ENS-Lyon <elodie [dot] goy [at] ens-lyon.fr>

<elodie [dot] goy [at] ens-lyon.fr>

Table of Contents

Introduction

Welcome on the "Chimithèque" project Web site, a chemical products management application developed by the ENS-Lyon.

This documentation is written in Docbook in French and English. We have used Open Source applications and operating systems such as Debian, Linux Mint, Inkscape and VLC for fun ! :)

Why Chimithèque

We needed a global method to manage chemical products of the different departments and laboratories of the ENS to:

- improve the security with a precise global listing of the chemicals products stored in the entire school
- reduce waste by encouraging chemical products managers to search in Chimithèque if a product can be borrowed from another department before ordering a new one

Concepts

"Chimithèque" uses the following notions:

- organization (typically the school, the université)
- entity (a department, a laboratory...): the organization has one or several entities
- store location (a fridge, a shelve...): an entity has one or several store locations
- user (a teacher, a student...): a user belongs to one or more entities and has rights in the application
- product card: id card of the product (contains the CAS number, names, chemical formulas, risk phrases...), the product card is entered one and only one time in the application by a user with the required rights. Product cards are visible by every user. They can be modified but modifications are saved.
- storage card: association between a product card an a store location. To store a product in one of his entities, a user must enter a storage card containing informations such has the volume or weight to store, the store location, the number of items and of course the product name. Like product cards, storage cards can be modified and changes are saved.

What Chimithèque is

- an application to access the safety informations of your chemical products
- an application to manage the stocks of your chemical products
- an application to search chemical products with different criteria

What Chimithèque is not

- a database of all existing chemical products in the world: the application comes with X product cards (for X look at the main site) and these cards can be shared thank to an import/export functionality
- an application immediately pluggable to devices such as bar-code scanners or applications

Technical aspects

"Chimithèque" is developed in *Python* with the web2py framework. It runs on Linux but should be installable under Windows (not covered by this documentation). A *PostGreSQL/MySQL/SQLite* database (other databases possible but not tested) and an *SMTP* server are needed. The application is available in English and in French but can be translated into any other language.

Known limitations

- it is currently impossible to create hierarchical entities. This feature would require to rethink the permissions policy.

Documentation

Installation documentation

This documentation deals with the installation under Linux (Debian latest stable) with Apache2.

Note

Feel free to send us feedbacks of installations under other platforms such as CentOS.

Chimithèque versions

In the download page you will find different files:

- `chimitheque_{version}_{date}.tar.gz` files. These files are the different versions of Chimithèque.
- a `chimitheque_stable.tar.gz` file, that is just a symbolic link to the last stable version.
- a `chimitheque_testing.tar.gz` file. This file is the most recent version of Chimithèque, currently tested at the ENS.

Prerequisites

- Linux server with 1GB of hard disk space and 1VCPU (for virtualized systems)
- *apache2-mpm-event* and *libapache2-mod-wsgi*
  PostGreSQL/MySQL database server, PostGreSQL version 8.4 or more, MySQL version 5 or higher
- *SMTP* server
- *Python 2.7* or more, *python-beaker*, *python-ldap*, *python-levenshtein*, *python-ldap (optionnal)*, *python-psycopg2* (for a PostgreSQL database, or the required database library for other types of databases),

> **Note**
> "Chimithèque" should run with other databases such as Oracle, MSSQL, FireBird, DB2, Informix, Ingres.
> The application has been successfully tested with PostgreSQL, MySQL and SQLite.

Directories organization

The installation is divided into:

- a source directory: `/usr/local/src/chimitheque_src` for example
- a deployment directory: `/var/www/chimitheque` for example

Installation steps

- dependencies installation
- "Chimithèque" installation
- PostGreSQL server installation (optionnal if you already have a databases server)
- Apache2 installation

Step 1: dependencies installation

You are supposed to be logged in `root`. `python 2.7` is installed by default.

```
bash$  aptitude install python-psycopg2 python-ldap python-beaker python-levenshtein unzip rsync
```

Step 2: application installation

Step A

Get the "Chimithèque" package in the download section.

```
bash$  cd /usr/local/src
bash$  wget http://chimitheque.ens-lyon.fr/download/chimitheque_stable.tar.gz
bash$  tar zxvf chimitheque_stable.tar.gz
bash$  mv web2py chimitheque_src
```

Step B

Choose an instance name - example: `prod`. Copy of the `chimitheque_sample.properties` file into `chimitheque_[instance].properties`.

```
bash$  cd /usr/local/src/chimitheque_src/applications/chimitheque
bash$  cp chimitheque_sample.properties chimitheque_prod.properties
```

Step C

Edit the `chimitheque_[instance].properties` file. It is documented.

