Costasiella

Edwin van de Ven

Mar 18, 2023

GENERAL

So far not much here, but will be added bit by bit in the coming months…

CHAPTER

ONE

FAQ

1.1 Internationization

Will any other languages besides English be officially supported?

Due to high development workloads, only support English is supported at the moment and unfortunately we don't have the capacity to add or manage other languages.

That being said, we're working on an improved translation system that will allow easy integration and maintenance of community maintained translations, however, this is still very much a work in progress and an ETA has not yet been set.

Can I add other translations myself?

Soon documentation on how to add additional languages will be available. For the time being, it's not possible yet to add additional languages yourself.

1.2 Integrations

Can Costasiella be integrated in a Squarespace or Wix website?

The short answer is no. As these website builders operate a closed platform, the options for integration are limited until there's an official plugin (which is very unlikely unfortunately, as only very big companies get an official plugin). There is an option for third parties to inject code, but the supported technologies aren't sufficient for a satisfying integration. In case you'd like to integrate Costasiella on your website, it's probably best to move to a more open system like WordPress, Drupal or CMS made simple to name a few.

Are other online payment providers supported?

Mollie is the only supported payment provider at the moment. This means that you won't be able to use online payments if you're based outside of the EU. More integrations are planned. Please let us know which ones you'd like to see, they'll be added in order of populate demand. In case you'd like to sponsor the integration of a payment provider by financial support of by donating code, please get in touch using the contact page!

1.3 Self hosted - DIY

Are 2 (v)CPU cores really necessary when hosting it yourself?

Well, the answer is yes and no. Technically you can run Costasiella on a single CPU core. Though you will find that Costasiella will run quite slow at times and pages might take more then a few seconds to load. This is caused by the fact that you can only run one app worker process comfortable per core. With one core and one process serving requests, it means that when the process is busy, all other requests will have to wait. Imagine the case where someone using a mobile phone using a poor connection tries to load a page with a few images like the list of events. The phone will make one request for the page itself and one for each image. Until all requests are handled, other users will have to wait. However this is in case there is only 1 CPU core and 1 app worker process. With a 2nd CPU core and worker process additional requests can be handled without everyone having to wait on the user with the poor connection. Another benefit is much quicker loading times in general. In many cases a few requests need to be made before a page can be shown. The page itself, style sheets, scripts and images are all separate requests. The more worker processes, the more requests can be handled in concurrently. Although this isn't true in all cases, for the sake of simplicity you could say that with a single CPU core and app worker process requests are handled one by one, in a serial way and with multiple CPU cores and app worker processes requests can be handled in parallel, greatly reducing loading times of even single pages.

CHAPTER

TWO

MANAGING ACCOUNT PASSWORDS

Due to spam filtering or other issues, a small subsection of users might have an issue using the password reset function. Follow the steps below to manually set a password for their account.

2.1 Prerequisites

Ensure the account you're using has the "Staff" permission to be able to access the admin area. (Not the backend, the admin area). In case you're not sure if your account has the "Staff" status, please check with your system administrator.

2.2 Assigning a password to an account

1. Sign in to Costasiella's admin page at <your-costasiella-address>/d/admin. eg. https://example.costasiella.com/d/admin
2. Navigate to accounts in the menu on the left.
3. Use the search box in the top of the page to search for the email address of the account you want to set a password for or find it in the list.
4. In the password field look for the very small text "Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using this form."
5. Click the "This form" link.
6. Set the desired password twice.
7. Click the Change password button.
8. The chosen password is now assigned to the account.
9. For security reasions, sign out of the admin section as soon as you're done using it.

Done.

CHAPTER

THREE

SETUP PRODUCTION ENVIRONMENT

A quick note: This guide should be mostly correct and complete, but hasn't passed final review yet. In case you run into a problem, please create an issue on github so it can be checked."

3.1 Prerequisites for this guide

- A DNS record pointing to the server should be in place, eg. demo.costasiella.com. This is not supported: costasiella.com/demo
- A Ubuntu 20.04 server (Another flavour of Linux is also ok, but you might have to change some package names and take into account that not every distro build their packages with the same parameters).
- At least 2 (v)CPU cores with performance equal to a 2nd generation Intel i5 desktop processor or greater.
- At least 4 GB of RAM
- Sufficient storage to hold the infrastructure components (5GB of free space should easily suffice, but more is recommended to have space for media in the application).

