/mods-available
$ sudo php5enmod mcrypt
$ sudo service apache2 restart

5.1.2.3.5. Exemple pour une distribution Fedora 20+ (fait en avril 2017)

(https://www.digitalocean.com/community/tutorials/how-to-install-lamp-linux-apache-mysql-php-on-fedora)

Pour commencer il faut mettre à jour l'OS :

$ sudo dnf update dnf
$ sudo dnf update kernel
$ sudo dnf update

Installer git:

$ sudo dnf install git

Installer php :

$ sudo dnf install php
$ sudo dnf install php-mysql

Installer un serveur web (Apache) :

$ sudo dnf install httpd
$ sudo systemctl enable httpd
(ln -s '/usr/lib/systemd/system/httpd.service' '/etc/systemd/system/multi-user.target.wants/httpd.service')
$ sudo systemctl start httpd

Installer un serveur de base de données (MySQL):

$ sudo dnf install mariadb mariadb-server -y
$ sudo systemctl enable mariadb
(ln -s '/usr/lib/systemd/system/mariadb.service' '/etc/systemd/system/multi-user.target.wants/mariadb.service')
$ sudo systemctl start mariadb
$ sudo mysql_secure_installation

(OPTIONNEL) Installer phpmyadmin (par defaut accessible uniquement depuis localhost), utile pour gérer plus facilement la BD :

$ sudo dnf install phpmyadmin
$ sudo systemctl restart httpd

Pour Ubuntu:
sudo apt-get php5-mcrypt
sudo apt-get install phpmyadmin
Pensez à activer l'extension mcrypt : sudo php5enmod mcrypt

5.1.3. (Optionnel) Configuration post Installation (pré-requis)

5.1.3.1. Configuration de Php (fichier php.ini)

Le fichier php.ini se trouve dans :

- /etc/php7/apache2/ sur Ubuntu
- /etc/ sur CentOS
- /usr/local/etc/php/<version>/ sur MacOS avec HomeBrew (version = 5.6, 7.1, 7.2, 7.3, …)
- ...

Ldap :

Il se peut que vous deviez installer puis activer l'extension ldap

```
extension=ldap
```

Dossier de log (optionnel):

Positionner votre répertoire de log :

```
error_reporting = E_ALL
error_log = /var/log/php/error.log
max_input_time = 30
```

Ensuite il vous faudra peut-être créer le dossier en question et donner à Apache les droits sur ce dossier (www-data pour Ubuntu, apache pour CentOS...):

```
$ sudo mkdir /var/log/php
$ sudo chown www-data /var/log/php
```

Configuration du fuseau horaire (timezone)

Pas sûr que ça soit nécessaire, mais ça peut pas faire de mal de positionner au timezone Paris :

[Date]
; Defines the default timezone used by the date functions
; http://php.net/date.timezone
;date.timezone =
date.timezone = **Europe/Paris**

Pour info seulement, il y a aussi ces paramètres.
Voici ceux que j'ai dans mon php.ini (php 7.3), valeurs par défaut, je n'y ai pas touché :

; Default timestamp format.
ibase.timestampformat = "%Y-%m-%d %H:%M:%S"

; Default date format.
ibase.dateformat = "%Y-%m-%d"

; Default time format.
ibase.timeformat = "%H:%M:%S"

Recharger la config du serveur Web

⇒ **Pour prendre en compte ces modifs, recharger la configuration du serveur Web**

$ sudo service httpd reload
(CentOS: $ sudo systemctl reload httpd)
(MacOS: $ sudo apachectl restart)

5.1.3.2. Configuration de MySql

A priori rien à faire...

5.1.4. (Optionnel) Installation d’un IDE (environnement de développement)

Ce pré-requis est optionnel et ne concerne que les développeurs qui souhaiteraient contribuer au développement du logiciel LabInvent avec un certain confort.
Nous vous conseillons d’installer l’IDE Eclipse avec le plugin PyDev.
Voici un lien qui peut vous aider pour cela : https://www.linuxrouen.fr/wp/programmation/developpement-python-avec-eclipse-et-pydev-introduction-21165/

Bien sûr vous pouvez utiliser un autre IDE dédié à Php.
Si vous avez déjà l’habitude d’un environnement, gardez le, sinon installez Eclipse.

5.2. Installation du logiciel LabInvent

Une fois les pré-requis bien en place, on peut procéder à l'installation du logiciel lui-même.

NB: Pour faire l'installation directement depuis l'IDE Eclipse, aller à l'annexe 25.3. Installation et configuration du logiciel depuis l'IDE Eclipse

5.2.1. Récupération du logiciel

Avant toute chose, placez-vous dans le dossier où vous voulez installer le logiciel LabInvent.

Deux options se présentent à vous :

- a) **soit vous récupérez une version statique** du logiciel, en le téléchargeant (pas besoin de login, c'est anonyme) : considérez alors cette version comme une **version de test jetable car il vous faudra recommencer pour obtenir chaque nouvelle version**, pas très pratique donc, mais rapide
- b) (**méthode préférée**) **soit vous récupérez une version synchronisée**, avec git (nécessité d'avoir un login), ce qui vous permettra de rester constamment à jour (sans réinstallation), et même de contribuer à l'évolution du logiciel si vous le désirez

5.2.1.1. (NON RECOMMANDÉ) Téléchargement anonyme (version statique, non synchronisée, jetable)

Vous pouvez télécharger la version actuelle du logiciel.

Pour cela, aller sur : https://gitlab.irap.omp.eu/epallier/labinvent/tree/master

Cliquez sur : "Download zip" dans le coin en haut à droite.

Double-cliquez dessus ou dézippez-le (ou lancez la commande gzip -d labinvent.zip). Vous devriez avoir un dossier "labinvent.git".

Dans sa documentation, le logiciel sera désigné par "LABINVENT".

Vous pouvez renommer "labinvent.git" en "labinvent" si vous le souhaitez ("mv labinvent.git labinvent" ou clic droit->Renommer).

5.2.1.2. (RECOMMANDÉ) Récupérer le logiciel via GIT (version dynamique, synchronisée, nécessité d'un login)

Pour cette option, vous devez avoir un login. Si vous n'avez pas déjà un login, allez sur la page https://gitlab.irap.omp.eu/users/sign_in,

puis remplissez la section "Sign up". Ensuite, envoyez un email à epallier et à ebourrec (AT irap.omp.eu)

en demandant l'autorisation d'accéder au gitlab du projet labinvent. On vous donnera alors la procédure à suivre pour vous connecter.

(Si vous utilisez Windows, vous DEVEZ avant tout installer git pour windows, voir plus bas la section "Seulement pour windows")

En récupérant directement le logiciel via git, vous allez avoir une version dynamiquement synchronisée. Vous serez donc en mesure de la mettre à jour dès qu'une nouvelle version sera disponible avec la commande "git pull".

Aller dans le dossier où vous voulez installer LabInvent puis téléchargez-le via git :

> $ **git clone https://gitlab.irap.omp.eu/epallier/labinvent.git labinvent**

(Ou aussi depuis ssh, seulement au sein de l'IRAP : git clone git@gitlab.irap.omp.eu:epallier/labinvent.git labinvent)

(Si vous récupérez le projet pour la première fois, git vous demandera un login et un mot de passe)

Si vous obtenez ce message d'erreur ... :

> fatal: unable to access 'https://gitlab.irap.omp.eu/epallier/labinvent.git/': Peer's certificate issuer has been marked as not trusted by the user.