Creation of the deployment directory specified in the `CHIMITHEQUE_PATH` variable.

```
bash$  mkdir -p /var/www/chimitheque
```

Step D

Deployment with the `deploy [instance]` command.

```
bash$  cd /usr/local/src/chimitheque_src/applications/chimitheque
bash$  ./chimitheque deploy prod
```

Step 3: PostGreSQL server installation

Step A

Packages installation. Is is the 8.4 version.

```
bash$  aptitude install postgresql postgresql-client
```

Step B

Serveur configuration. We do NOT use a dedicated cluster to make the installation easier.

Choose a database user name. Must be the same as the `SKEL_DBUSERNAME` variable. We will use the name `chimitheque_user`.

Append at the end of the `/etc/postgresql/8.4/main/pg_hba.conf` file:

```
host    all             chimitheque_user     127.0.0.1/32            md5
```

> **Tip**
> We allow the user `chimitheque_user` to connect to the database from `localhost` with a password (`md5`).

Restarting the server:

```
bash$  /etc/init.d/postgresql restart
```

Step C

`chimitheque` database initialization.

```
bash$  su - postgres
postgres$  psql -p 5432
postgres$  CREATE USER chimitheque_user WITH PASSWORD 'chimitheque_password' SUPERUSER;
postgres$  CREATE DATABASE chimitheque OWNER chimitheque_user;
postgres$  GRANT ALL PRIVILEGES ON DATABASE chimitheque TO chimitheque_user;
```

- line 3: `chimitheque_user` = `SKEL_DBUSERNAME`
- line 3: `chimitheque_password` = `SKEL_DBPASSWORD`
- line 4: `chimitheque` = `SKEL_DBNAME`

Step 4-b: Apache2 installation

Installing the package:

```
bash$  aptitude install apache2-mpm-event libapache2-mod-wsgi
```

> **Warning**
> Do NOT use "apache2-mpm-worker", you could encounter performances issues.

Activating the modules:

```
bash$  a2enmod ssl  # only for HTTPS use
bash$  a2enmod wsgi
bash$  a2enmod deflate
bash$  a2enmod expires
```

Creating a new site (please look at the comments in the file):

```
bash$  vim /etc/apache2/site-available/chimitheque
```

```
<VirtualHost *:443>

# change your servername here
ServerName chimitheque.ens-lyon.fr

# remove the following line if you do not use SSL and change the 443 port in the virtualhost to 80
SSLEngine on
SSLCertificateFile /etc/ssl/certs/chimitheque.ens-lyon.fr.pem
SSLCertificateKeyFile /etc/ssl/private/chimitheque.ens-lyon.fr.key

WSGIDaemonProcess web2py user=www-data group=www-data \
display-name=%{GROUP} inactivity-timeout=120 maximum-requests=500 processes=5 threads=1

WSGIProcessGroup web2py

SetOutputFilter DEFLATE
AddOutputFilterByType DEFLATE text/html text/plain text/xhtml text/css text/javascript application/x-javascript

ExpiresActive On

# 1 month
```

```
ExpiresByType image/ico A2592000
ExpiresByType image/png A2592000

# 5 days
ExpiresByType text/javascript A432000
ExpiresByType text/css A4320000
ExpiresByType application/x-javascript A432000

# change /var/www/chimitheque by your CHIMITHEQUE_PATH variable
WSGIScriptAlias / /var/www/chimitheque/wsgihandler.py

# change /var/www/chimitheque by your CHIMITHEQUE_PATH variable
<Directory /var/www/chimitheque>
AllowOverride None
Order Allow,Deny
Deny from all
<Files wsgihandler.py>
Allow from all
</Files>
</Directory>

# change /var/www/chimitheque by your CHIMITHEQUE_PATH variable
AliasMatch ^/([^/]+)/static/(.*) \
/var/www/chimitheque/applications/$1/static/$2

# change /var/www/chimitheque by your CHIMITHEQUE_PATH variable
<Directory /var/www/chimitheque/applications/*/static/>
ExpiresDefault "access plus 7 days"
Order Allow,Deny
Allow from all
</Directory>

<LocationMatch ^/([^/]+)/appadmin>
Deny from all
</LocationMatch>

CustomLog /var/log/apache2/access.log common
ErrorLog /var/log/apache2/error.log

</VirtualHost>
```

