2Org-Cows Software Documentation

Thomas Rahimi: thomas.rahimi@openmailbox.org,
Boris Kulig: bkulig@uni-kassel.de

Universität Kassel,
Fachbereich 11, FG Agrartechnik

Contents

1 Introduction

The 2Org-Cows project is multinational and multi-partner project of research institutions in organic agriculture in Europe. It aims to improve the breeding of dairy cattle for organic dairy cow keeping and therefore aims to collect data to measure different parameters of animal welfare, health and keeping.
In order to analyze all these data, a huge database has been created, which consists of the following components:

- a webbased, graphical user interface for the final user to access and query the database using a standard webbrowser
- a variety of interfaces to other programs
- software to analyze database stored datasets using big data methods, mainly based on R and on Python3
- MySQL databases as the storage backbone of the whole software and project
- Ubuntu 16.04 operating system, PHP7 scripting language and Apache2 webserver to provide the user with the data required

In the following all parts of the software for the 2Org-Cows project should be documented. Please note that this file is work in progress and will therefore change steadily.

2 Common information regarding the coding

The project is generally based in its programming on several concepts, which should be briefly displayed in the following. The programming is therefore differenciated in the parts regarded.

Web Frontend The web frontend is largely written in PHP7 and HTML5, following the following paradigms:

- clear structure of code
- clear separation of display code and code for the processing of input
- easy to write, understand and maintain code
- mostly procedural code style, only exception is the database system, which is based on object oriented code style

3 The webbased User Interface

The user interface for the database has been developed as a graphical interface to allow broad acces to the database. It largely relies on standard web technologies such as HTML5, CSS3, JavaScript and PHP7. All sites are connected in the session management, which regulates the access of users to the pages. Therefore all sites, except login, are only visible to authenticated users.
The appearance of all sites is specified in the 2Org-Cows.css, which acts as the only stylesheet in the whole user interface. Only in some limited situations, style information is provided in HTML5 inline commands. These cases mainly occur in table-styled content, where the font definition is separetely on each sheet. In the first development stage only limited responsiveness of the design is achieved, further development will focus on the responsiveness of the design and the right alignment of user interface elements in HTML5 on small screen devices, such as smartphones.

3.1 login.php

The login.php site serves as the welcome site of the whole project, including basic information on the function of the sites for user access to the database.
For security reasons and to track the visitors of the page, basic information about the visitor is stored in a separate database. This includes IP addresses, user agent of the browser and the time of access.
The login works with HTML5 forms, organized in a table to allow easier configuration to various screen sizes and offer the entering of username and password. These forms consists of a text input with the input widget for text, to enter the username and a password input with the password widget. To prevent CSRF-attacks on the visitors of the page, a random sequence is generated by the server and linked to the page. This sequence is requested in the processing of the login during login_script.php to check for CSRF-attacks and to issue an alert if the sequences do not match. Furthermore there is a submit button, which is submits the form data. Submitting this form starts processing of the form data with login_script.php .

3.2 login_script.php

The login_script.php script processes the data submitted from the login form. For security reasons, authentication makes use of two different databases. The authentication database does not include any other data than the data required to initiate a session. In contrast, the agri_star_001 database includes more details on the specific user, including the insititution and the role of the user.
To ensure the integrity of the user request for authentication the form on login.php includes a random sequence, which is both stored, in the session and the POST submission. The compar-

ison of the both results should ensure better protection from XSS-attacks.
Data processing happen in 4 steps, which come up sequencially.

1. At first the script checks, whether a username exists, which matches the username provided in the form. In case the username can not be found in the database “auth”, the script redirects the user to login.php and creates error message, indicating unknown username. Otherwise the script continues to step 2.

2. If the username has been found, the script checks, whether the password, provided by the user, matches the password stored in the “auth“ database. Therefore the PHP function password_verify() is used. If the password provided does not match to the hash value, stored in the database, the user is redirected to login.php, receiving an error message about the password. Otherwise the script starts to initiate a session.

3. In the next step the script starts to create a session for the use of the research database. In order to ensure the security of the session, several steps are taken:

   - the IP address of the user is stored to prevent session hijacking
   - the user agent of the user’s browser is stored to prevent session hijacking
   - the user’s institution is stored to regulate access to the data stored in the research database, this information is encoded in the group id, which is set as a session variable, instead of the group name and the department
   - the username is stored into the session to allow recognition of the user
   - the user-id is stored into the session, because it is the primary key to the users in the different databases

   Afterwards, the user is redirected to search.php, which acts as the main page.

4. In case no login information has been provided by the user, the user is redirected to login.php .

The database connections are closed at the end of each script to prevent accumulation of unused database connection by various scripts.