Essayez une de ces 2 solutions en tapant les commandes suivantes :

- solution la plus simple mais aussi radicale :
  $ git config --global http.sslVerify false http

- solution un peu plus compliquée mais plus soft :
  $ mkdir -m 0700 ~/.gitcerts
  $ echo | openssl s_client -connect gitlab.irap.omp.eu:443 -servername gitlab.irap.omp.eu 2>/dev/null | openssl x509 > ~/.gitcerts/gitlab.irap.omp.eu.crt
  $ git config --global credential.helper 'cache --timeout=3600'
  $ git config --global http.sslCAinfo ~/.gitcerts/gitlab.irap.omp.eu.crt

Puis essayez à nouveau de cloner le projet (commande “git clone” ci-dessus).

Enfin, ajoutez vos infos personnelles :
  $ git config --global user.email "Vous@exemple.com"
  $ git config --global user.name "Votre Nom"

Vérifiez que votre configuration est OK :
  $ git config --list
  (ou encore : cat ~/.gitconfig)

Cela devrait afficher quelque chose comme :
  http.sslverify=true
  http.sslcainfo=/home/labinv/.gitcerts/gitlab.irap.omp.eu.crt
  credential.helper=cache --timeout=3600
  user.email=...
  user.name=Etienne Pallier
  core.repositoryformatversion=0
  core.filemode=false
  core.bare=false
  core.logallrefupdates=true
  remote.origin.url=https://gitlab.irap.omp.eu/epallier/labinvent.git
  remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
  branch.master.remote=origin
  branch.master.merge=refs/heads/master

```
branch.dev.remote=origin
branch.dev.merge=refs/heads/dev
...
```

Git a normalement créé un dossier "**labinvent**" qui contiendra votre projet (avec un sous dossier ".git" qui sert à la synchronisation avec le dépôt git).

ATTENTION:

Par défaut, vous êtes sur la **branche "master"** du git. Elle contient une version stable du logiciel. Si vous désirez seulement utiliser ce logiciel SANS LE MODIFIER, alors restez sur la branche "master", vous y serez très bien ;-). Attention seulement de ne faire AUCUNE modification sur la branche master !!!

Par contre, **si vous souhaitez contribuer au développement de ce logiciel, et donc le modifier**, **vous devez absolument changer de branche et vous placer sur la branche "dev"** (ou bien une sous-branche dédiée comme "dev-IRAP", ou "dev-LATMOS") :

```
$ cd labinvent/
$ git branch
$ git checkout dev
$ git branch
```

Si jamais votre dossier "labinvent" appartient à root (vous avez fait un "git clone" depuis root…, c'est pas bien !!), il serait préférable que vous en soyez vous-même (ou un autre user) le propriétaire :

```
sudo chown -R nom_utilisateur labinvent/
```

Seulement pour Windows :

Pour pouvoir faire les manipulations décrites ci-dessus sur Windows, il faut d'abord installer git.

- Téléchargez git sur https://git-scm.com/download/win
- Lancez l'installation (gardez la configuration par défaut)
- Une fois installé, lancer une invite de commande (Touche Windows+R, tapez cmd, touche entrer) pour commencer à taper des commandes git

Vous pouvez désormais utiliser git depuis une invite de commande ou depuis l'interface graphique de git.

5.2.2. (Optionnel) Gérer le projet avec Eclipse

Vous pouvez ignorer cette section si vous n'utilisez pas Eclipse (ou si vous ne comptez pas contribuer au logiciel)

Voici comment importer le projet LabInvent dans Eclipse (juste faire pointer Eclipse sur votre dossier LabInvent pour pouvoir le gérer depuis Eclipse) :

File/Import
General/Existing project into workspace
[Next]
Select root dir : le dossier contenant le projet labinvent

(peut-être à faire pour résoudre le pb récurrent de "missing node.js" : http://download.eclipse.org/wildwebdeveloper/releases/latest/)

5.2.3. Installation du logiciel

(EP updated 28/1/2021)

L'installation du logiciel s'effectue via la simple exécution d'un script.
Avant d'exécuter ce script, si vous n'êtes pas administrateur du serveur de base de données (MySql), vous devez demander à l'administrateur de vous créer une base de données du nom de "labinvent" (par exemple) ainsi qu'une deuxième base de données qui servira uniquement à l'exécution des tests (par exemple "labinvent_test"), et de vous donner tous les droits sur cette base de données (et sur la BD de test) avec un login et un mot de passe (par exemple, un login "labinvent_user"). Ces informations vous seront demandées par le script d'installation.
Si par contre vous êtes administrateur du serveur de base de données (vous avez le login root et son mot de passe), vous n'avez rien à faire, vous pouvez exécuter directement le script d'installation.

Par défaut, le script d'installation s'exécutera en mode INTERACTIF, c'est à dire qu'il vous posera des questions.
Si vous préférez une installation en mode BATCH (silencieux), il suffit de créer un fichier **ENV.sh** contenant les variables d'environnement dont l'installation a besoin :

$ cd install/
Le fichier **ENV.sample.sh** sert de template pour créer votre fichier **ENV.sh** :
$ **cp ENV.sample.sh ENV.sh**
Le fichier ENV.sh est votre copie privée, il est ignoré par git

Exécution du script :
Il suffit d'exécuter le script **installation.sh** depuis le dossier install/ :

$ cd install/
$ **./intallation.sh**

NB :

- *Attention: sur **Mac OS X**, utiliser installation-macos.sh au lieu de installation.sh*
- *pas besoin d'exécuter ce script en tant que "root"*
- *Conseil pour le mode interactif : à la plupart des questions, laissez les réponses par défaut)*

5.2.4. Contrôles rapides après installation

- **Verifier que la BD d'inventaire (par défaut "labinvent" sauf si vous lui avez donné un autre nom) à bien été créée (avec phpmyadmin par exemple)**

- **Tester le site web avec le serveur web de développement inclus**
  Attention, ne faites aucune action sur cette page, c'est juste pour vérifier qu'on accède bien au site
  En attendant de configurer votre serveur web (Apache), vous pouvez déjà voir à quoi ressemble le logiciel en utilisant le serveur web de dev inclus :
  Depuis la racine du projet
  $ chmod +x TEST_WEB
  $ ./TEST_WEB
  ⇒ CONNECTEZ-VOUS MAINTENANT A http://localhost:8765
  Attention, ne faites aucune action sur cette page, c'est juste pour vérifier qu'on accède bien au site
  CTRL-C pour stopper
  On ne peut malheureusement pas utiliser ce mode rapide de visualisation, c'est juste un aperçu.
  Il faut passer par une configuration du serveur web Apache que l'on verra plus loin.