Activating the site:

```
bash$  a2ensite chimitheque
```

Step 5: Starting the services

Apache2:

```
bash$  /etc/init.d/apache2 restart
```

You can also test without Apache, the application will then be reacheable at `http(s)://urldechimitheque:8000/chimitheque`:

```
bash$  /var/www/chimitheque/web2py.py -a a_password -i 0.0.0.0
```

`/var/www/chimitheque` is your `CHIMITHEQUE_PATH`.

Step 6: First connection

Go to the address: `http(s)://urldechimitheque/chimitheque`

Use the following login/password:

```
admin@admin.fr
admin
```

Then click on the `password` link (top of the screen) to change the admin password.

Step 7: Importing the product database and creating the first user

Go to the `tools > system tools > manage database` menu and `import product database`. Download the product database file in the download area and upload it. When the upload process is finished the number of product cards is updated (on the top right corner).

You now have to create a first user with the administrative privileges. This user will then create other users and entities. It is often a chemistry engineer or a person in charge of chemical products management in your organization. Go to the `tools > manage users/store locations/entities` menu and click on `create user`. Check the `is admin` check box, and fill in the required fields. An email is sent to the new user once the form is submitted so that he can initialize is password.

0.2 to 0.3 migration

A database metadata conversion is required to migrate from the 0.2 to the 0.3 version.

Install the new version as specified in the step2 but before the `./chimitheque deploy prod` command run the `./chimitheque -c migrate-to-03 -i prod -o no-check-dependencies` command.

Updating Chimithèque

- before the 0.3 version, repeat the procedure described at the step 2 with the new package.
- since the 0.3 version use the command `./chimitheque -c update -i prod -o update-stable` and then `./chimitheque -c deploy -i prod`.

In case of problem

Check that the `/var/www/chimitheque` directory belongs to `www-data:www-data` (Apache2 installation) or `uwsgi:uwsgi` (Nginx + Uwsgi installation).

Check that *Apache2 mod-python* is *not* installed (only for Apache2 installation).

Check that your database server is reachable with the command `telnet db_base_ip db_base_port`.

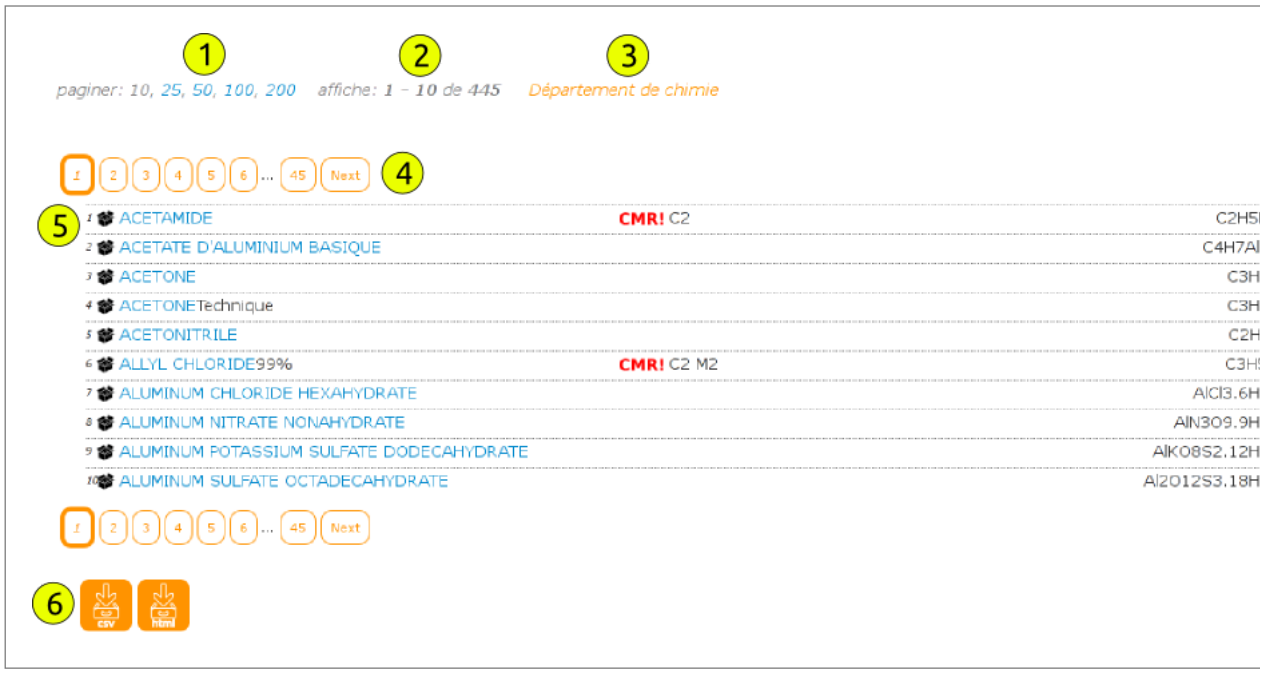
Check that your database user in your `SKEL_DBCONNECTION` string can connect to the database with its password.