3.2 Target audience

This guide assumes some experience in Linux system administration. In case you're not comfortable with or knowledgeable about any of the following items, it might be best to ask an experienced Linux system administrator for help. Although you might be able to follow this guide by simply copying commands, there's a chance you create security issues exposing your customer data if you don't actually understand what you're copying and why.

- Configuring DNS records
- Running commands on a Linux terminal
- Managing Nginx
- Managing MySQL
- Managinx Docker
- Managing SSL certificates
- Configuring backups on a Linux server

3.3 Introduction

This guide is one possible way of setting up a production environment for Costasiella. Some components have been put into containers, while others are deployed in a more traditional way on a Linux server. Basically these choices come down to personal preference. This guide keeps some things on the host so they can be easily backed up using traditional methods. Costasiella code and required modules are packed in docker containers, to ensure compatibility and prevent messing with (perhaps incompatible) modules on a server. Of course, you're completely free to deploy everything directly on a server or containerize everything. You've got access to the source code, so build the environment that suits you.

The production setup example in this guide consists of the following components:

A Ubuntu 20.04 server (Host) with the following components:

- Nginx (Host)
- MySQL server (Host)
- Hashicorp Vault (Host)
- Costasiella frontend (Host)
- Redis (Docker)
- Costasiella backend (Docker)
- Costasiella celery worker (Docker)
- Costasiella celery beat (Docker)

3.4 Quick start

In case you're looking to evaluate Costasiella, the guide on setting up a local development environment might be a bit quicker. However, never use that setup on a production system. It's not secure enough, doesn't scale well when it comes to performance and will be hard to maintain.

For a production system, there are a number of components that need to be installed and configured to get Costasiella up and running. No way around it.

3.5 Preparation

Sign up for reCAPTCHA and create keys that apply to your domain, eg. demo.costasiella.com. https://www.google.com/recaptcha/about/

3.6 Install required packages

```
sudo apt install docker docker-compose git mysql-server nginx-full
```

3.7 Set server time to UTC

```
sudo timedatectl set-timezone UTC
```

3.8 Find docker interface IP address

Use the command "ip a s" to show all network interfaces and look for the one called "docker0"

```
$ ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen␣
↪1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
    valid_lft forever preferred_lft forever
2: enp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group␣
↪default qlen 1000
    link/ether 52:54:00:11:41:6d brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.233/24 brd 192.168.122.255 scope global dynamic noprefixroute enp3s0
    valid_lft 3581sec preferred_lft 3581sec
    inet6 fe80::3607:3352:99b6:b438/64 scope link noprefixroute
    valid_lft forever preferred_lft forever
3: br-a80887c9ec37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP␣
↪group default
    link/ether 02:42:fa:8e:cc:22 brd ff:ff:ff:ff:ff:ff
    inet 172.18.0.1/16 brd 172.18.255.255 scope global br-a80887c9ec37
    valid_lft forever preferred_lft forever
    inet6 fe80::42:faff:fe8e:cc22/64 scope link
    valid_lft forever preferred_lft forever
4: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group␣
↪default
    link/ether 02:42:50:7f:0c:07 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
    valid_lft forever preferred_lft forever
```

In this case the IP address of the docker interface is 172.17.0.1. This address will be used a few times in this guide. Check which address your docker interface is configured to use and make a note somewhere it's easy to reference when you need it.

3.9 MySQL configuration

Edit mysql server config in /etc/mysql/mysql.conf.d/mysqld.cnf. Set the bind address to localhost (127.0.0.1) and the docker interface address (172.17.0.1 in this example). Restart the mysql service after chaning the configuration

```
bind-address            = 127.0.0.1,172.17.0.1
```

Create database for Costasiella & Vault. In this example a user with the username "user" and password "password" is created. For your environment, set something more secure than "password" as the password. This user can access the MySQL server from the 172.17.0.0/16 docker subnet. Something more secure is hightly recommended.

```
sudo mysql
mysql> create database costasiella;
mysql> create database vault;
mysql> create user 'user'@'172.%' identified by 'password';
mysql> grant all privileges on costasiella.* to 'user'@'172.%';
mysql> grant all privileges on vault.* to 'user'@'172.%';
```

In case you can't restart the vault service because the user can't connect:

```
mysql> create user 'user'@'localhost' identified by 'password';
mysql> grant all privileges on vault.* to 'user'@'localhost';
mysql> flush privileges;
```

3.10 Install Hashicorp Vault client

Vault is installed on the host for easier management of the Vault container in the Docker compose stack. It isn't used to serve Costasiella.