5.2.5. Contenu du projet

(updated 18/12/18 - EP)

Voici la description des dossiers et fichiers de l'application

Fichier ou Dossier	Description
README.md	Le fichier README général du projet, contient l'historique des versions importantes
TESTS.sh	Script d'exécution des tests (à exécuter régulièrement, après chaque modif)
bin/	Quelques fichiers binaires du framework CakePhp (à utiliser uniquement pendant la phase de développement)
composer.json	Le fichier de description de tous les plugins utilisés par CakePhp et installés automatiquement par Composer
config/	Dossier des fichiers de configuration, notamment app.php qui décrit la configuration de la base de données, des mails, et des tests
database/	Dossier contenant le schéma de base de données pour l'installation ainsi que les scripts de mise à jour à exécuter lorsque la BD a été modifiée (sous-dossier "update")
doc/	Quelques documents, notamment celui-ci
index.php	Le point d'entrée du site web
install/	Le dossier pour l'installation contenant notamment le script général d'installation "installation.sh"
logs/	Les logs de l'application (utiles en pour le debug)
src/	

tests/	C'est le dossier principal contenant tout le code source de l'application LabInvent pour CakePhp (il aurait aussi pu s'appeler "app"). Voir ci-dessous pour le contenu de ce dossier.
tmp/	Le dossier contenant tous les tests exécutés par le script TESTS.sh (voir ci-dessus)
vendor/	Dossier utilisé par LabInvent comme cache des pages web et des tables de la BD
webroot/	Tous les plugins pour CakePhp installés automatiquement par Composer. Lors d'une première récupération du projet, ce dossier est vide. Une fois l'installation faite, ce dossier sera plein de sous-dossiers, un par plugin. Par exemple, vous y trouverez le plugin **composer**/ lui-même, ainsi que le plugin principal **cakephp**/ et le plugin **phpunit**/ qui sert à exécuter les tests (on y trouve même un peu des frameworks symfony et zend...). Le dossier contenant toutes les "ressources" ou éléments utilisés par les pages web, comme les images, les feuilles de style CSS pour la mise en page, les modules javascript pour l'animation des pages, les documents attachés, etc.

Description du dossier principal de l'application src/ :

Controller/	Dans un framework MVC (Modèle-Vue-Contrôleur) tel que CakePhp, le **Contrôleur (le C du MVC)** contient toute la logique métier, en gros presque tout le code de traitement des données. Il récupère une demande d'un utilisateur (qui clique sur une page web), réclame les données nécessaires pour cette demande à la couche Modèle (voir ci-dessous), et les passe à la couche Vue (voir ci-dessous) pour qu'elle les présente à l'utilisateur (une nouvelle page web par exemple). Ce dossier contient donc tous les contrôleurs, un par entité (matériel, utilisateur, fournisseur…), en gros un par table de la BD (mais pas toujours). Par exemple le contrôleur MaterielsController est responsable de traiter les demandes concernant l'entité "matériel". Comme vous pouvez vous en douter, c'est le plus gros contrôleur de LabInvent, étant donné qu'il est centré sur la ressource "matériel". C'est lui qui contient les méthodes CRUD (Create, Read, Update, Delete) permettant de gérer un matériel. Il contient aussi les méthodes permettant de changer le statut d'un matériel pour gérer son cycle de vie (CREATED => VALIDATED => TOBEARCHIVED => ARCHIVED).

Model/ - Behavior/ - **Entity**/ - **Table**/ **Template**/ (et View/)	Tous les contrôleurs sont des classes qui héritent de la classe **AppController** qui définit un ensemble de comportements par défaut pour tout contrôleur. Notamment, tout comportement général d'un contrôleur doit être déplacé dans AppControlleur plutôt que d'être répété (recopié) dans chaque contrôleur. AppController lui-même hérite de **Controller** qui est une super-classe de CakePhp offrant elle-même beaucoup de comportements par défaut de tout contrôleur (comme les opérations standard CRUD). Contrairement à AppController, il vaut mieux ne pas modifier la classe Controller car elle risque d'évoluer avec chaque nouvelle version de CakePhp. **C'est la couche "M" (Model)** du pattern MVC implémenté par le framework CakePhp. Cette couche est responsable de lire ou écrire les données demandées ou rendues par la couche C (Contrôleur), quelque soit le système de gestion de données sous-jacent (MySql, ou PostGreSql, ou Sqlite, ou simplement des fichiers, etc.). Elle assure ainsi un niveau d'abstraction par rapport à ce système sous-jacent, qui permet de s'en rendre indépendant et même de le remplacer facilement si besoin était. Le niveau d'abstraction est même poussé un peu plus loin en offrant un mapping relationnel-objet (ORM) qui permet de manipuler les données sous forme d'objets plutôt qu'avec des requêtes SQL plus difficiles à écrire et à maintenir. Ainsi, par exemple, $materiel->save() permettra tout simplement d'enregistrer un objet matériel (représentant une ligne de la table) dans la table matériels, en le créant s'il n'existait pas déjà, ou en le mettant à jour sinon. Le dossier Model/ contient notamment les sous-dossiers Entity/ et Table/ : - le sous-dossier **Entity**/ contient un fichier par table qui est une représentation en Php de la structure de la table. Chacun de ces fichiers définit une classe qui hérite de la super-classe **Entity** de CakePhp. Une fois ce fichier écrit, on peut alors générer la table SQL correspondante dans la base de données. Toute modification de la table devrait donc être faite ici plutôt que directement dans la table elle-même. Cela permet la portabilité du projet d'un SGBD à un autre étant donné que cette représentation Php est indépendante de tout formalisme spécifique à un SGBD particulier. - le sous-dossier **Table**/ contient lui aussi un fichier par table, mais décrit plutôt la façon dont les champs (colonnes) de la table doivent être validées. Par exemple, les règles qui permettent de valider un "email" avant de l'enregistrer dans le champ "email" de la table. Chacun de ces fichiers définit une classe qui hérite de la classe **AppTable** (qui elle-même hérite de la super-classe **Table** de CakePhp, à ne pas toucher vu qu'elle peut évoluer avec les nouvelles versions du framework) qui définit quelques règles de validation générales utilisables pour toutes les tables (par exemple, la validation d'un

email, ou d'une chaine de caractères pour définir les caractères interdits par défaut...).

C'est la couche "V" (View) du pattern MVC implémenté par le framework CakePhp (pour plus d'infos, voir https://book.cakephp.org/3.0/fr/views.html). Elle est responsable de présenter les données la plupart du temps sous forme d'une page web. Ces données lui sont passées par la couche (C)ontrôleur. Le dossier View/ (normalement utilisé pour y placer des classes de vue, sous-classes de AppView, elle-même sous-classe de View ; par exemple, PdfView) n'étant pas utilisé pour le moment dans LabInvent, nous ne parlerons que du dossier **Template**/. Celui-ci contient **des sous-dossiers portant chacun le nom du contrôleur qui va l'utiliser**. Les fichiers templates (.ctp) qui sont dans ces sous-dossiers portent un nom correspondant à l'action effectuée. Par exemple, le fichier de vue pour l'action « view() » du controller Products devra normalement se trouver dans **src/Template/Products/view.ctp.**
Un template (.ctp) est un fichier HTML qui peut en plus contenir des instructions particulières dans un format de template propre à CakePhp. Il construit la page web qui sera présentée à l'utilisateur. En général, il y a au moins un fichier par ACTION faite par un contrôleur. Par exemple, les actions CRUD (Create, Read, Update, Delete) sur une table donnée sont permises grâce à des vues dédiées (pages web) telles que (resp.) les templates **add.ctp, view.ctp/index.ctp, et edit.ctp**, l'action Delete n'ayant en général pas besoin d'une page web pour être exécutée (donc a priori pas de template delete.ctp). Les templates view.ctp et index.ctp représentent l'action Read respectivement sur UNE ligne de la table ou sur TOUTES les lignes. Ensuite, on peut avoir d'autres templates selon les besoins, tels que **find.ctp** par exemple pour afficher une page web de recherche des matériels.
Dans le dossier **Template**/, on trouve quelques sous-dossiers particuliers :