User guide

Overall presentation

1. User profile management: change password and logout
2. Connected users, system console, number of product cards in the application
3. Advanced search
4. Application language
5. Main menu (items displayed according to privileges)

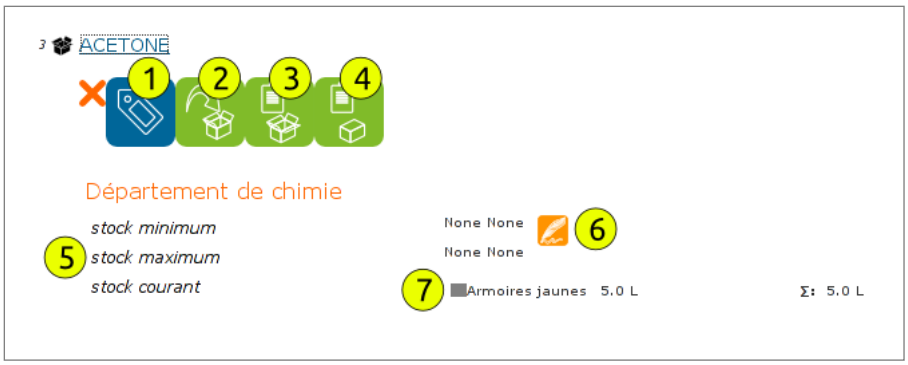
Search results



1. Number of results per page
2. Current page, number of results

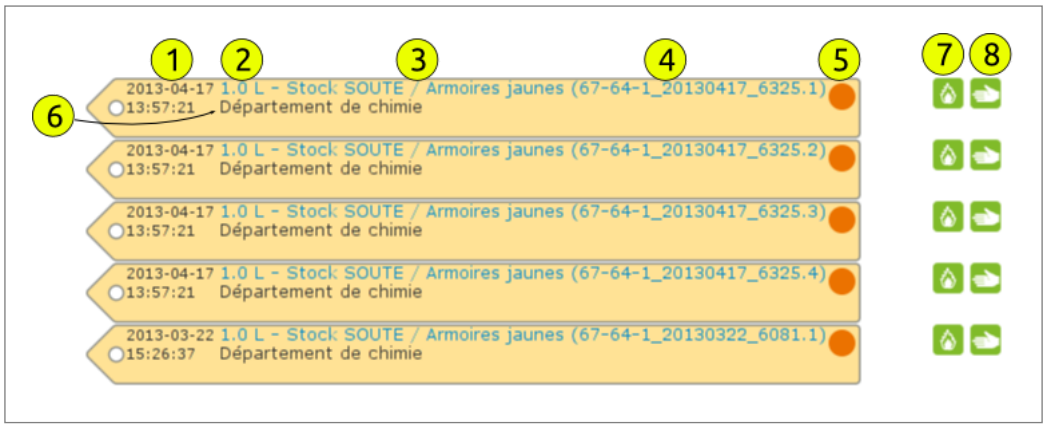
3. Current request
4. Paginator
5. Results in the following form:
    - line number
    - or icon: product stored in one of my entities, product not stored in one of my entities but present in the organization
    - product name
    - CMR flag, CMR category
    - empirical formula
6. HTML/CSV export

Product details



1. Product card
2. Store the product in my entity
3. Display storage of this product in my entity
4. Display points of contact of other entities storing this product
5. Stocks
6. Edit stock min and max (for information purpose only, do not send any alert)
7. Current stock: in the entity - Σ: including the sub entities

