LABINVENT - INSTALLATION

URL officielle de ce doc : https://tinyurl.com/labinvent-install
Auteurs: E. Pallier, E. Bourrec
Version: 30/11/2020
⇒ 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

- 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

COPYRIGHT (C) 2012-2021 IRAP (Institut de Recherche en Astrophysique et Planetologie) Toulouse - France

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. Elle doit toutefois être partagée et ré-injectée dans le logiciel, afin que toute la communauté des utilisateurs puisse en profiter, et afin que le logiciel LabInvent reste une entité unique et bien définie.

2. Version de démonstration

Si vous désirez juste voir à quoi ressemble le logiciel LabInvent, nous mettons à votre disposition un site de démo.

Notamment, en s'y connectant avec le login "utilisateur", on peut voir le workflow par défaut qui s'affiche sur la page d'accueil (pour tout utilisateur lambda, non privilégié).

Logins disponibles (sans mdp), correspondant aux 3 profils principaux (excepté le profil "superadmin"):

"utilisateur", "responsable", et "admin"

⇒ http://planetoweb2.cesr.fr/labinvent2

3. A quel niveau intervenir selon les besoins (où faire quoi ?)

\oufairekoi

Cette section répond au besoin de **savoir à quel niveau on doit intervenir selon la modification qu'on souhaite réaliser**.

Elle peut aussi être une réponse pour les **personnes qui souhaitent contribuer** au logiciel mais ne savent pas où agir.

Il y a 6 niveaux principaux d'intervention selon ce qu'on souhaite faire :

- **DOC** : on souhaite améliorer l'information sur le logiciel => il suffit pour cela de compléter cette présente documentation
- **INSTALL** : on souhaite améliorer la phase d'installation du logiciel => il faut agir sur le script **install/install.sh** (on pourrait aussi traduire ce script en php pour une compatibilité multi-plateformes) => voir le chapitre "installation du logiciel"
- **TESTS** : on souhaite renforcer la qualité du logiciel => il faut pour cela renforcer la suite de tests déjà existante (voir le chapitre sur les tests)
- **PERF** : on souhaite accélérer l'application => on peut paralléliser certaines tâches telles que l'envoi d'emails, la génération des pdfs, l'exportation <des listes de matériels, les traitements automatiques tels que la mise en cache du LDAP ou le processus de relance par mail pour les suivis…
- **AUTORISATIONS** : on souhaite modifier les droits des différents profils associés aux utilisateurs (qui a le droit de faire quoi) => voir le chapitre sur le workflow
- **CODE** : on souhaite modifier l'apparence ou le comportement du logiciel => il faut pour cela agir sur le code source

Nous allons maintenant détailler le dernier niveau d'intervention ci-dessus qui concerne la **modification du code source**. La modification du code source **se subdivise elle-même en 3 niveaux** selon ce qu'on désire modifier. Ces 3 niveaux correspondent à ceux du **pattern "MVC" (Modèle/Vue/Controleur)** sur lequel le framework cakephp est basé. En modifiant le logiciel, on agira :

- soit sur l'aspect (présentation) : qui peut Voir quoi => niveau (V)ue
- soit sur le comportement : qui peut Faire quoi => niveau (C)ontrolêur
- soit sur les entités et leurs relations : de quoi parle le logiciel => niveau (M)odèle

3.1. Modification de l'ASPECT du logiciel => Niveau "Vue" du MVC

En agissant à ce niveau, on modifiera **l'apparence** du logiciel, sa **présentation**.

C'est le niveau "le plus intéressant" car toute modification à ce niveau est immédiatement visible, c'est donc plus gratifiant que les 2 autres niveaux suivants.