Please visit the following URL and follow the setup steps of your chosen method. For this guide using the package manager (apt) is assumed. https://learn.hashicorp.com/tutorials/vault/getting-started-install

After installing vault, disable it from starting as a service.

```
sudo systemctl stop vault
sudo systemctl disable vault
```

Add the following to your .bashrc or .zshrc or whatever file your shell uses. Also type the command in your current shell to be able to execute vault commands.

```
export VAULT_ADDR=http://127.0.0.1:8200
```

3.11 Backend setup preparation

Create directories to hold docker bind mounts

```
mkdir -pv /opt/docker/compose/costasiella
mkdir -pv /opt/docker/mounts/costasiella/logs
mkdir -pv /opt/docker/mounts/costasiella/media
mkdir -pv /opt/docker/mounts/costasiella/media_protected
mkdir -pv /opt/docker/mounts/costasiella/sockets
mkdir -pv /opt/docker/mounts/costasiella/static
mkdir -pv /opt/docker/mounts/costasiella/vault_config
```

Edit Django settings

Get the costasiella common.py and production.py settings from https://github.com/costasiella/costasiella/tree/main/app/app/settings. Put common.py and production.py in /opt/docker/mounts/costasiella/settings.

Edit /opt/docker/mounts/costasiella/settings/common.py

- Replace the SECRET_KEY value with a random string that's 50 characters long.
- Update the databases section to allow the backend to connect to the MySQL server running on the host.

- Find the vault section and update it with the settings created earlier. (Note that the address 172.17.0.1 is the address of the docker interface).

```
...

else:
    DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'costasiella',
        'USER': 'user',
        'PASSWORD': 'password',
        'HOST': '172.17.0.1',
        'PORT': 3306
    }
}

...

RECAPTCHA_PUBLIC_KEY = '<Your site key here>'
RECAPTCHA_PRIVATE_KEY = '<Your secret key here>'

...
```

Save the settings file

Create Vault configuration file

Configure Vault to use MySQL storage and don't use TLS for this example guide to keep things simple. In production it's recommended to configure this.

Copy /etc/vault.d/vault.hcl to /opt/docker/mounts/costasiella/vault_config/vault.hcl Then edit the vault.hcl in /opt/docker/mounts/costasiella/vault_config

- Comment out the file storage section
- Configure MySQL storage
- Disable TLS

```
# Full configuration options can be found at https://www.vaultproject.io/docs/
↪configuration

ui = true

#mlock = true
disable_mlock = true

#storage "file" {
#  path = "/opt/vault/data"
#}
#

storage "mysql" {
username = "user"
password = "password"
database = "vault"
```

(continues on next page)

(continued from previous page)

```
plaintext_connection_allowed = true
}

#storage "consul" {
#  address = "127.0.0.1:8500"
#  path    = "vault"
#}

# HTTP listener
#listener "tcp" {
#  address = "127.0.0.1:8200"
#  tls_disable = 1
#}

# HTTPS listener
listener "tcp" {
address       = "0.0.0.0:8200"
tls_disable   = true
#  tls_cert_file = "/opt/vault/tls/tls.crt"
#  tls_key_file  = "/opt/vault/tls/tls.key"
}

# Enterprise license_path
# This will be required for enterprise as of v1.8
#license_path = "/etc/vault.d/vault.hclic"

# Example AWS KMS auto unseal
#seal "awskms" {
#  region = "us-east-1"
#  kms_key_id = "REPLACE-ME"
#}

# Example HSM auto unseal
#seal "pkcs11" {
#  lib            = "/usr/vault/lib/libCryptoki2_64.so"
#  slot           = "0"
#  pin            = "AAAA-BBBB-CCCC-DDDD"
#  key_label      = "vault-hsm-key"
#  hmac_key_label = "vault-hsm-hmac-key"
```

Restart the vault service to reload the configuration file.

Fetch the Docker compose & costasiella env files from GitHub

Get the docker-compose.yml and costasiella.env file from https://github.com/costasiella/costasiella and put them in /opt/docker/compose/costasiella. Then start the costasiella containers. At this point Vault will be unconfigured and Costasiella won't be functional yet. The next step is configuring Vault.

```
cd /opt/docker/compose/costasiella
docker compose up
```

Perform initial setup for Vault

Create an SSH tunnel to map port 8200 on your Costasiella server to port 8200 on your device. Port 8200 should not

be reachable on the server from the word wide web, please make sure to firewall it. Alternatively a cleaner approach is to create multiple listeners. One for localhost and one for the docker interface. Have a look here at the Vault docs for more info: https://www.vaultproject.io/docs/configuration/listener/tcp

For now we keep it simple in this guide. Vault will listen on all interfaces and we'll assume that you've firewalled the external interface of your Costasiella server. Using this command on your computer (Linux or Mac) will allow you to access the Vault UI on the server from http://localhost:8200 on your computer.

```
ssh -C -L 8200:127.0.0.1:8200 -N <IP of your Costasiella server>
```

Add a transit key

Open a browser and open the Vault web UI at http://localhost:8200 to do the initial setup. Set for example 5 key shares, with a threshold of 3 and click Initialize.