Storage

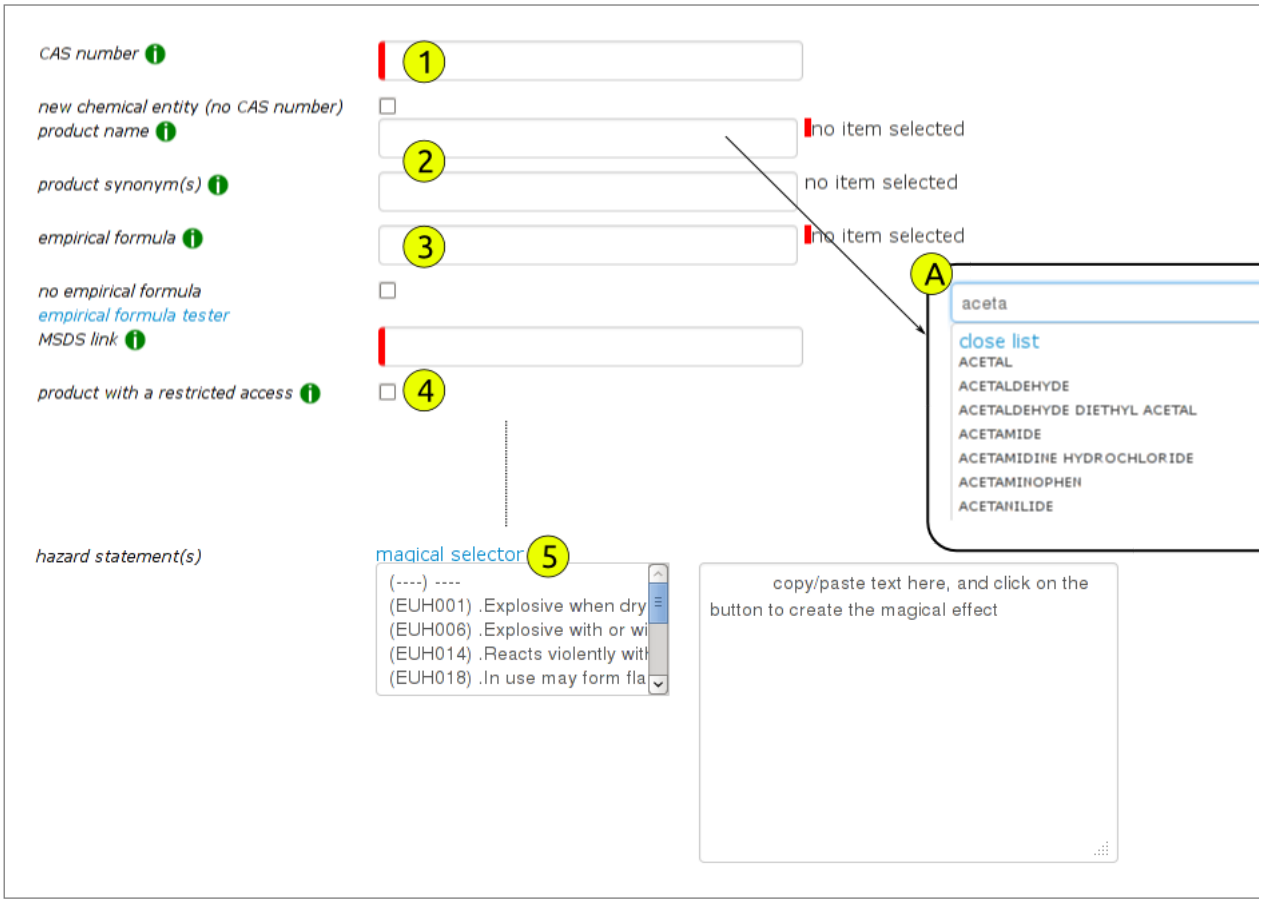


1. Storage creation date
2. Volume or weight
3. Store location name
4. Barecode
5. Store location color
6. Entity name
7. Mark this storage for disposal (does not delete the storage)
8. Borrow this storage