A ce niveau, on définit à la fois :

- la structure, la présentation, et le contenu des différentes pages web de l'application, en gros "où on met quoi et comment",
- et aussi "qui peut Voir quoi" (le **qpVq**)

Exemples :

- tout le monde peut voir qui a créé ou modifié une fiche matériel => actuellement ça n'est pas le cas, ça n'est visible qu'à un profil "superadmin", mais on pourrait changer ça à ce niveau
- faire en sorte que l'application soit visible sur un petit écran (smartphone) => ceci a été fait début 2020
- paginer la liste des matériels
- afficher le QrCode en haut à droite de toutes les fiches matériel
- quand on visualise une fiche matériel, on doit aussi voir la liste des ses suivis et des ses emprunts
- … etc

Pour intervenir à ce niveau, les compétences techniques suivantes sont nécessaires :

- **Php** (niveau assez basique) : génération dynamique du code html, et insertion des données dans les pages, en tenant compte du profil de la personne connectée, de la configuration du logiciel, de l'état des données présentes
- **Html** : structure et organisation des pages
- **Css** : aspect des pages
- **Javascript** : comportement dynamique des pages

3.2. Modification du COMPORTEMENT du logiciel (règles métier) => Niveau "Contrôleur" du MVC

En agissant à ce niveau, on définit **ce qui est possible et par qui**, en gros "qui peut Faire quoi" (le **qpFq**).

En simplifiant à outrance, on pourrait dire qu'on définit ce qui est "read only" et pour qui.

Dans le détail, on définit à ce niveau à la fois :

- les **ACTIONS** possibles sur les divers "objets" (tables de la BD) de l'application,
- et **QUI** a le droit de les exécuter.

L'ensemble des actions et des droits qui sont associés constituent ce qu'on appelle "les **règles métier**" du domaine étudié, en l'occurence ici un "inventaire des matériels".

Il existe **4 actions fondamentales** sur les différents "objets" (tables de la BD) de l'application, les actions nommées **CRUD pour Create, Read, Update, et Delete**. A ces actions fondamentales, on peut en ajouter d'autres plus spécifiques qui correspondent au domaine métier de l'application. Par exemple, pour gérer le statut d'un matériel, on a besoin de modifier son statut en le validant, ou encore en l'archivant, ce qui correspond à 2 nouvelles actions "valider" et "archiver", qui sont vraiment spécifiques à un inventaire matériel.

Exemples d'actions et droits associés :

- tout le monde peut **créer** une nouvelle fiche matériel
- tout le monde peut **modifier** tous les matériels => actuellement, ce n'est pas le cas, seul l'utilisateur d'un matériel peut modifier sa fiche, ou alors il faut avoir un profil supérieur de type "admin" ou "superadmin"
- le profil "admin" peut **modifier** un matériel validé
- on ne peut pas **supprimer** la fiche d'un matériel qui est validé
- ...

Tous ces exemples représentent des **règles métier**.

La couche “contrôleur” fait l’intermédiaire entre les 2 autres couches (modèle et vue). Elle a pour rôle de répondre aux requêtes de l’utilisateur en récupérant les données nécessaires via la couche “Modèle”, et en les organisant avant de les fournir à la couche “Vue” pour qu’elles soient présentées à l’utilisateur.

A ce niveau les compétences techniques requises sont :

- **Php** (plus haut niveau, orienté objet)

3.3. Modification des données (entités) et de leurs relations => Niveau “Modèle” du MVC

A ce niveau, on définit les **entités** (objets, tables de la BD) du domaine étudié (ici un “inventaire”) et leurs **relations**. En quelque sorte on “modélise” le domaine étudié. Par exemple, un inventaire est constitué des **entités** “matériel”, “utilisateur”, “emprunts”, “suivis”, “qrcode”, etc. Dans un tel “modèle”, on doit créer les **relations** entre les entités qui reflètent le fait qu’un matériel “appartient” à un “utilisateur”, qu’il a été “créé” par tel autre, que ce matériel est de telle catégorie, qu’un utilisateur a tel profil, que ce matériel a été emprunté par tel ou tel utilisateur, qu’il est suivi pour une intervention technique, etc…

C’est aussi à ce niveau qu’on définit les “**règles de validation**” des diverses entités du domaine, c’est à dire tout ce qui doit être vérifié pour dire qu’une entité a été correctement saisie par l’utilisateur. Seule une entité validée peut être sauvegardée dans la BD.