Download the keys and store them somewhere secure (eg. encrypted in a password manager database). You'll need them everytime Vault starts to unseal it and you'll need the root token for administration. *Continue to unseal*

Add 3 of the 5 keys, one by one, to unseal. Log in using the root token.

Go to Secrets and choose *Enable new engine*. Choose transit and click Next. Accept the default path called transit and click *Enable engine*.

Create an encryption key for Costasiella by clicking *Create encryption key*. In this guide the key name *costasiella* will be used. Add that to the name field and click *Create encryption key*.

Create a policy

To avoid having to use the root token in Costasiella, we'll create a new token to which we'll assign a policy that's limited to using the Costasiella transit key and no other functionality withing vault.

Click *Policies* in the main menu. Click *Create ACL policy*.

Name it something clear and easy to remember. In this guide "use_costasiella_transit" will be used for the policy name. Add the following to the *Policy* field.

```
# Vault transit key policy
path "transit/encrypt/costasiella" {
capabilities = ["update" ]
}
path "transit/decrypt/costasiella" {
capabilities = ["update" ]
}

# List existing keys in UI
path "transit/keys" {
capabilities = [ "list" ]
}

# Enable to select the orders key in UI
path "transit/keys/costasiella" {
capabilities = [ "read" ]
}
```

Click *Create policy*

Create a token for Costasiella

In an SSH or console session on your server:

```
vault login <root token>
vault token create -policy=use_costasiella_transit -period=768h
```

Note down this token for later use in this guide and note that the token expires in 32 days (768 hours). For security reasons, Vault doesn't allow tokens to live longer than this by default. However, it's a periodic token so it can be renewed an unlimited number of times.

```
vault login <your created token>
vault token renew
```

Don't forget to regularly renew your token to ensure Costasiella doesn't lose access to Vault.

Set token in costsasielle env file

Edit /opt/docker/compose/costasiella/costasiella.env and add the token that was created in the previous step.

Configure email

Edit /opt/docker/mounts/costasiella/settings/production.py Change the values in the email configuration section to reflect your email infrastructure.

As a general suggestion (feel free to take it or leave it) it could be wise to set up a local postfix server and point your Costasiella to that. This way there's a message queue that will hold the messages in case the SMTP server you're sending to isn't accepting email for any reason. Another benefit is simpler email configuration in your Costasiella installation. You can simply point it to the IP of the system holding your postfix server and port 25.

For example:

```
...
# Email configuration
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = '172.17.0.1'  # In case you run postfix on your docker host
EMAIL_PORT = 25
DEFAULT_FROM_EMAIL = 'My Name <my_from_email@domain.com>'
...
```

For a full list of email options, please refer to the Django documentation

3.12 Backend setup

Starting containers

Now it's time to restart the containers holding the backend code to make them aware of the config changes made in the previous steps. To do this, we're going into the folder holding the code and use docker-compose to bring the environment online.

```
cd /opt/docker/compose/costasiella
sudo docker-compose stop
sudo docker-compose up
```

Getting the environment ready

Now a few commands need to be executed inside the backend container to:

- Load fixtures
- Create an initial super user

```
docker exec -it <costasiella backend container name> /bin/bash
cd /opt/app
# Load fixtures
python manage.py loaddata costasiella/fixtures/*.json
# Create super user
./manage.py createsuperuser
```

3.13 Frontend setup

Fetch frontend code from GitHub and copy into the webserver directory

```
cd /opt
git clone https://github.com/costasiella/frontend.git costasiella_frontend
cp -prv /opt/costasiella_frontend/build/ /var/www/html/
```

Configure Nginx

Create a file representing your hostname in /etc/nginx/sites-available. In this example the file *demo.costasiella.com* will be used.

Restart the Nginx service

Apply any database updates that might be available

Open a browser and go to http://<your domain>/d/update

3.14 Configure the superuser account as a Costasiella admin

Open a webbrowser (tab) and go to <your domain>/d/admin. Log in using the initial superuser credentials created earlier.

