Our Ocean Conference Software Stack

Technical Documentation

Version 2.0
Brussels, April 2020

TABLE OF CONTENTS

SYSTEM ARCHITECTURE

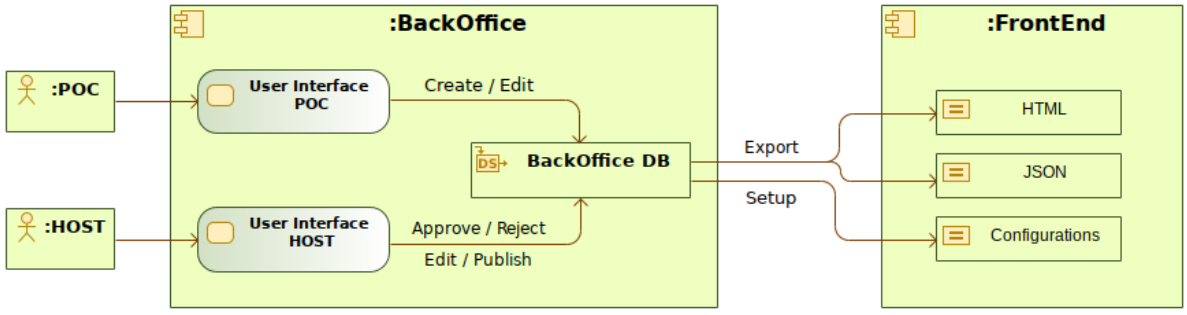
The system is composed of two separate applications:

1. BackOffice

   This application is dedicated to the creation and management of commitments by persons of contact (called POC) and conference hosts (called HOST). It is also used to control the publishing of submitted commitments, users and organisations. The application is not publicly available and requires authentication.

2. FrontEnd

   This application and displays all the commitments and related organisations and persons of contact under the form of maps, tables and statistics. It is publicly available and doesn't require authentication



The BackOffice application is used to create and manage all types of data like organisations, users, commitments, themes, schedules, etc. The functionalities provided by the application differ between the type of user:

1. **POC**
   - Update user profile
   - Update organisation profile
   - Invite colleagues
   - Create and edit commitments
   - Submit commitments for approval and publication
   - Update commitment's progress and impact
2. **HOST**
   - Update user profile
   - Update organisation profile
   - Invite colleagues

- Approve, reject, request changes and publish commitments
- Approve and reject users
- Edit organisations
- Manage commitment themes
- Manage organisation types
- Configure the FrontEnd application
- Configure a publishing scheduler

The FrontEnd, an embeddable JavaScript web application, is consuming BackOffice data through static JSON and HTML files that are generated while publishing commitments. On the first FrontEnd access these files are loaded and cached. Search and refining processes are then performed on the client side only. This architecture was chosen in order to reduce as much as possible network, server and database processes. Following the green-IT initiative, this strategy reduces the carbon footprint at the server level and also limits its exposure to hacking.

SPECIFICATIONS AND TECHNOLOGY STACK

This section specifies the requirements to run both applications and the technologies used in each of them. Links to all used technologies are also provided.

1. **BackOffice**

Requirements	Technologies
Apache Web Server with rewrite engine enabled	CodeIgniter v2.2.6
PHP v7.1	GD module for PHP
MySQL or MariaDB	PHPMailer v6.0
	PHPDotEnv v3.3
	Cron-Expression v2.2
	Bootstrap v3.3
	jQuery v3.3.1
	Leaflet v1.4.0
	C3.js v0.7.15

2. **FrontEnd**

Requirements	Technologies
Apache Web Server with rewrite engine enabled	Bootstrap v3.3
PHP v7.1	jQuery v3.3.1
	Leaflet v1.4.0
	C3.js v0.6.12