Exemples :

- ajouter un attribut à l’entité “matériel” : couleur, dimension, poids, adresse Mac ou IP, etc.
- ajouter une nouvelle “entité”, par exemple la notion de “type de matériel”
- ajouter la gestion des entités abstraites telles que les logiciels, les licences…
- ajouter une relation de composition entre matériels permettant de dire qu’un matériel est composé de tel et tel autre, qui sont ses composants

A ce niveau les compétences techniques requises sont :

- **Php** (plus haut niveau, orienté objet)
- **Modèle relationnel**
- **Requêtes Sql**

C’est sans doute cette couche qui est la plus sensible et dans laquelle toute modification doit être faite avec une extrême prudence.

4. Visite guidée rapide

- Version mobile-ready (réduire l'écran => menu s'adapte)
- Impression du QrCode sur l'étiquette => on peut flasher avec un mobile
- Accueil => changements
- Accueil => Stats
- A propos => Doc + guide user
- Outils => Autorisations en cours
    - rouge = restriction
    - vert = accès autorisé
- [Outils => Notifications en cours]
- Liste matériels :
    - par défaut : récents (- 5ans)
    - les divers filtres + filtrer : "dell moniteur"
    - On peut trier par colonne
    - Export :
        - la liste en cours (filtrage actif) OU encore tout
- Liste utilisateurs (Outils)
    - Lien vers Annuaires labo
    - tous OU seulement les responsables
    - trier par colonne
- Autres listes : groupes, catégories, fournisseurs…
    - Groupes métiers :
        - Lien vers page web labo
        - GACL => description plus complète :
            - Utilisateurs associés (avec mention "responsable")
            - Matériels associés
    - Groupes thématiques : même principe que groupes métiers
        - PIME => plutôt un service, mais ici en attendant :
            - Utilisateurs et matos associés
            - PROJETS (cf GAHEC)
    - Projets
        - Liens vers pages web projets du labo
        - projet "Achats labos" :
            - Matos associés

5. Astuces (HOWTO) d'utilisation du logiciel

(E. Pallier 19/10/2020)

5.1. Pour TOUS

- Voir Mes matériels
- Comment retrouver un matériel (filtre ou find)
- Voir les stats d'utilisation
- Connaître les nouveautés
- Exporter la liste de mes matériels
- Prêter ou Emprunter un matos
- Déclarer une maintenance sur un matos
- Demander à sortir un matos de l'inventaire
- Filtrer la liste des matos selon plusieurs critères
- Commander un matos
- Annuler une commande
- Comment facilement retrouver un matos à partir de son étiquette

5.2. Pour les Administratifs (Gestion)

- (adm) Voir matériels à commander
- (adm) Voir matériels à archiver
- (adm) Voir matériels "zombies" => créés mais non commandés depuis longtemps
- Annuler la validation (livraison) d'un matos
- Comment remplacer un fournisseur par un autre

6. Installations existantes (liste des labos)

(EP updated 29/9/20)

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

Depuis	**Laboratoire**	**Nom donné au logiciel** (et aux groupes thématiques/métiers)	**Versions et dates**		**LDAP**	**Etiqueteuse**
			LabInvent	**Composants**		

2012	**IRAP version développement sur Mac OS**	**InventIrap**	Dernière version (branche dev)	**Mysql** 15.1 Distrib 10.3.12-MariaDB **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	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**	**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	oui même que IRAP, mais peu utilisée (utilisent plutot une dymo manuelle)

	(= 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 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)	**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)	**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	oui
2021 ? (TODO)	**IP2I (Institut Physique 2 Infinis) (IPNL et LMA, 250 pers)** Lionel Capoani					

7. Installation des pré-requis

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

7.1. 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

- **MAC OS X :**
  - (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

- **WINDOWS 10** (TODO)

7.2. Installation (pré-requis)

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

- *un serveur web **Apache** (récent)*
- *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 ce tiercé est déjà présent 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 ces 3 éléments (il n'y aura rien d'autre à faire ensuite).