Navigate to Costasiella > Accounts and click the email address of the superuser. Now add the user to the Admins group and click save.

Run the following code in a mysql terminal with a user that has permissions to modify your Costasiella database.

```
use costasiella;
update costasiella_account set employee=1 where id=1;
```

This enables the superuser to sign in to the backend with admin privileges.

Note: The superuser isn't created in the "regular" way. It doesn't have records in all the tables that regular accounts have. It's highly recommened to use an account created under relations > accounts that's been granted admin privileges to manage Costasiella.

3.15 Create additional admin accounts

Creating at least one other admin account is always a good idea. This way you don't have to sign in for daily use with a user that has superuser status.

Create additional account

Log in using the superuser credentials your created on <your domain name> (eg. demo.costasiella.com).

Navigate to the backend and then to relations > accounts. Add a new account. Edit the newly created account and set the Employee switch to on.

Grant Admin privileges

Go go to <your domain>/d/admin and sign in with your superuser account.

Navigate to the accounts list under the COSTASIELLA section. Click the email address of the account you just created to edit it. Add the account to the Admins group and click save.

Set password

Again click the email address of the account you just created to edit it. In the password field, click the small link "this form" to set a password.

After setting an initial password, the additional admin account is ready to be used.

Note

Preferably test it in another webbrowser. It's possible to sign in to <your domain>/d/admin and <your domain> using different accounts in the same browser. This can give rise to unexpected behavour. In case Costasiella behaves strangely for you during these steps, sign out of both the <your domain> and <your domain>/d/admin and try again.

3.16 Setting the site name

The set site name is used in some system email messages, eg. the password reset email.

Open a webbrowser (tab) and go to <your domain>/d/admin. Log in using the initial superuser credentials created earlier.

Navigate to the *Sites* section and click the default domain name *example.com*. Change the domain name and display name to reflect your installation and click *Save*.

3.17 Configuring background tasks

To have invoices marked as overdue a background task should run daily.

Open a webbrowser (tab) and go to <your domain>/d/admin. Log in using the initial superuser credentials created earlier.

Navigate to the *Periodic tasks* section and check *Add* next to *Periodic tasks* Give the task a recognizable name, eg. "Mark invoices as overdue" Choose the task *costasiella.tasks.finance.invoices.tasks.finance_invoices_mark_overdue* Ensure the task is set to *Enabled* Add a crontab schedule with the minutes and hours set to 0 and the other values to * to have the task run at midnight. The task can run at any time convenient for you, running it at midnight is just an example. Click save to finish configuring the background task

3.18 Next steps

Now the Costasiella environment is up and running, you can integrate it into your (hosting) infrastructure as required. As these steps can differ greatly depending on the environment Costasiella is installed in, these won't be detailed in this guide. Some common next steps can be:

- Publishing it through your load balancer / reverse proxy / Web application firewall
- Adding SSL certificates
- Configuring backups
- Adding the Costasiella url to your monitoring system to check if it's online and performing as it should

3.19 Troubleshooting

Database wasn't reachable on backend container start

In case a firewall or incorrect configuration prevented the backend from connecting to the database on startup, database migrations might not have run. Symptoms might include messages that tables can't be found.

Resultion: Ensure the backend container can connect to the database and restart it.

Fixtures don't load with error: Connection refused

This error occurs when the backend container can't connect to your Vault instance.

Resolution: Ensure the backend container can connect to your Vault instance.

CHAPTER

FOUR

OPENSTUDIO IMPORT

This page will explain how to import your OpenStudio data into Costasiella. Please note that the import might override some information like details of your organization. It's therefore strongly recommended to create a database & file backup of your Costasiella installation before proceeding any further.

4.1 Manual import

A few items can't be migrated automatically and should therefore be migrated manually.

- Settings
- Mail templates
- Events without a school location set
- Event ticket earlybird discounts
- School subscription price glaccount and costcenter information
- Prices for drop-in classes & trial classes
- Groups and group permissions

4.2 Data imported

Organization (school)

- sys_organizations > organization
- accounting_glaccounts > finance_glaccount
- accounting_costcenters > finance_costcenter
- tax_rates > finance_tax_rate
- payment_methods > finance_payment_method
- school_memberships > organization_membership
- school_classcards > organization_classpass
- school_classcards_groups > organization_classpass_group
- school_classcards_groups_classcard > organization_classpass_group_classpass
- school_classtypes > organization_classtype
- school_discovery > organization_discovery