- **Element**/ : contient notamment des parties (blocs) de pages web réutilisés dans différentes pages
- **Error**/ : contient les pages web qui sont affichées en cas d'erreur
- **Fichemetrologiques/, Formules/, et Unites**/ : contiennent toutes les vues spécifiques pour le module Métrologie
- **Layout**/ : contient notamment la page web **default.ctp** qui sert de page de base (template) à toutes les pages web du site ; c'est elle qui donne la structure générale des pages du site (menu à gauche, header, footer, …) ; cette page contient aussi le numéro de version et la date de LabInvent qui doit être mise à jour à chaque modification du projet
- **Pages**/ : contient toutes les "pages" web simples (plutôt "statiques) du site, comme la page "**home**.ctp" (page d'accueil), la page "**about**.ctp", la page "**tools**.ctp" (menu

	Outils), la page “**printers**.ctp” (présentation de l’étiqueteuse), ou la page **infos**.ctp (informations systèmes concernant le système hôte, accessible uniquement au profil SuperAdmin) - **QrCodes**/ : page creer.ctp permettant la création du QrCode associé à un matériel, sur chaque page web de visualisation d’une fiche matériel La couche vue de CakePHP peut être constituée d’un certain nombre de parties différentes. Chaque partie a différents usages qui seront présentés dans ce chapitre : ● **templates**: Les templates sont la partie de la page qui est unique pour l’action lancée. Elles sont la substance de la réponse de votre application. ● **elements** : morceaux de code de view plus petits, réutilisables. Les elements sont habituellement rendus dans les vues. ● **layouts** : fichiers de template contenant le code de présentation qui se retrouve dans plusieurs interfaces de votre application. La plupart des vues sont rendues à l’intérieur d’un layout. ● **helpers** : ces classes encapsulent la logique de vue qui est requise à de nombreux endroits de la couche view. Parmi d’autres choses, les helpers de CakePHP peuvent vous aider à créer des formulaires, des fonctionnalités AJAX, à paginer les données du model ou à délivrer des flux RSS. ● **cells**: Ces classes fournissent des fonctionnalités de type controller en miniature pour créer des components avec une UI indépendante. Regardez la documentation View Cells pour plus d’informations.

5.3. Démarrage (re-) des services nécessaires

(EP 30/12/2020)

Avant de pouvoir se connecter au site web de l'application, si ce n'est déjà fait, activez tous les services nécessaires :

- Vérifier que php est bien installé (même si ce n'est pas un service, c'est une dépendance)
- Vérifier le Serveur de Gestion de Base de Données : MySql (ou MariaDB)
- Vérifier le Serveur Web : Apache (bien configuré avec Php)

5.3.1. Sur LInux

(TODO)
On peut s'inspirer de ce qui est écrit pour Mac OS ci-dessous.

5.3.2. Sur Mac OS

Pour avoir une description d'un soft (formule), faire "brew info nom-du-soft", exemple : brew info php

Voir les services qui sont actuellement actifs

$ brew services list

```
httpd   stopped
mariadb  stopped
php     stopped
```

php@5.6 stopped
php@7.1 stopped
php@7.2 stopped

NB: En mode “développeur”, il n’est pas nécessaire de faire en sorte que ces services soient actifs au boot (et donc en permanence), on peut les démarrer seulement au besoin. Donc, c’est très bien que ces services soient stoppés.

Démarrage des services nécessaires (voir surtout la 3e colonne “START”) :

	PATH (bin)	CONFIG	START

Apache (httpd)	$ httpd -v :
	Apache/2.4.46
	
	/usr/local/bin/**httpd** -> ../Cellar/httpd/2.4.46/bin/httpd
	
	$ brew info httpd
	=> httpd: stable 2.4.46
	=> /usr/local/Cellar/httpd/2.4.46
	=> DocumentRoot is /usr/local/var/www
		/usr/local/etc/httpd/
		
		DocumentRoot is /usr/local/var/www
		
		The default ports have been set in /usr/local/etc/httpd/httpd.conf to 8080 and in /usr/local/etc/httpd/extra/httpd-ssl.conf to 8443 so that httpd can run without sudo
			To have launchd start httpd now and restart at login:
			$ **brew services start httpd**
			
			Or, if you don't want/need a background service you can just run:
			$ (sudo) **apachectl (re)start**
			*(**stop** pour arrêter le service)*
			*(aussi possible : **sudo apachectl -k restart**)*

MySql (MariaDb)	On installe plutôt MariaDB (en fait “mysql” pointe sur mariadb)
	
	/usr/local/bin/**mariadb** -> ../Cellar/mariadb/10.5.8/bin/mariadb
	
	/usr/local/bin/**mysql** -> ../Cellar/mariadb/10.5.8/bin/mysql (lien vers mariadb)
	
	$ mysql -V
	=> mysql Ver 15.1 Distrib 10.5.6-MariaDB
	
	$ brew info mariadb
		/usr/local/etc/my.cnf
		
		Warning à l’installation :
		A "/etc/my.cnf" from another install may interfere with a Homebrew-built server starting up correctly
			To have launchd start mariadb now and restart at login:
			brew services start mariadb
			
			Or, if you don't want/need a background service you can just run:
			$ **mysql.server start**
			*mysqld_safe Logging to '**/usr/local/var/mysql/macp1219.local.err**'.*
			201026 15:34:00 mysqld_safe Starting mariadbd daemon with databases from /usr/local/var/mysql

	=> mariadb: stable 10.5.6 => /usr/local/Cellar/mariadb/10.5.6		*SUCCESS!* *Connexion :* $ **mysql -uroot**
PHP	$ php -v /usr/local/bin/**php** -> ../Cellar/php@7.2/7.2.34_1/bin/php Si on veut changer de version de php : $ **sphp <version>** (par exemple : sphp 7.4 ou sphp 74)	- INI : The **php.ini** and **php-fpm.ini** file can be found in: /usr/local/etc/php/7.2/ - Apache : LoadModule php7_module /usr/local/opt/php@7.2/lib/httpd/modules/libphp7.so $ brew info php => php: stable 7.4.11 => /usr/local/Cellar/php/7.4.11 => /usr/local/etc/php/7.4/ : contient php.ini et php-fpm.ini	*(normalement pas nécessaire)* To have launchd start php@7.2 now and restart at login: **brew services start php@7.2** **(ou** *brew services start php)* Or, if you don't want/need a background service you can just run: **php-fpm**
PhpMyAdmin	/usr/local/Cellar/phpmyadmin/5.0.4 URL : http://localhost/phpmyadmin	- INI : usr/local/etc/phpmyadmin.config.inc.php - Apache : Alias /phpmyadmin /usr/local/share/phpmyadmin	**N/A** A faire pour supprimer le warning du style : “ne peut accéder à /tmp, phpmyadmin sera lent car pas de cache possible”

			$ mkdir /usr/local/Cellar/phpmyadmin/5.0.4/share/phpmyadmin/tmp $ chmod 777 ...
L'info suivante (sur python) n'est pas utile pour ce projet			
Python	/usr/local/bin/**python** -> ../Cellar/**python@2**/2.7.17/bin/python /usr/local/bin/**python3** -> ../Cellar/**python@3.9**/3.9.1_2/bin/python3 Install Python packages with: **pip3 install** <package> ⇒ will install into: /usr/local/lib/**python3.9/site-packages**	???	**N/A**

5.4. Accès local à l’application web

(EP 01/02/2021)

Comment accéder à l’application web LabInvent depuis le navigateur web du poste d’installation ?