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

7.2.1. Installation sur Mac OS (avec HomeBrew)

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

7.2.2. Installation sur Windows 10

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

7.2.3. Installation sur Linux

7.2.3.1. Exemple pour une distribution UBuntu 18.04.3 LTS (janvier 2020)

(fait par stagiaire Jeanne Prugniel, et aussi par E. Pallier)

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 pour 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

7.2.3.2. 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...

7.2.3.3. 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

7.2.3.4. 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

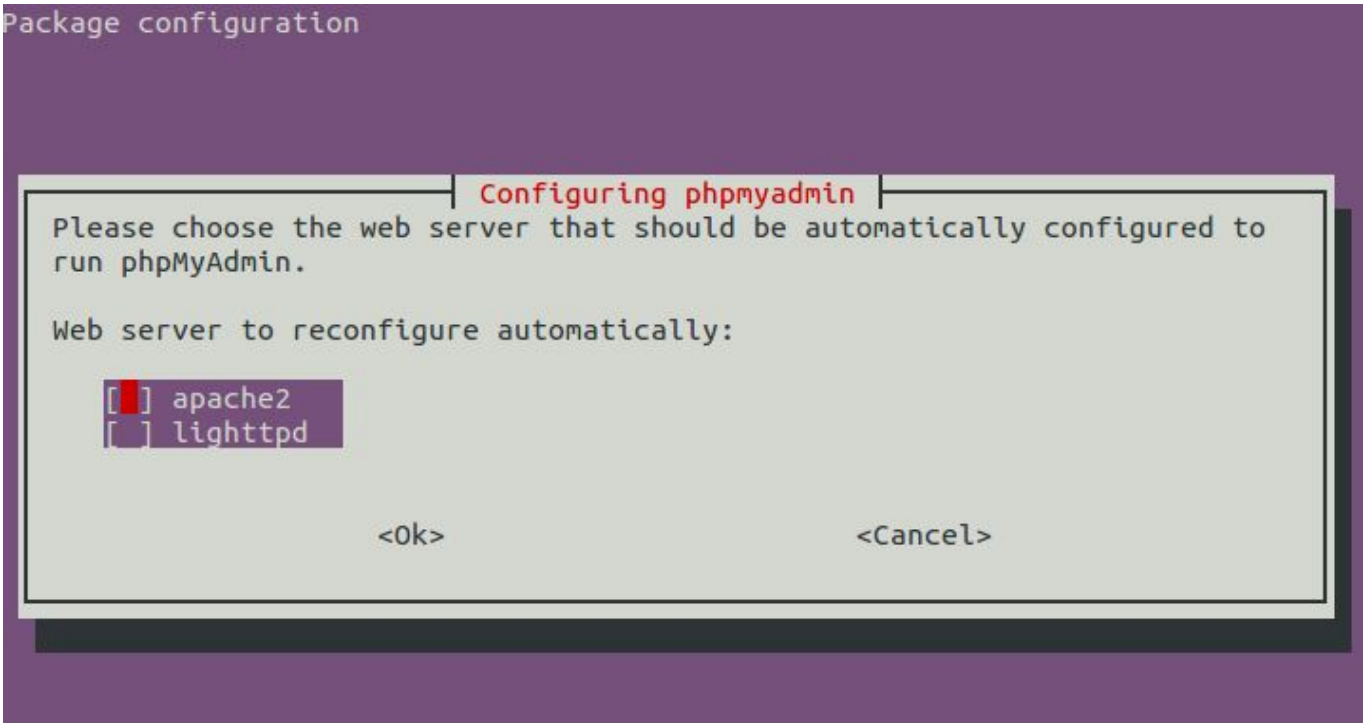
Installer un serveur web (Apache) :
$ sudo apt-get install apache2

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

7.2.3.5. 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

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

7.3.1. 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, …)
- ...