- school_levels > organization_level
- school_locations > organization_location (Add one room for each added location)
- school_shifts > organization_shifts
- school_subscriptions > organization_subscription
- school_subscriptions_groups > organization_subscription_group
- school_subscriptions_groups_subscriptions > organization_subscription_group_subscription
- school_subscriptions_price > organization_subscription_price

Customer accounts

- auth_user (customers) > account & all_auth_email
- auth_user (businesses) > business
- auth_user (teacher info) > account_instructor_profile
- customers_classcards > account_classcard
- customers_notes > account_note
- customers_orders > finance_order
- customers_orders_amounts > finance_order
- customers_orders_items > finance_order_item
- customers_orders_mollie_payment_ids > integration_log_mollie
- customers_payment_info > account_bank_account
- customers_payment_info_mandates > account_bank_account_mandate
- customers_subscriptions > account_subscription
- customers_subscriptions_alt_prices > account_subscription_alt_price
- customers_subscriptions_blocked > account_subscription_block
- customers_subscriptions_credits > account_subscription_credit
- customers_subscriptions_paused > account_subscription_pause

Invoices

- invoices_groups > finance_invoice_group
- invoices_groups_product_types > finance_invoice_group_default
- invoices > finance_invoice
- invoices_amounts > finance_invoice
- invoices_customers > finance_invoice
- invoices_items > finance_invoice_item
- invoices_items_customers_classcards > finance_invoice_item
- invoices_items_customers_subscriptions > finance_invoice_item
- invoices_items_workshop_products_customers > finance_invoice_item
- invoices_payments > finance_invoice_payment
- invoices_mollie_payment_ids > integration_log_mollie

Payment batches

- payment_batches > finance_payment_batch
- payment_batches_export > finance_payment_batch_export
- payment_batches_items > finance_payment_batch_item
- payment_categories > finance_payment_batch_category
- alternativepayments > account_finance_payment_batch_category_item

Class schedule

- classes > schedule_item
- classes_mail > schedule_item
- classes_attendance > schedule_item_attendance
- invoices_items_classes_attendance > schedule_item_attendance
- classes_otc > schedule_item_weekly_otc
- classes_otc_mail > schedule_item_weekly_otc
- classes_school_classcards_groups > schedule_item_organization_classpass_group
- classes_school_subscriptions_groups > schedule_item_organization_subscription_group
- classes_teachers > schedule_item_account

Staff schedule

- shifts > schedule_item
- shifts_otc > schedule_item_weekly_otc
- shifts_staff > schedule_item_account

Events / workshops

- workshops > schedule_event
- workshops > schedule_event_media
- workshops_mail > schedule_event
- workshops_activities > schedule_item
- workshop_activities_customers > schedule_item_attendance
- workshops_products > schedule_event_ticket
- workshops_products_activities > schedule_event_ticket_schedule_item
- workshops_products_customers > account_schedule_event_ticket

Announcements

- announcements > organization_announcements
- customers_profile_announcements > organization_announcements

4.3 Import data

OpenStudio data can be imported using the *openstudio_import* management command.

```
./manage.py openstudio_import --db_name=<openstudio> --db_user=<user> --db_password=
↪<password> --db_host=<openstudio db server> --os_uploads_folder=<path/to/web2py/
↪applications/openstudio/uploads>
```

4.4 Review import log

After the import a new log file containing import errors (if any) will be available in the logs directory in the Costasiella application root folder.

CHAPTER

FIVE

OPENSTUDIO MIGRATION

This page contains a list of things to keep in mind when migrating from OpenStudio by importing OpenStudio data.

5.1 For customers

- In case the address changes, users should update their bookmarks.
- Passwords can't be imported. Users should follow the password reset procedure to set a password.
- In Costasiella the first page customers will see is the shop page. In OpenStudio this is the log in page.
- A user can view their account details under the "My account" link in the shop once signed in.

5.2 For system administrators & studio staff

- In case the hosting address has changed, any links to it have to be updated. Eg. on the studio's website.
- The OpenStudio import page contains a list of items that can't be imported. These have to be added manually.

CHAPTER

SIX

SETUP DEVELOPMENT ENVIRONMENT

This page will explain how to setup a development environment for Costasiella. A Linux development environment based on Ubuntu 20.04 is presumed in this guide.