3.3 create-user.php

The create-user.php site offers the opportunity to create a new user and to set the right for this specific user. User creation follows the idea of dropping rights, which only allows a user to create another user with less rights than he has himself. To ensure this, create-user.php is just visible to users with a role, higher than student. This includes the following roles: admin [role code: 4], professor [role code: 3], scientific coworker [role code: 2].
Furthermore only professors and admins are allowed to create users at a different institution,

scientific coworker are only allowed to create users at the same institution as they are. In the same way, the erasing of users is limited to professors and admin users.
Creating a user leads to modifications in the Dim_User table and the authentication database. This database updates are proceeded, if the user clicks on the “create user” button, which calls the “create_user_script.php” file to process the form data and insert them into the database.
The creation of new institution is also limited to people logged in as professors and admin. Each department is created separetely, even if an institution owns several departments, taking part in the 2Org-Cows project. This improves the adjustability of rights sticked to the specific department in case of splitted user rights on certain datasets. The data inserted for the creation of an institution are processed by the create_institution_script.php file.

3.4 create_user_script.php

The create_user_script.php is the processing file for the entries to create a new user in the web interface for the database. Basically the script just performs the validity checks of the form data provided in create-user.php and an **INSERT** query to the MySQL database.
At the beginning the script checks whether the provided check bytes submitted in create-user.php match those from the session. Again this is a CSRF-protection in case the session has been hijacked. If there are no problems on the CSRF protection, the script checks for the validity of the form provided. This part already takes place in the html, where the tag “required” is set for all fields, but a controll in the script should avoid empty fields in the database.
For the validity checks the following steps are taken:

- validity check of the e-mail adress provided
- check whether the desired username is available
- check, whether the password matches matches the control field
- check whether the user permissions, entered in the form, are allowed to be performed by the user invoked
- check whether the user invoked is allowed to select an institution different from his own

If any of these checks fails, the user receives an error notice and is directed to the form.As the group, department and the country entries are stored in a different way in the database, than in the user session, the database values have to be retrieved from the database. Therefore, the corresponding institution, department and country are queried from the Dim_Group table. As the country is stored as an integer value, the corresponding ID_Country has to be queried from the Dim_Country in the next step.
The passwords are stored as bcrypt hash values in a separate database, the hashing is performed after the validity checks for the form. The next step is to insert all data into the

Dim_User table in the agri_star_001 database. The insert is proceeded in an objective style, generating the query first, then assigning the values to the query (mysqli::prepare($query), mysqli::bind_param(parameters)) and than executing the query (mysqli::execute). The transaction id of that insert is retrieved, as it is used as ID_User to identify the user in the various databases and therefore also in the auth database. In the next step the username, the password hash and the ID_User are inserted into the auth database, using the same process as for the Dim_User table.
The scripts quits with a success message.

4 Database

The database for the projects are API-compatible, SQL databases, which run on freely available linux systems. Eventually the used and tested MySQL or MariaDB databases could be replaced by other systems, such as MSSQL. Nevertheless, a full compatibility of other databases with the source code provided can not be guaranteed. To ensure safe operation of the server, databases, which are not directly linked to the project data, are separated from the main database. As a result, the login database and the external logging database are separated from the main database. Therefore, three different databases exist for the purpose of the project.
To increase the security of the database and to limit the risk of damage in case of SQL-injections, all databases are accessed by non-privileged users. Furthermore, there is a separation in the users, regarding the function of the database invoked, to limit access to sensitive data in the database.

5 Operating System

5.1 Ubuntu

In Ubuntu less modifications than in CentOS are necessary to allow a stable and safe operation of the server. As the operating system ships another mandatory access controll system (MAC), configuration of user rights is far easier and faster proceded. During the installation of the operating system, the LAMP stack and the openssh server have been chosen for automatic installation. After the installation and the required reboot, updates should be checked and installed.

Database installation and configuration As Ubuntu ships MySQL as its default database, MySQL is used in this installation aswell. The database comes with a basic configuration, which should be checked after installation to match the conditions, under which the database server is operating.
If necessary the configuration file under /etc/mysql/conf.d/server.conf has to be altered. It is

important to check, if the networking access to the database server is allowed and if, which clients are gained access to the databases. For security and compatibility the setting listen: 127.0.0.1 has been chosen here, which allows the network interface of PHP7 to access the database locally, but hinders external clients from initiating a network connection to the database.