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)

7.3.2. Configuration de MySql

A priori rien à faire...

7.4. (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.

8. 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

8.1. 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

8.1.1. (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).

8.1.2. (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
- 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.

8.2. Installation du logiciel

Il suffit d’exécuter le script installation.sh depuis le dossier install/ :

```
$ cd install/
$ ./intallation.sh
```

Attention: sur **Mac OS X**, utiliser installation-macos.sh au lieu de installation.sh

(vous pourriez éventuellement le faire en tant qu'administrateur "root", mais ça n'est pas nécessaire, le script fera quelques petits "sudo" seulement au besoin).

(Conseil : à la plupart des questions, laissez les réponses par défaut)

8.3. Contrôles rapides

- **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 :

  ```
  $ 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.

8.4. 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/	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.
tests/	Le dossier contenant tous les tests exécutés par le script TESTS.sh (voir ci-dessus)
tmp/	Dossier utilisé par LabInvent comme cache des pages web et des tables de la BD
vendor/	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...).
webroot/	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). 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.
Model/ - Behavior/ - **Entity**/ - **Table**/	**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...).
Template/ (et View/)	**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.

9. Mise à jour du logiciel (update)

(updated 14/01/19 - 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) n'a pas été suffisante.

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 chronologique**, un par un

Exemple:

$ ./db-update-2016-06-03.sh
$ ./db-update-2016-06-06.sh
$ ./db-update-2016-06-07.sh

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

10. Démarrage (re-) des services nécessaires

(EP 13/10/2020)

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)
- Le Serveur de Gestion de Base de Données : MySql (ou MariaDB)
- Le Serveur Web : Apache (bien configuré avec Php)

<u>Voir les services qui sont actuellement actifs</u> :

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

- (Linux) : TODO
- (MacOs) :

```
$ brew services list
httpd   stopped
mariadb  stopped
php     stopped
php@5.6  stopped
php@7.1  stopped
php@7.2  stopped
```

<u>Vérification de php</u> :

Version :

$ **php -v**

Si, on veut changer de version de php :

(MacOS) : $ **sphp <version>**

(par exemple : sphp 7.4)

Démarrage (pas nécessaire) :

To have launchd start php now and restart at login:

$ brew services start php

Or, if you don't want/need a background service you can just run:

$ php-fpm

Infos :

```
$ 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
```

<u>(re-)**Démarrage de MySql/MariaDB**</u> :

- (Linux) : TODO
- (MacOS) : mysql ne semble pas fonctionner, on installe plutôt MariaDB

Démarrage ponctuel :

$ **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

SUCCESS!

Démarrage et lancement automatique au boot par launchd :

$ brew services start mariadb

Version :

$ mysql -V

=> mysql Ver 15.1 Distrib 10.5.6-MariaDB

Connexion au serveur en local :

$ **mysql -uroot**

Remarques suite à l'installation :

A "/etc/my.cnf" from another install may interfere with a Homebrew-built server starting up correctly.

Infos :

$ brew info mariadb
=> mariadb: stable 10.5.6
=> /usr/local/Cellar/mariadb/10.5.6

(re-)**Démarrage de Apache (httpd)** :

- (Linux) : TODO
- (MacOS) :

Démarrage :

$ sudo apachectl (re)start
*(**stop** pour arrêter le service)*
*(aussi possible : **sudo apachectl -k restart**)*

Version :

$ **httpd -v**
Server version: Apache/2.4.46

Infos :

$ brew info httpd
=> httpd: stable 2.4.46
=> /usr/local/Cellar/httpd/2.4.46
=> DocumentRoot is /usr/local/var/www

11. Accès local à l’application web

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 de ces 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       labinvent.test
```