Maintenant que l'application est installée localement sur votre pc ou serveur, il y a 3 moyens d'accéder (localement) au site web de l'application LabInvent :

- (1) Soit classiquement via "http://localhost/labinvent"
- (2) Soit via un port dédié tel que par exemple "http://localhost:8081"
- (3) Soit via un nom de serveur dédié tel que par exemple "http://labinvent.test"

Voici comment faire dans chacun des 3 cas :

- **Cas (1) :**
  - soit vous déplacez le dossier "labinvent/" directement dans le repertoire des sites webs du serveur web local (DocumentRoot, en général c’est /var/www/html/ ou /var/www/html/localhost/public_html/, ...)
  - soit vous laissez ce dossier là où il est et :
    - vous faites un lien depuis votre DocumentRoot (/var/www/html/ ou autre) vers ce dossier labinvent/
    - ou bien vous indiquez au serveur web où le trouver via une directive dans le fichier de configuration du serveur web httpd.conf :

```
Alias /labinvent /Users/epallier/Documents/_W_more/PROJECTS/LABINVENT/SOURCE/labinvent2

<Directory /Users/epallier/Documents/_W_more/PROJECTS/LABINVENT/SOURCE/labinvent2/>
    Options FollowSymLinks
    AllowOverride All
    Require all granted
```

```
</Directory>
```

- **Cas (2)** : utiliser un Virtual Host avec un port dédié (ici 8081) :

```
Listen 8081
<VirtualHost *:8081>
<Directory /PROJECTS/LABINVENT/SOURCE/labinvent2>
   Options FollowSymLinks
   AllowOverride All
   Require all granted
</Directory>
      ServerAdmin epallier AT irap...
      DocumentRoot "/PROJECTS/LABINVENT/SOURCE/labinvent2"
      # logs are relative to <ServerRoot> :
      ErrorLog logs/labinvent2.error_log
      CustomLog logs/labinvent2.access_log combined
</VirtualHost>
```

- **Cas (3)** : utiliser un Virtual Host avec un nom de serveur (ici labinvent.test) :

  Il faut d'abord ajouter le nom du serveur (ici labinvent.test) à la fin de votre fichier **/etc/hosts** :

```
127.0.0.1  (TAB)     labinvent.test
```

  Attention, j'ai déjà essayé un nom tel que "labinvent.dev", CA NE MARCHE PAS !!! (par contre, "labinvent.devv" oui !!!)

Vérifier que ce nouveau host est bien pris en compte par le système :

```
$ ping labinvent.test
```

MacOS : il se peut qu'il faille effacer le cache DNS pour que la nouvelle entrée de /etc/hosts soit prise en compte :

$ sudo killall -HUP mDNSResponder

Voire même :

$ sudo dscacheutil -flushcache; sudo killall -HUP mDNSResponder

Puis créer un hôte virtuel de ce genre (sur MacOS/Homebrew, dans /usr/local/etc/httpd/extra/httpd-vhosts.conf)

```
<VirtualHost *:80>
    ServerName labinvent.test
    ServerAdmin votre-email@toto.com
    # Mettre ici le chemin vers votre dossier labinvent :
    DocumentRoot "/home/labinv/PROJECTS/LABINVENT/labinvent"
    # Logs are relative to <ServerRoot> :
    #ErrorLog logs/labinvent2.error_log
    #CustomLog logs/labinvent2.access_log combined
    ErrorLog ${APACHE_LOG_DIR}/labinvent_error.log
    CustomLog ${APACHE_LOG_DIR}/labinvent_access.log combined
    # ATTENTION, mettre ici exactement le meme chemin que
    # dans DocumentRoot ci-dessus (vers votre dossier labinvent) :
    <Directory /home/labinv/PROJECTS/LABINVENT/labinvent>
        Options FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

(pour **Ubuntu**, ce texte doit être ajouté à la fin du fichier /etc/apache2/sites-enabled/000-default.conf)

Dans les 3 cas, redémarrer le serveur web suite à ces modifications :

Sur Ubuntu :

$ sudo systemctl restart apache2

(sur MacOS/Homebrew : $ sudo apachectl restart)

Passez maintenant au chapitre suivant pour exploiter cette configuration.

6. Vérification de la conformité du site web LabInvent

(EP 01/02/2021)

Le site est installé, il vous faut maintenant le configurer.

Si vous voulez le conserver, le script de création de la BD est **database/build.sql** (s'il existait déjà avant votre installation, il a été sauvegardé dans build.sql.ORIG). Votre fichier de configuration est **config/app.php** (s'il existait déjà avant votre installation, il a été sauvegardé dans app.php.ORIG).

Très important : tout d'abord, il faut s'assurer que le cache et les logs sont toujours accessibles en écriture :