Web server installation and configuration Both, apache2 and PHP7 have been installed in the initial installation, together with the database. As the web server should be configured as a virtual host, it is important to modify the settings here. There have been two virtual hosts defined, both referring to the same content folder, one host serves as the non-tls demo installation, whereas the other host is set up with tls support to deliver the results of the requests as encrypted material. The non-tls host is not intended to be used for login, as all user data sent to this host are in danger of being intercepted. The configuration proceeds in the following steps:

1. First the directories for the virtualhost are created, using the following command:

```
1  sudo mkdir -p /var/www/path_to_directory/content
```

2. Next the permissions on the folder are set to the user invoked or a specific user for the deployment of the web server:

```
1  sudo chown -R $USER:$USER /var/www/path_to_directory/
       content
```

3. As the next step, the configuration files for the virtual host are created. To minimize spelling errors, the files are generated from the default configuration, which is copied into a new file:

```
1  sudo cp /etc/apache2/sites-available/000-default.conf /
       etc/apache2/sites-available/name_virtual_host.conf
```

4. Now the virtual host has to be configured, with the following settings:

```
1  <VirtualHost *:443>
2          SSLEngine On
3          SSLCertificateFile /etc/ssl/certs/name_of_cert.crt
4          SSLCertificateKeyFile /etc/ssl/private/key.key
```

```
5           SSLCACertificateFile /etc/ssl/certs/certificate.crt
6           ServerName www.servername.com
7           ServerAlias servername.com
8           ServerAdmin admin@e-mail.com
9           DirectoryIndex login.php
10          ErrorDocument 404 /error.php
11          DocumentRoot /var/www/path_to_directory/content
12          ErrorLog /var/www/path_to_directory/error.log
13          CustomLog /var/www/path_to_directory/access.log
              combined
14          Alias "/admin" /var/www/path_to_directory/content/
              create-user.php
15          Alias "/measurement" /var/www/path_to_directory/
              content/create-measurement.php
16          Alias "/search" /var/www/path_to_directory/content/
              search.php
17          Alias "/results" /var/www/path_to_directory/content
              /search-results.php
18          Alias "/license" /var/www/path_to_directory/content
              /license.php
19          Alias "/upload" /var/www/path_to_directory/content/
              database-update.php
20          Alias "/cow" /var/www/path_to_directory/content/cow
              .php
21          Alias "/logout" /var/www/path_to_directory/content/
              scripts/logout.php
22          Alias "/home" /var/www/path_to_directory/content/
              home.php
23          Alias "/user" /var/www/path_to_directory/content/
              user-properties.php
24      </VirtualHost>
```

The "Alias" settings maskerade the files, visible to the user with a shorter, easier URL.

5. In the next step, all files named in the virtual host configuration, must be created or copied to the server. To create the files, the following command is used:

```
1   sudo touch /var/www/path_to_directory/error.log
2   sudo touch /var/www/path_to_directory/access.log
```

5.2 CentOS 7

In CentOS 7 several modifications were necessary to provide a stable and safe operation of the database and the interfaces. The following describes the installation of the required packages to a scratch installation. In case a LAMP installation has been chosen during the installation process of the operating system, some of these steps might be superfluous.

Database installation and configuration First the database had to be installed. Therefore **MariaDB** has been chosen, as it is the default database environment for RHEL-based Linux distributions. For the installation, mariadb-server has to be chosen, all dependencies are automatically fixed by the yum RPM wrapper. Furthermore PHP7 had to be installed, here it is important to install the database drivers as well, which are shipped in the package php70w-mysqlnd. Together with PHP7 httpd, as the web server, needs to be installed. For the access of httpd to the database it is important to execute the following commands:

```
1 sudo sestatus
2 sudo getsebool -a | grep httpd
3 sudo setsebool -P httpd_can_network_connect_db 1
```

Web server installation and configuration As the LAMP stack is used for the project development, **apache2** is deployed as the default web server. apache2 is installed as a dependency of PHP7 and is shipped in the package httpd. To ensure proper operation, the server has to be configured for shipping of multiple websites beside each other, which is ensured by using virtual hosts. Therefore the following steps are necessary:

1. create a folder for the virtual host, using the following command:

```
1 sudo mkdir -P /var/www/path_to_directory/content
```

2. grant the required permissions on the folder to the user:

```
1 sudo chown -R $user:$user /var/www/path_to_directory/
    content
2 sudo chmod -R 755 /var/www/
```

3. create configuration files for the virtual host:

   - Create the folders for the virtual host configuration:

```
1 sudo mkdir /etc/httpd/sites-available
2 sudo mkdir /etc/httpd/sites-enabled
```

- Add a line at the end of /etc/httpd/conf/httpd.conf:
  IncludeOptional sites-enabled/*.conf

6 License

The whole program and this documentation underlie the following license:

7 Attachements