Puis créer un hôte virtuel de ce genre :

```
<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

Passez maintenant au chapitre suivant pour exploiter cette configuration.

12. Vérification de la conformité du site web LabInvent

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).

Tester l'accès web à l'application :

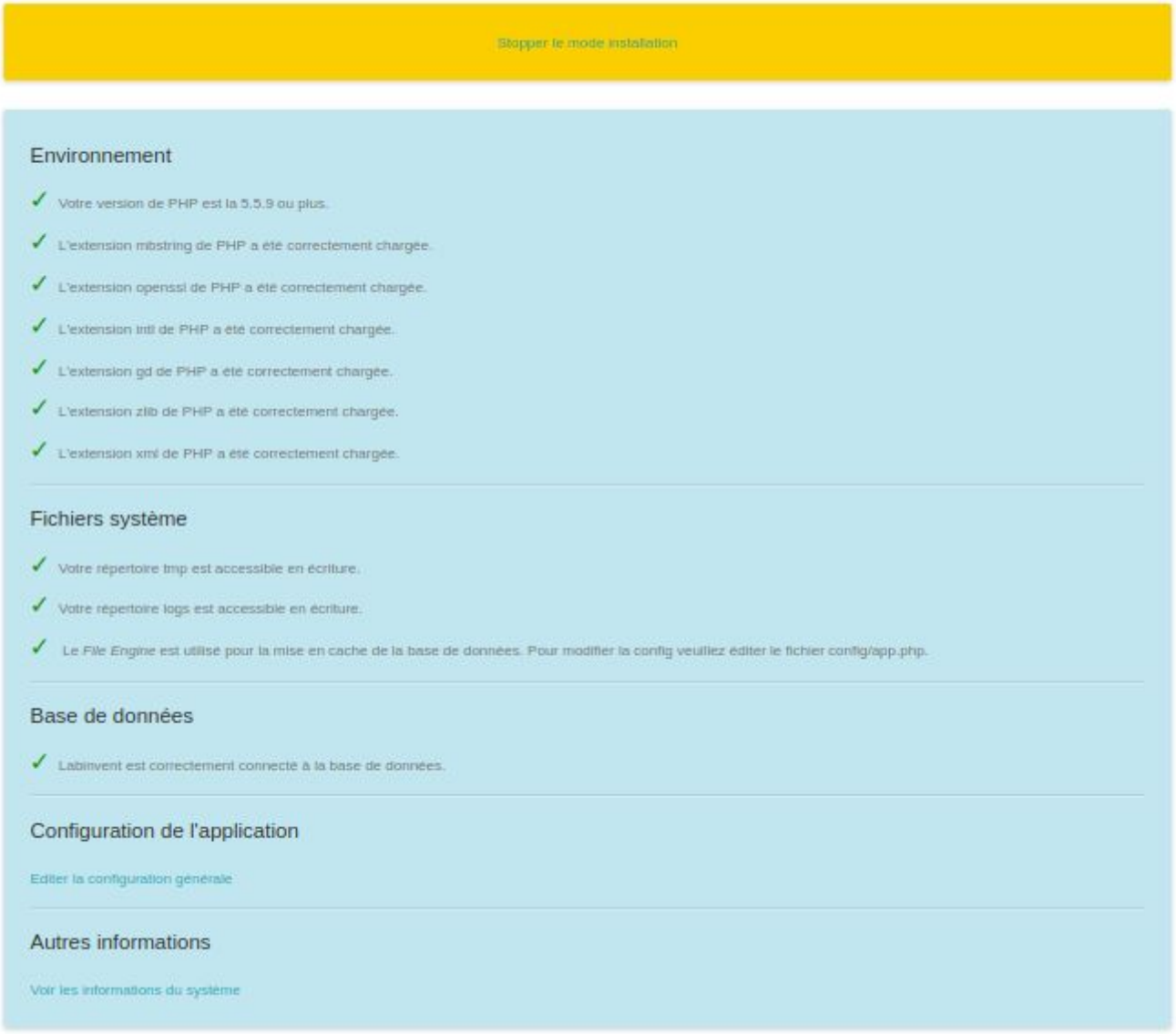
- http://localhost/labinvent (cas 1)

- 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

- **<u>URL-Rewriting (autoriser les fichiers .htaccess)</u>**

  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

- **Librairie PHP "php-gd"**

  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

- **KeepAlive:**

  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

- **event et prefork modules:**

  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

13. Test de l’application

(updated 24/01/20 - 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 :

  $ ./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 :

  Time: 28.5 seconds, Memory: 32.00MB

  OK (99 tests, 515 assertions)

- **Temps d’exécution constaté sur différentes plateforme**
  - 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 10.14 avec brew, sur un MacBook Pro (15-inch, 2017) 2.9GHz Core i7 (fait le 10/3/20)
    - Avec php7.2, les 99 tests s'exécutent en 36 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()
```