DATABASE STRUCTURE

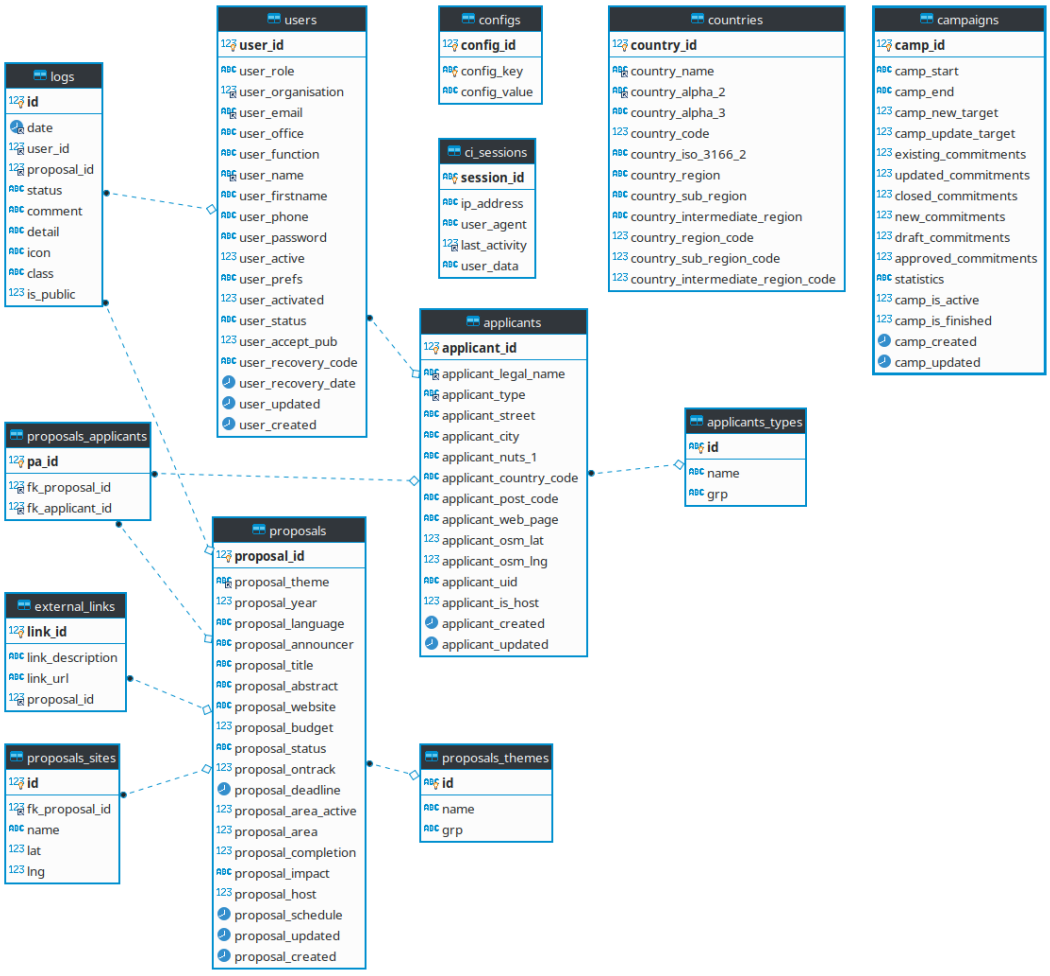
The database structure and nomenclature have been inherited from an external existing system (SME Datahub) and, therefore, the naming conventions differ slightly. The following convention was used:

- *PROPOSAL* entity corresponds to *COMMITMENT*
- *APPLICANT* entity corresponds to *ORGANISATION*

It should also be noted that, in the context of OOC, some of the inherited entities are not being used within the system. These are the following:

- *CALLS*
- *IMPORTS*
- *IMPORTS_DATAS*
- *IMPORTS_LOGS*
- *NUTS*
- *TOPICS*

The structure of the relational database is shown in the following entity-relationship diagram.



For further and more detailed database documentation please refer to the DB Docs folder provided with the software.

INSTALLATION AND CONFIGURATION

This section describes the necessary steps to make a fresh install of both the BackOffice and FrontEnd applications. It assumes that the Apache Web Server, PHP and GD module for PHP are installed and working as expected.