```
$ chmod -R 777 tmp/cache/
$ chmod -R 777 logs/
```

Maintenant, on peut tester l'accès web à l'application.

Si vous avez installé le logiciel via Docker, vous accédez au site web à l'adresse "http://localhost:8081"

Sinon, vous y accédez de l'une des ces 3 façons :

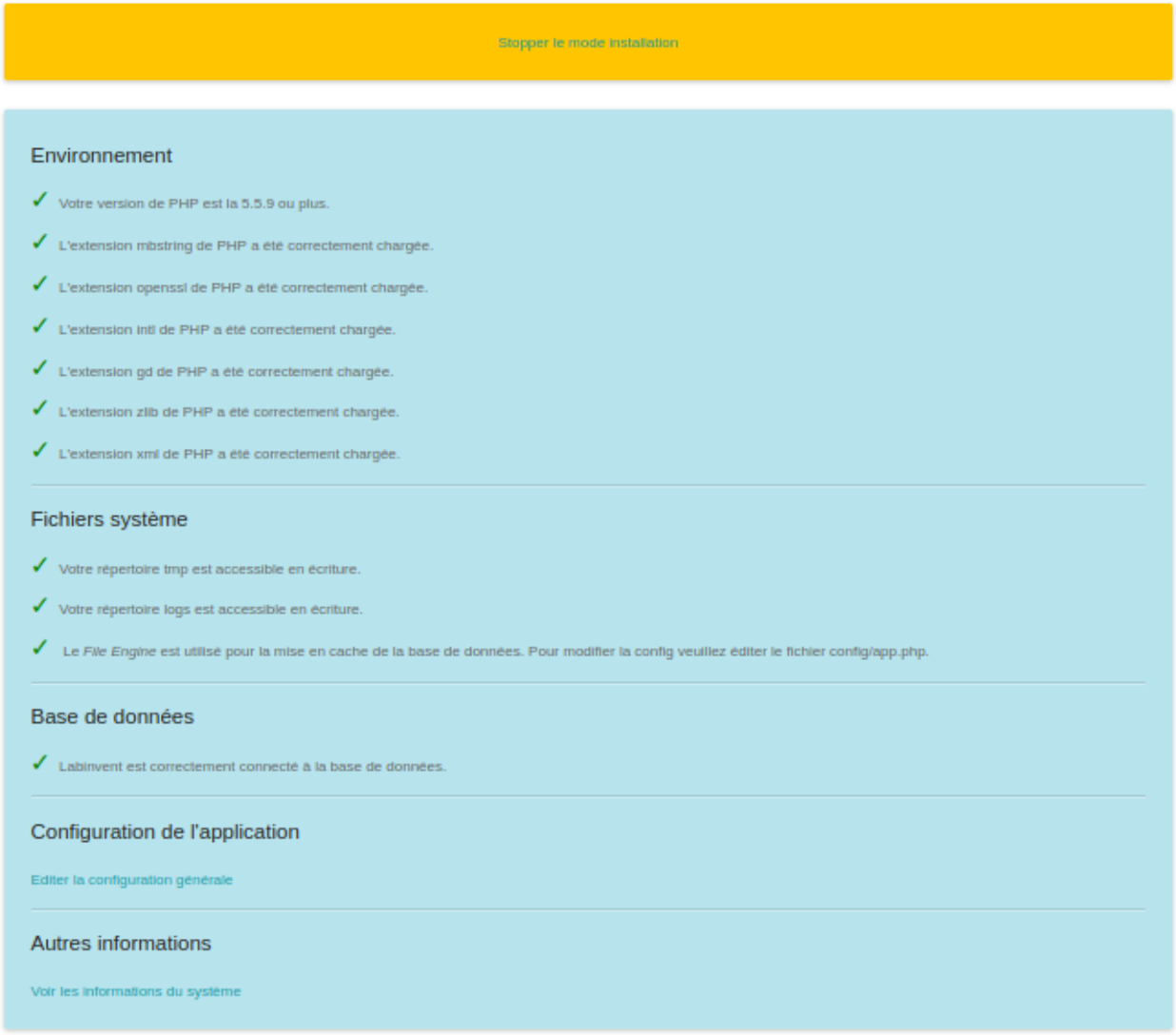
- http://localhost/labinvent (cas 1)

- ou bien http://localhost:8081 (cas 2)

- ou bien http://labinvent.test (cas 3, voir chapitre précédent)

Vous devriez arriver par défaut sur une page correspondant au mode install, elle vous indique la bonne configuration de l'application et de ses dépendances.

Vérifier simplement que **TOUS LES POINTS SONT AU VERT**, comme ci-dessous :



(La page affichera un message si l'URL-Rewriting ne fonctionne pas correctement).

Si tout est au vert, passez au chapitre suivant (Fin de l'installation).

Sinon (il y a des points rouges), reportez-vous ci-dessous à l' (ou les) extension(s) qui pose problème, pour l'installer ou la configurer :

- **<u>Conformité de votre php.ini</u> :**

  Taper la commande suivante pour voir si tout va bien avec votre fichier php.ini

  $ php -i > /dev/null

  Si aucun message ne s'affiche, tout va bien.

  Par contre, si vous obtenez ce genre de warning concernant votre "timezone" :

  *PHP Warning: Unknown: It is not safe to rely on the system's timezone settings. You are *required* to use the date.timezone setting or the date_default_timezone_set() function. In case you used any of those methods and you are still getting this warning, you most likely misspelled the timezone identifier. We selected the timezone 'UTC' for now, but please set date.timezone to select your timezone. in Unknown on line 0*

  Alors, vous devez définir votre fuseau horaire dans le fichier php.ini :

  date.timezone = Europe/Paris

  Pour savoir où est le fichier php.ini:

  $php -r "print phpinfo();" | grep ".ini"

  (sur XAMPP, c'est dans /Applications/XAMPP/xamppfiles/etc/php.ini)

  ⇒ **Maintenant, recharger la configuration du serveur Web, et recharger la page web pour voir si le problème est réglé :**

  $ sudo service httpd reload

  (Ubuntu: $ sudo systemctl restart apache2)

  (CentOS: $ sudo systemctl reload httpd)

  (MacOS: $ sudo apachectl restart)

  ($ sudo service apache2 restart

- **URL-Rewriting (autoriser les fichiers .htaccess)**

  Pour info, page dédiée pour Cakephp : https://book.cakephp.org/3/fr/installation.html#url-rewriting

  Activer le mode rewrite de Apache:
  (Ubuntu)
  $ sudo a2enmod rewrite

  Allez aussi dans le fichier /etc/apache2/apache2.conf (ou httpd.conf sur CentOS), ou dans votre configuration “virtual host”, et vérifier que la propriété 'AllowOverride' soit à la valeur 'All' pour le dossier Labinvent :

```
#<Directory votre-chemin-vers-labinvent>
<Directory /var/www/html/labinvent>
        Options FollowSymLinks
        AllowOverride All
</Directory>
```

  ⇒ **Recharger la configuration du serveur Web, et recharger la page web pour voir si le problème est réglé**

  Si ça ne règle pas le problème, il est aussi possible que le mode rewrite ne fonctionne pas correctement si les fichiers .htaccess dans la racine et dans webroot disparaissent. Sur certains systèmes les fichiers commençant par un point peuvent disparaître lors d'une copie. Si c'est le cas pour vous, voici le contenu de ces deux fichiers et leurs emplacements :

  Fichier ./.htaccess :

```
<IfModule mod_rewrite.c>
   RewriteEngine on
   RewriteRule    ^$    webroot/    [L]
   RewriteRule    (.*) webroot/$1    [L]
</IfModule>
```

Fichier ./webroot/.htaccess :

```
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteRule ^ index.php [L]
</IfModule>
```

- **Extension PHP "intl"**:

  Vérifier qu'elle est activée dans le php.ini:

  ```
  $ php -ini | grep intl
  ```

  Sinon, l'ajouter dans le php.ini:

  ```
  extension=intl.so
  ```

  Vérifier qu'elle est bien installée :

  ```
  $ php -m | grep intl
  ```

  (ça devrait retourner "intl")

  Sinon, l'installer:

  Ubuntu:

  ```
  $ sudo apt install php-intl
  ```

  (pour chercher un package, taper “apt search php-intl”)

  CentOS:

  ```
  $ sudo yum install php-intl
  ```

MacOS avec brew:
Installer l'extension
Les extensions sont recherchées dans /usr/local/Cellar/php72/7.2.0_11/lib/php/
$ brew install php72-intl
Fichier créé /usr/local/etc/php/7.2/conf.d/ext-intl.ini donne chemin de l'extension : /usr/local/opt/php72-intl/intl.so
(Pour info, le fichier php.ini est dans /usr/local/etc/php/7.2/php.ini, mais c'est inutile de le modifier)

MacOS avec XAMPP:
cf http://stackoverflow.com/questions/27886117/php-intl-installation-on-xampp
Il se pourrait que vous ayez besoin de ré-exécuter l'installeur de XAMPP afin de cocher l'option "XAMPP Developer Files", si vous ne l'aviez pas déjà fait lors de votre installation de XAMPP
cd /Applications/XAMPP/bin
sudo ./pecl install intl
(Vérifier qu'elle a bien été installée dans /Applications/XAMPP/xamppfiles/lib/php/extensions/no-debug-non-zts-20131226/)
Attention, cette méthode semble ne plus fonctionner avec XAMPP 7 (qui inclut php 7).
Sur Mac, j'ai dû installer l'extension intl via macport :
sudo port install php71-intl
Puis copie de cette extension dans le dossier de XAMPP :
sudo cp /opt/local/lib/php71/extensions/no-debug-non-zts-20160303/intl.so
XAMPP_716-0/xamppfiles/lib/php/extensions/no-debug-non-zts-20160303/

⇒ **Recharger la configuration du serveur Web, et recharger la page web pour voir si le problème est réglé**

- **Extension PHP "mbstring" et librairie "zlib"**

La collection de librairies est normalement déjà installée avec PHP en tant que dépendance.
Sinon, faire :
(Ubuntu) :

$ sudo apt-get install libapache2-mod-php5

⇒ **Recharger la configuration du serveur Web, et recharger la page web pour voir si le problème est réglé**

- **<u>Librairie PHP "php-gd"</u>**

  Sur Ubuntu :
  - pour php7.3 :
    $ sudo apt install php7.3-gd
  - pour php5 :
    $ sudo apt-get install php5-gd

  ⇒ **Recharger la configuration du serveur Web, et recharger la page web pour voir si le problème est réglé**

  Si ce n'est toujours pas le cas, aller dans le fichier /etc/php5/apache2/php.ini et vérifier que la ligne suivante est présente et décommentée (absence de ";" devant la ligne), sinon il faut la rajouter :

  extension=gd.so

  ⇒ **Recharger la configuration du serveur Web, et recharger la page web pour voir si le problème est réglé**

- **<u>KeepAlive</u>:**

  Editer le fichier de configuration du serveur web (**/etc/apache2/apache2.conf** sur Ubuntu, /etc/httpd/conf/http.conf sur CentOS/Fedora) et Mettre le paramêtre "KeepAlive" à "Off" :

  KeepAlive Off

- **<u>event et prefork modules:</u>**

Par défaut Apache utilise un "event module" et PHP un "prefork module".
Il faut désactiver le 1er module et activer le second :