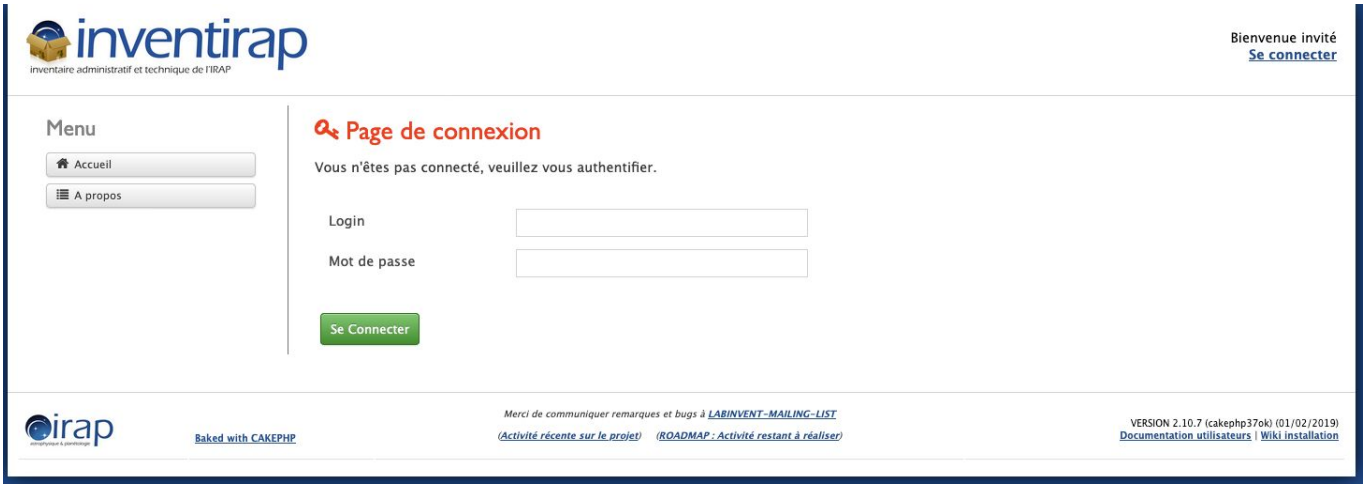
14. Fin de la phase installation, première connexion

(updated 22/01/20 - EP)

Si vous n'y êtes pas déjà, allez sur la page d'accueil de LabInvent : 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, allez dans le dossier database/ et exécutez simplement ./mode_panique.sh (./mode_panique_macos.sh sur Mac OS) puis revenez sur la page d'accueil.

Vous arrivez alors sur la page de connexion suivante :



Si vous êtes en train d'installer une **version de "production"**, alors il vaut mieux enlever le mode DEBUG général.
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, au chapitre suivant.

Donc pour l’instant, **connectez-vous à l'aide du login utilisateur créé lors de l'installation (par défaut "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 *Nom Prénom* avec le niveau d'authentification Super Administrateur."

NB1 : Les autres comptes utilisateurs que le script d'installation aura éventuellement créés (si demandé lors de l’installation) ont aussi le même mot de passe par défaut : "login"
NB2 : 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.