Files and Folders

Copy both the *backoffice* and *datahub* folders to your server root. Create two additional folders in your server root:

- A folder called *uploads* where all uploaded files will be temporarily living
- A folder called *logs* where all report files will be stored (mainly useful for development and debugging purposes)

Make sure that you adjust folder permissions in order to make them writable for the application (typically setting user rights to 0755) according to the following:

- server_root/uploads
- server_root/logs
- server_root/backoffice/assets/images/ooc
- server_root/datahub/config.php
- server_root/datahub/json
- server_root/datahub/json/beneficiaries
- server_root/datahub/json/projects
- server_root/datahub/src/images/ooc

Install PHP Dependencies

Start by installing Composer. Then navigate to server_root/backoffice and run the following command:

```
php composer.phar install
```

This will install the necessary PHP dependencies for the BackOffice application, namely:

1. PHPMailer
2. PHPDotEnv
3. Cron-Expression

You should also manually install PHP MySQL drivers in your system.

Install the Database

You can install either MySQL or MariaDB. Once installed you should create a new database and run the SQL files provided with the software, namely:

1. dg_mare.sql
2. data.sql

These will create all necessary tables and seed the default initial data. A host user is also created according to the following:

- user email: admin@ooc.com
- user password: admin

Apache Web Server Configuration

Two different virtual hosts need to be configured in you web server, one for the BackOffice and another for the FrontEnd. In order to have human readable URLs, rewriting rules have to be added to each virtual host configuration.

1. Virtual Host Configuration for the BackOffice

```
<VirtualHost *:80>
    ServerName server_name
      ServerAlias server_alias
      DocumentRoot server_root/backoffice

    <Directory /var/www/html/backoffice>
        Options +FollowSymLinks -MultiViews
        RewriteEngine on
        RewriteBase /
        RewriteCond %{REQUEST_FILENAME} !-f
        RewriteCond %{REQUEST_FILENAME} !-d
        RewriteRule ^(.*)$ index.php/$1 [L]
    </Directory>

    <Files ~ "\.env$">
        Order allow,deny
        Deny from all
    </Files>
</VirtualHost>
```

2. Virtual Host Configuration for the FrontEnd

```
<VirtualHost *:80>
    ServerName server_name
      ServerAlias server_alias
      DocumentRoot server_root/datahub

    <Directory /var/www/html/datahub>
        Options +FollowSymLinks -MultiViews
        RewriteEngine on
        RewriteBase /
        RewriteRule ^project\/(.*)$ project.php?id=$1 [L]
        RewriteBase /
        RewriteRule ^beneficiary\/(.*)$ beneficiary.php?id=$1 [L]
    </Directory>
</VirtualHost>
```

Application Level Configurations

All configurations related with the application must be done by manually editing the *config.env* file inside the *backoffice* folder. All the necessary settings are described below.

Setting Name	Description	Example Value
BO_APP_ENV	Specifies the running environment. It suports two values: *development* or *production*	production
BO_APP_URL	The URL which will be in use for the backoffice application	https://ooc.com/
BO_APP_COOKIE_DOMAIN	The cookie domain that will be created by the backoffice application	.ooc.com
BO_APP_TIMEZONE	The timezone to be used by the backoffcie application. Supported values can be viewed here.	UTC
BO_APP_UPLOADS	System folder where all uploaded files will be stored	server_root/uploads
BO_APP_REPORTS	System folder where application logs will be stored	server_root/logs

BO_EMAIL_FROM	Email used for sending out emails	no-reply@ooc.com
BO_EMAIL_FROM_NAME	Name to be used in the FROM clause of all emails being sent out	Our Ocean Conference
BO_CAPTCHA_SITE_KEY	The site key from google captha v2	
BO_CAPTCHA_SECRET_KEY	The secret key google captcha v2	
DH_APP_URL	The URL which will be in use for the frontend application	https://viewer.ooc.com/
DH_MAPS_FOLDER	The system folder containing the frontend application	server_root/datahub/
DH_JSON_EXPORT_FOLDER	The system folder where all JSON files for the frontend application will be stored	server_root/datahub/json
DH_JSON_ITEMS_PER_FILE	The maximum number of records that each JSON file can contain	200
DH_IMAGE_EXPORT_FOLDER	The system folder where all theme images will be stored for the frontend application	server_root/datathub/src/images/ooc
MAIL_HOST	The host smtp address to be used to send out emails from the backoffice application	smtp.gmail.com
MAIL_USER	The user email account to be used within the smtp configuration	user@gmail.com
MAIL_PASS	The user password for the email account being used within the smtp configuration	my_password
MAIL_PORT	The port being used by the smtp service	465
MAIL_PROTOCOL	The protocol being used by the smtp service	ssl
BO_DB_HOST	The host address of the database	10.10.10.10
BO_DB_NAME	The database name	dg_mare