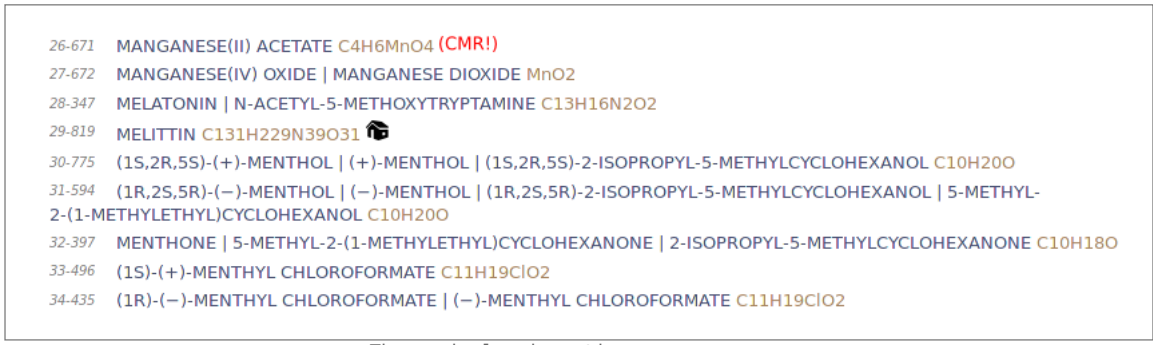
> **Note**
> Borrow a storage: if you take a bottle on a shelve, mark it as "borrowed" to inform other users.

> **Note**
> Mark a storage for disposal: expired products can be marked until they are disposed.

Product card creation



1. CAS number
   - The couple "CAS number/specificity" must be unique.
     You may enter two products with the same CAS number but with two differents specificities (example: DIETHYL ETHER C4H10O 99.5%, Extra Dry over Molecular Sieve, Stabilized AND DIETHYL ETHER C4H10O 99+%, pure).
   - The application will automatically display the products with the same CAS number.
   - The application will automatically check that the CAS number format is correct.
2. Product name and synonyms
   - You can enter only one name but several synonyms.
   - A drop down list (A) displays the existing entries. You can select one or use the ☞ icon to insert a new one.
   - Can be entered with the syntax *stereochemistry@product_name*.
     example: `(1S,2R,5S)-(+)-MENTHOL` can be typed `(1S,2R,5S)-(+)@MENTHOL`.
     With this method products can be listed in alphabetical order with the product name only. In our example `(+)@MENTHOL` will be with the "M" products and not at the top of the list (parenthesis are always at the top).



The magical @ in action.

3. Empirical formula - must follow the following rules:
   - The atoms must be in the periodic table of the chemical elements.
   - *Salts and hydrous* compounds must be typed like `AgClO4.xH2O` (put a *dot* and not a comma as a separator).
   - Use a *comma* (and not a dot) for decimal numbers.
   - Numbers for isotopes must be precede by the sign `^`
   - Atoms will be automatically sorted: *C*, *H* and the other atoms in the alphabetical order.
   - Some compounds have no empirical formula. Is this case check the "`no empirical formula`" checkbox.
4. Restricted access: you can restrict access to some products. Only people with the `view restricted product card` privilege will be able to search and see these products.
5. You can select more than one risk/safety phrase holding the `CTRL` key while left clicking on phrases with the mouse. A quicker way is to use the *magic selector*.
   You can copy and paste full phrases directly from a supplier web site (and click on the "`do the magic!`" button). The selector is smart enough to accept text like (Acrylamid from the Acros Organic web site):

```
H340:   May cause genetic defects
H350:   May cause cancer
H361f:  Suspected of damaging fertility
H317:   May cause an allergic skin reaction
H302:   Harmful if swallowed
H319:   Causes serious eye irritation
H373:   May cause damage to organs through prolonged or repeated exposure
```