```
$ sudo a2dismod mpm_event
$ sudo a2enmod mpm_prefork
```

7. Test de l’application

(updated 4/3/21 - EP)

Ces tests sont à exécuter après chaque modification (mais aussi après une nouvelle installation), et surtout avant tout “git commit”, pour éviter toute régression (vérifier que ce qui fonctionnait avant continue de fonctionner).

- **Verifier que la suite de tests passe complètement**

  Placez-vous à la racine du projet, exécutez la commande suivante (ça prend quelques minutes, et pendant ce temps là vous risquez de ne pas pouvoir consulter le site web à cause des dossiers tmp/cache/models et tmp/cache/persistent qui ne sont plus accessibles...)

  $ ./**TESTS.sh**
  (équivalent à 'vendor/bin/phpunit' ; si ça ne marche pas, essayez 'vendor/phpunit/phpunit/phpunit')
  (Php7 utilise Phpunit 6 alors que Php 5 utilise Phpunit PHPUnit 5.7)

  A la fin, vous devez voir quelque chose du genre :

  194 / 194 (100%)

  Time: 2.91 minutes, Memory: 58.00MB

  OK, but incomplete, skipped, or risky tests!

  Tests: 194, Assertions: 4557, Incomplete: 16.

- **Temps d'exécution constaté sur différentes plateforme**
  - Sur Mac OS 11.1 avec brew, sur un MacBook Pro (15-inch, 2017) 2.9GHz Core i7 (fait le 4/2/21)
    - Avec php7.2, les 194 tests s'exécutent en moins de 3 mn

    Anciens tests :

  - Sur une VM Ubuntu18 installée sur un MacBook Pro (15-inch, 2017) 2.9GHz Core i7 (fait le 21/1/20)
    - Avec php7.2, les 99 tests s'exécutent en un peu moins de 50 secondes
  - Sur Mac OS 10.13 avec brew (fait fin 2018) :
    - Avec php5.6, les 94 tests s'exécutent en 55 secondes
    - Avec php7.2, les 94 tests s'exécutent en 49 secondes
  - Sur Mac OS 10.12 avec XAMPP (fait en 2017) :
    - En php5.6, les 53 tests s'exécutent en 25-26 secondes
    - En php7.1, les 53 tests s'exécutent en 9-10 secondes, soit 2 à 3 fois plus vite !!! (vive Php 7)
  - Sur ScientificLinux (CentOS) 6.4 (serveur de test pweb) :
    - En php5.6, les 53 tests s'exécutent en 20 secondes
    - En php7.1, les 53 tests s'exécutent en 16-17 secondes, soit pas beaucoup plus rapide, bizarre...
  - Sur CentOS 6.4 (serveur de production), dans une VM :
    - avec php5.6 avec phpunit 5.7, les 53 tests s'exécutent en 47 secondes !!!
    - avec php7.1, les 53 tests s'exécutent en ??? (Php7, c'est pour très bientôt...)

- **Remarque : si jamais vous voulez n'exécuter que certains tests**

  Marquer le(s) test voulu avec une annotation :

```
/**
 * @group failing
 * Tests the api edit form
```

```
     */
    public function testEditAction()
```

Ensuite, exécuter avec :

```
$ phpunit --group failing
```

Attention, ne pas oublier d'enlever ces annotations avant de commiter avec git !

On peut attribuer plusieurs groupes à un test :

```
    /**
     * @group failing
     * @group bug2204
     */
    public function testSomethingElse()
```

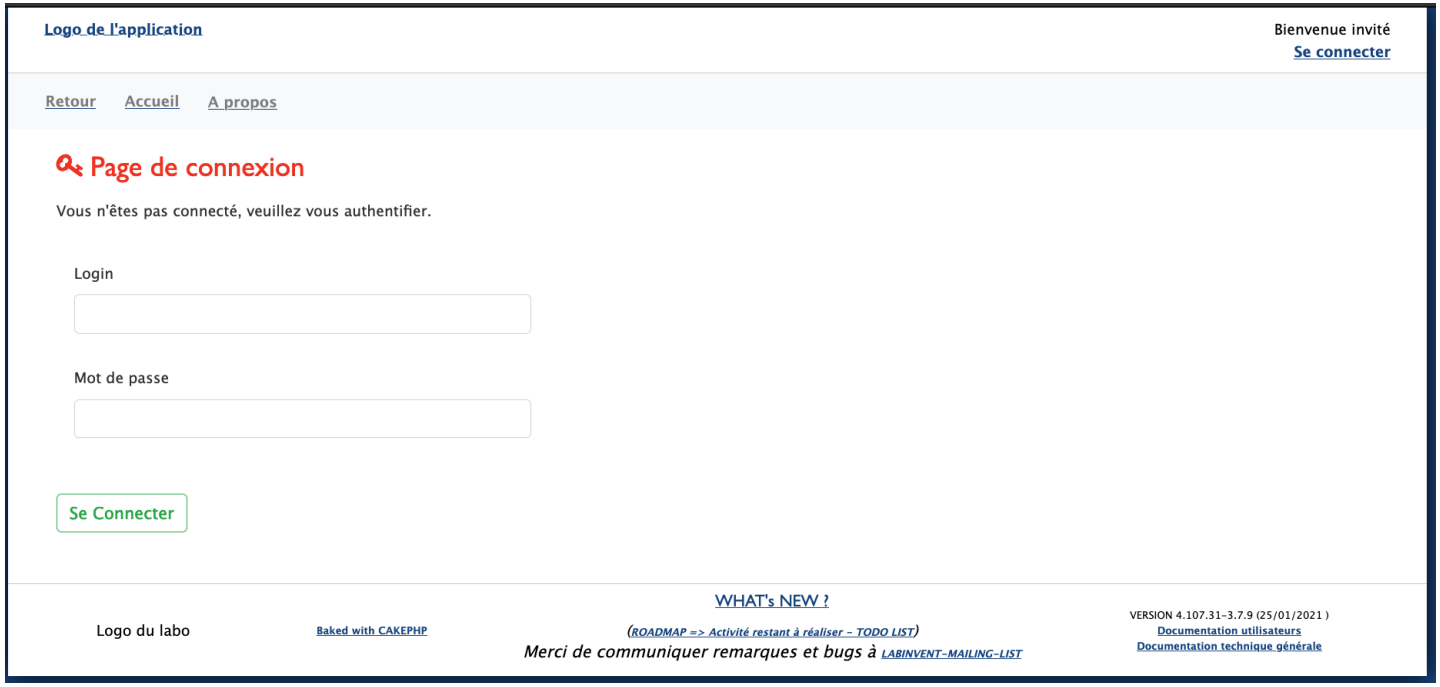
8. Fin de la phase installation, première connexion

(updated 03/02/2021 - EP)

Si vous n'y êtes pas déjà, allez sur la page d'accueil de LabInvent :

- Si installation via docker : http://localhost:8081
- Sinon, http://localhost/labinvent, ou bien http://localhost:8081, ou encore http://labinvent.test

Maintenant, sortez du mode "installation" en cliquant sur le bouton "Stopper le mode installation"
Si vous voulez revenir au mode "installation" plus tard, exécutez simplement ./PANIC_MODE (./PANIC_MODE macos sur Mac OS) puis revenez sur la page d'accueil. Vous arrivez alors sur la page de connexion suivante :



Une fois que vous aurez mis en place vos logos, vous aurez quelque chose de ce genre (attention, c'est une ancienne photo) :

Si vous êtes en train d'installer une **version de "production"**, alors il vaut mieux enlever le mode DEBUG général (mais vous pouvez le garder un peu encore, le temps de bien terminer la configuration).
Par contre, si vous installez une **version de "développement"**, alors vous pouvez laisser le logiciel en mode DEBUG général.
(le mode DEBUG général est différent de celui proposé dans le chapitre suivant sur la "configuration")

Pour désactiver (manuellement) le mode DEBUG général : allez à la racine du projet, et ouvrez votre fichier de configuration générale de labinvent (**config/app.php**).
Allez environ à la ligne 12 (au début du fichier):