6.1 Preparation

Sign up for reCAPTCHA and create keys that apply to the localhost domain. https://www.google.com/recaptcha/about/

6.2 Required packages

Install required packages

```
sudo apt install curl git mysql-server python3-mysqldb libmysqlclient-dev redis-server␣
↪redis-tools python3-pip libffi-dev
sudo pip install virtualenvwrapper
```

6.3 MySQL database & Redis server

```
sudo mysql
CREATE DATABASE costasiella CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
CREATE DATABASE vault CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
create user 'user'@'localhost' identified by 'password';
grant all privileges on costasiella.* to 'user'@'localhost';
grant all privileges on vault.* to 'user'@'localhost';
flush privileges;
```

6.4 Vault setup

Download Vault from https://www.vaultproject.io

Create an empty file called vault_config_dev.hcl (or anything to your liking) and put the following in it:

```
ui = true
disable_mlock = true

storage "mysql" {
username = "user"
password = "password"
database = "vault"
}

listener "tcp" {
address     = "127.0.0.1:8200"
tls_disable = true
}
```

Edit the storage section with your database information.

Start the Vault server using

```
vault server -config=/path/to/config/config_dev.hcl
```

Open a webbrowser and go to http://localhost:8200/ui to perform an initial setup for Vault. For key shares and key threshold, fill in a 1. Though note for production something like 5 and 3 would be more appropriate.

Important

Save the root token and key somewhere safe. When these keys are lost, you lose all access to Vault and the encrypted data.

Continue by clicking the "Unseal" button and enter the key from the previous step (not the root token). Then sign in to vault using the root token.

Set up the transit engine Click "Enable new engine" and choose Transit under generic. Enter a recognizable name for the Path, in this document the default name "transit" will be used for the transit engine path. Click enable engine on the bottom of the page.

Click "Create encryption key" and enter a recognizable name. Then click the Create entryption key button. In this document *costasiella* will be used as the name for the encryption key.

In case you'd like to do further vault setup (at some point) using the terminal; add the following lines to the .bashrc or .zshrc file in your home directory. Also type the command in your current shell if you need to be able to execute vault commands without restarting your current shell session.

```
# Vault config
complete -C /usr/local/bin/vault vault
export VAULT_ADDR=http://127.0.0.1:8200
export VAULT_TOKEN=<Your root token here, definitely bad idea for production, but fine␣
↪for development>
export VAULT_TRANSIT_KEY=costasiella
```

6.5 Fetch source from git

Go to the folder where you want to keep the Costasiella source files

```
git clone https://github.com/costasiella/costasiella.git
git clone https://github.com/costasiella/frontend.git
```

6.6 NPM

```
curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash -
sudo apt install nodejs
cd frontend
# install node modules
npm install --legacy-peer-deps
```

6.7 Python virtual environment

Add the following lines to the .bashrc or .zshrc file in your home directory

```
# virtualenvwrapper stuff
export WORKON_HOME=$HOME/Development/virtualenvs
export PROJECT_HOME=$HOME/Development
export VIRTUALENVWRAPPER_SCRIPT=/usr/local/bin/virtualenvwrapper.sh
source /usr/local/bin/virtualenvwrapper_lazy.sh
```

Open a new terminal to start the development environment, to make sure .bashrc or .zshrc is reloaded.

Create a new virtual environment and install required python modules

```
mkvirtualenv cs_dev -p /usr/bin/python3
cd <your costasiella root dir>/
pip install -r requirements.txt
```

6.8 Django settings

Go to your costasiella root dir/app/app and edit settings/common.py

- Edit the databases section as required
- Under the vault configuration section edit the following setting to reflect your environment

```
...

RECAPTCHA_PUBLIC_KEY = '<Your site key here>'
RECAPTCHA_PRIVATE_KEY = '<Your secret key here>'

...
```

6.9 Prepare for lift off

Init database; create admin user; start django (back-end) development server. Go to <your costasiella root dir>/app (this folder should contain a file called manage.py).

```
./manage.py migrate
./manage.py createsuperuser
# fill out questions to create initial super admin user
./manage.py loaddata costasiella/fixtures/*.json
./manage.py runserver
```

Start the npm development server; Open a new terminal tab or window and go your costasiella frontend root dir.

```
npm run start
```

A webbrowser will open to localhost:3000. There's a proxy that'll allow access to some django pages using the /d path in the address. eg. http://localhost:3000/d/admin

The Django development server runs on port 8000 in case you'd like to access it directly.

Apply any database updates that might be available

Open a browser and go to http://localhost:3000/d/update

6.10 Configure the superuser account as a Costasiella admin

Open a webbrowser (tab) and go to <your domain>/d/admin. Log in using the initial superuser credentials created earlier.