BO_DB_USER	The database user name	db_user
BO_DB_PASSWORD	The password for the database user name	db_user_password

Setup the Publishing Scheduler

The BackOffice application allows to define a scheduler for automatically publishing approved commitments. This functionality is only available for HOST users. In order to have it working properly a *cronjob* must be setup on the server by following the next steps:

1. Open the *cronjob* file

```
sudo crontab -e
```

2. Insert a new *cronjob*

```
* * * * * sudo su www-data -s /bin/sh -c "/usr/local/bin/php
/server_root/backoffice/index.php publish publish_cron"
```

Setup the Campaign Scheduler

The BackOffice needs to have a scheduler setup in order to properly compute statistics for a running campaign and also to automatically close in due date .

1. Open the *cronjob* file

```
sudo crontab -e
```

2. Insert a new *cronjob*

```
1 0 * * * sudo su www-data -s /bin/sh -c "/usr/local/bin/php
/server_root/backoffice/index.php campaign campaign_cron"
```

AMAZON AWS DEPLOYMENT

Both applications were deployed in Amazon AWS Lightsail. Lightsail is an easy-to-use and cost-effective cloud platform that supports all the necessary infrastructure and technology stack to deploy the OOC system. An instance was created in Lightsail using the LAMP stack:

- Linux Ubuntu Server v16.04.5 LTS
- Apache Web Server v2.4.37
- MySQL v8.0.13
- PHP v7.1.25
- 80 GB of internal storage available

The installation procedure was similar to the one explained before. Some adaptations were needed due to the particularities of the cloud service. The following sections highlight these changes.

Registered DNS

The following domains and subdomains were registered and configured:

- ouroceanconference.org
- www.ouroceanconference.org
- viewer.ouroceanconference.org

Files and Folders

All files and folders were copied to:

```
/opt/bitnami/apache/htdocs
```

Database

A database instance was created running MySQL RDBMS. All previous OOC data (prior to 2019) was imported into this database.

Apache Web Server Configuration

The virtual hosts configuration was made in:

```
/opt/bitnami/apache/conf/bitnami/bitnami.conf
```

SSL certificates were create using Let's Encrypt. All traffic towards non encrypted HTTP are being redirected to HTTPS. A *cronjob* was created in order to automatically renew the certificates every 90 month.

Email SMTP Configuration

A production Simple Email Service (SES) was requested to Amazon and is already setup in the OOC AWS account. The AWS SES credentials were used to configure the emailing functionality of the application.

Application Level Configurations

All configurations at the *config.env* file were adjusted to reflect the Lightsail setup.

Setup the Publishing Scheduler

The cronjob was adjusted in order to reflect the specificities of the cloud instance.

```
* * * * * sudo su daemon -s /bin/sh -c "/opt/bitnami/php/bin/php
/opt/bitnami/apache2/htdocs/backoffice/index.php publish publish_cron"
```

Setup the Campaign Scheduler

The cronjob was adjusted in order to reflect the specificities of the cloud instance.

```
* * * * * sudo su daemon -s /bin/sh -c "/opt/bitnami/php/bin/php
/opt/bitnami/apache2/htdocs/backoffice/index.php campaign campaign_cron"
```

Setup the SSL certificate renewal Scheduler

The cronjob was adjusted in order to reflect the specificities of the cloud instance.

```
* 1 1 * * sudo sh /opt/bitnami/letsencrypt/scripts/renew-certificate.sh
2 > /dev/null
```