```
'debug' => filter_var(env('DEBUG', true), FILTER_VALIDATE_BOOLEAN),
```

Et remplacez juste le **true** par **false**, comme ceci :

```
'debug' => filter_var(env('DEBUG', false), FILTER_VALIDATE_BOOLEAN),
```

Le logiciel est configuré par défaut en **mode “SANS LDAP”**, ce qui veut dire que les utilisateurs sont définis localement dans le logiciel, il n’y a pas de connexion LDAP. Vous pourrez changer ça plus tard si vous le souhaitez, dans la configuration (voir le lien vers la documentation technique ci-dessous).

Donc pour l’instant, **connectez-vous à l'aide du login "superadmin",** avec le mot de passe "**login**". Il vous permet de vous connecter en tant que "Super Administrateur" **pour pouvoir configurer l'application**.
Vérifiez que c'est bien le cas : sur la page d'accueil, vous devriez avoir une ligne qui affiche "Vous êtes connecté en tant que *SuperAdmin Stephane* (Super Administrateur)"

Vous avez aussi accès à 3 autres utilisateurs (tous avec le mot de passe “login”) qui vous permettront de tester tous les profils utilisateurs :

- “**gestion**” (Gestionnaire Geraldine) : il a le profil “**Administration**”
- “**resp**” (Responsable Robert) : il a le profil “**Responsable**”
- “**user**” (Utilisateur Ulysse) : il a le profil “**Utilisateur**” (profil de base, le moins privilégié)

~~*NB : Il y a toujours un utilisateur par défaut en mode "SANS LDAP", il s'agit de l'utilisateur "**_fake_ldap_user_**" avec le mot de passe "**_fake_ldap_user_pass**" (il a le profil minimum, c'est à dire "**Utilisateur**" ; il est utile notamment pour les tests, afin de tester le profil "Utilisateur" accordé par défaut à toute personne du ldap qui n'a pas de privilège particulier)*~~

⇒ **Pour savoir comment configurer ce logiciel et l’adapter à vos besoins, veuillez maintenant consulter la documentation technique. Vous pouvez aller directement à la partie configuration de cette documentation** (chapitre 9).

9. Mise à jour du logiciel (update)

(updated 01/02/2021 - EP)

Si vous venez juste d'installer le logiciel, vous pouvez passer directement au chapitre suivant.

Une fois le logiciel installé, il est très facile de le mettre à jour à la dernière version disponible.

Pour information, le fichier README-LABINVENT.md contient la liste des mises à jour.

1) Noter votre version actuelle

Voir sur la page d'accueil en bas à droite

2) Mettre à jour le code source (et la base de données si nécessaire)

Il suffit d'exécuter le script "./**UPDATE**" depuis la racine du projet :

```
$ cd LABINVENT/
$ chmod +x UPDATE
$ ./UPDATE
```

Vérifiez que la version du logiciel a bien changé (en bas à droite de la page d'accueil).

Ce script fait deux choses pour vous :

- il met à jour le code source du logiciel (avec un simple "git pull")
- il met à jour la base de données (seulement si nécessaire)

*NB: Ensuite, il est préférable de **vider le cache de votre navigateur** (idem pour tous les utilisateurs du logiciel) afin qu'il utilise bien la dernière version des pages web et des scripts javascript :*

- *pour cela, quand vous êtes devant une page web de labinvent, taper la combinaison de touches suivantes pour obliger votre navigateur à vider son cache : **Maj + Ctrl + R** (on peut aussi essayer Ctrl + F5 ou Maj + F5) (sur Mac, remplacer la touche Ctrl par ⌘)*
- *pour **effacer complètement le cache** (méthode plus radicale et donc plus efficace), on peut tenter la combinaison de touches `Ctrl` + `Maj` + `Suppr` (sur Mac, remplacer la touche Ctrl par ⌘)*

Seulement si nécessaire, vous pouvez aussi **mettre à jour MANUELLEMENT la base de données si vous le souhaitez. Attention toutefois,** ceci n'est utile que si la mise à jour AUTOMATIQUE (ci-dessus) a échoué.

Pour cela, aller dans le dossier **database/update**/

Exécuter TOUS les scripts qui sont d'une date postérieure à la date de votre version (notée à l'étape 1).

Attention, il faut les exécuter dans l'ordre anti-chronologique, un par un.

Pour cela, on utilise le script **db-update.sh** afin d'exécuter les scripts SQL qui sont dans le sous-dossier **script_sql**/ ; pour exécuter le script qui date de JJ-M-AAAA :

```
$ ./db-update.sh script_sql/db-update-AAAA-MM-JJ.sql
```

Exemple (toujours dans l'ordre anti-chronologique) :

```
$ ./db-update.sh script_sql/db-update-2020-11-04.sql
$ ./db-update.sh script_sql/db-update-2020-10-23.sql
$ ./db-update.sh script_sql/db-update-2020-10-22.sql
$ ./db-update.sh script_sql/db-update-2020-10-21.sql
...
```

3) Tester que tout fonctionne bien

Aller à la racine du projet.

```
$ ./TESTS.sh
```

Si jamais les tests ne veulent pas s'exécuter, essayer ceci:

```
$ cd install/
$ ./plugins_update.sh
(équivalent à faire "php composer.phar update", ce qui met à jour les plugins, et notamment CakePhp)
(et si ça ne marche toujours pas, on peut aussi essayer ./plugins_install.sh)
$ ./TESTS.sh
```

Tous les tests doivent passer

NB: Si phpunit ne se met pas à jour, ne pas hésiter à supprimer tout le dossier vendor/phpunit/ avant de lancer la commande ./plugins_update.sh