Navigate to Costasiella > Accounts and click the email address of the superuser. Now add the user to the Admins group and click save.

Run the following code in a mysql terminal with a user that has permissions to modify your Costasiella database.

```
use costasiella;
update costasiella_account set employee=1 where id=1;
```

This enables the superuser to sign in to the backend with admin privileges.

Note: The superuser isn't created in the "regular" way. It doesn't have records in all the tables that regular accounts have. It's highly recommened to use an account created under relations > accounts that's been granted admin privileges to test things.

Done, the superuser created can now sign in to both the frontend and backend.

6.11 GraphiQL

The GraphiQL interface is available at http://localhost:8000/d/graphql

CHAPTER

SEVEN

TINYMCE SETUP

7.1 Hosting

TinyMCE version 5.8.2 has been added to the assets folder in the backend. The url it can be accessed on is /d/static/tinymce/tinymce.min.js

7.2 React

The TinyMCE react module is used (npm install @tinymce/tinymce-react)

In the frontend an Editor component can be instructed to load it like so:

```
<Editor
    tinymceScriptSrc="/d/static/tinymce/tinymce.min.js"
    ...
/>
```

CHAPTER

EIGHT

DJANGO GRAPHENE QUERY ORDERING

8.1 Summary

The django-graphene docus on filtering & ordering don't seem completely up to date. (https://docs.graphene-python.org/projects/django/en/latest/filtering/) A quick reference below on how it does work.

8.2 React

The TinyMCE react module is used (npm install @tinymce/tinymce-react)

In the frontend an Editor component can be instructed to load it like so:

```
from django_filters import FilterSet, OrderingFilter


# For a single field
class ScheduleEventTicketScheduleItemFilter(FilterSet):
    class Meta:
        model = ScheduleEventTicketScheduleItem
        fields = ['schedule_event_ticket', 'schedule_item', 'included']

    order_by = OrderingFilter(
        fields=(
            ('schedule_item__date_start', 'date_start'), # order by field, display name␣
↪(used in query)
        )
    )
# Single field end

# For multiple fields
class CustomOrderingFilter(OrderingFilter):
"""
A Custom ordering filter to allow ordering by multiple fields
"""
def __init__(self, *args, **kwargs):
    super().__init__(*args, **kwargs)
    self.extra['choices'] += [
        ('date_start', 'Date start'),
        ('-date_start', 'Date start (descending)'),
```

(continues on next page)

(continued from previous page)

```
    ]

def filter(self, qs, value):
    # OrderingFilter is CSV-based, so `value` is a list
    if any(v in ['date_start', '-date)start'] for v in value):
        # sort queryset by relevance
        return qs.order_by('schedule_item__date_start', 'schedule_item__time_start')

    return super().filter(qs, value)


class ScheduleEventTicketScheduleItemFilter(FilterSet):
    class Meta:
        model = ScheduleEventTicketScheduleItem
        fields = ['schedule_event_ticket', 'schedule_item', 'included']

    order_by = CustomOrderingFilter()

# Multiple fields end


# The setting of filterset_class is the same in all cases
class ScheduleEventTicketScheduleItemNode(DjangoObjectType):
    class Meta:
        model = ScheduleEventTicketScheduleItem
        filterset_class = ScheduleEventTicketScheduleItemFilter
        interfaces = (graphene.relay.Node,)

    @classmethod
    def get_node(self, info, id):
        user = info.context.user
        require_login_and_permission(user, 'costasiella.view_scheduleeventscheduleitem')

        return self._meta.model.objects.get(id=id)
```

Query example of how we can now use orderBy in a subQuery

```
import { gql } from "@apollo/client"


export const GET_SCHEDULE_EVENT_TICKET_QUERY = gql`
    query ScheduleEventTicket($id: ID!) {
        scheduleEventTicket(id:$id) {
        id
        name
        price
        priceDisplay
        totalPrice
        totalPriceDisplay
        description
        isSoldOut
        isEarlybirdPrice
```

(continues on next page)

(continued from previous page)

```
        ticketScheduleItems(included: true, orderBy: "scheduleItem_DateStart") {
            pageInfo {
                hasNextPage
                hasPreviousPage
                startCursor
                endCursor
            }
            edges {
                node {
                    id
                    included
                    scheduleItem {
                    name
                    dateStart
                    timeStart
                    timeEnd
                    organizationLocationRoom {
                        organizationLocation {
                            name
                            }
                        }
                    }
                }
            }
        }
            scheduleEvent {
                id
                name
            }
        }
    }
`
```