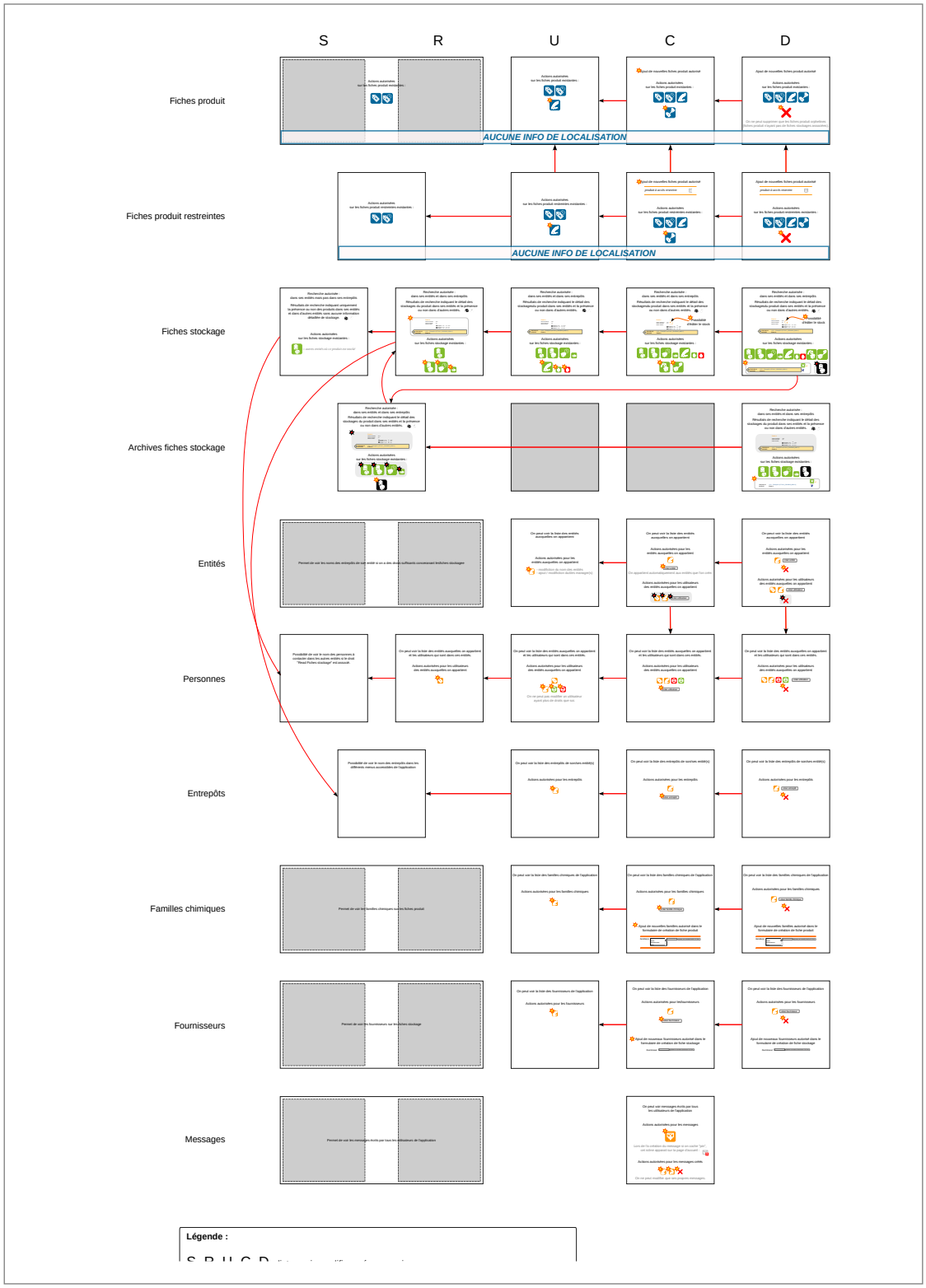
Note that phrases codes (H302, P301+P312) MUST follow the official nomenclature.

User creation

1.
   - is admin: the user will have full privileges and belongs to all of the entities
   - is in all entities: the user will belongs to all of the entities

   Tip
   Security agents may have a profile where the belongs to all of the entities but with limitated privileges. In this way they can have an overview of the stocks in the organization but can not modify them.
2. entity(ies): hold the `ctrl` key to select several entities
3. permissions:
   - `S, R, U, C, D`: select, read, update, create, delete
   - Permissions are linked: the application will automatically select linked permissions
   - Permissions are set by default: example: R product card
   - Permissions are set by default: example: R product card

Permissions dependencies



To conclude

We hope you will enjoy using "Chimithèque". If you have difficulties installing or using this application, if you find bugs or if you have ideas to improve it please contact us. Of course we can not develop features for specific use but features that might interest the community will be considered.

Some notes... I do not know where to put them. :)

Why is "Chimithèque" developed in Python? : Because it is the best language ever ! ;) No, we could have choosen Java, Php or whatever language. I use Python everyday for system administration and I wanted to test Python as a Web development language.

Why is "Chimithèque" developed with the Web2py framework ? : Because it is the b... Oups ! I have studied 4 frameworks (Pylon, Jango, Zope and Web2py). They are all good frameworks, but given my own skills Web2py had the fastest learning curve to me.

Why is Master Yoda green ? : Well... We do not know yet.