

Modality Documentation

Table of Contents

Introduction	2
What is Modality?	2
Why the name?	3
Development progress	3
Open-source	3
The Modality Architecture	3
Java for the Web	3
Technologies	3
Layers	4
All-Layer Aggregation	6
The Modality Apps	6
Web apps	6
Mobile apps	6
Desktop apps	6
Installation	6
1. Install Java JDK	6
2. Install WebFX CLI	7
3. Install IntelliJ IDEA	7
4. Install Docker	7
5. Install Git	7
6. Create the Modality root	7
7. Clone the codebase	7
8. Unzip the development database	8
9. Prepare Docker environment variables	8
10. Build the Docker containers	8
11. Start the containers to build the database	8
Configure Modality for Development	8
Resolve all Modality Dependencies	8
Modality Server	9
Back-Office Desktop	11
Front-Office Desktop	11
Build & Run Modality on Development	11
Docker *	12
Modality Server *	12
Back-Office Desktop	12
Back-Office Web	17

Back-Office Mobile	19
Front-Office Desktop	19
Front-Office Web	19
Front-Office Mobile	19
Modality Database	20
Modality Session	20
Using Docker	20
Connect to the Docker database container	21
Connect to the Docker session container	21
Shut down Docker	21
Prune Docker content	21
Deploy Modality to Heroku	22

Introduction

What is Modality?

Modality is a free, open-source, hospitality-oriented booking system, currently in development. It will support significant event complexity, unmatched by any existing open or closed-source solution.

Modality will support:

- short events (1 evening, 1 day, 1 weekend,...)
- long events (1 month, 6 weeks,...)
- recurring events (daily, weekly, monthly,...)
- full attendance, or partial attendance
- multiple venues
- translation option for international events
- multiple accommodation locations (including onsite accommodation and partner hotels)
- multiple accommodation types (potentially specific to each location)
- shuttles (for example: airport shuttles at beginning and end of events)
- daily shuttles (transport from hotels to the venues)
- parking for different vehicle types
- catering options: multiple meals (breakfast, lunch, supper,...) with multiple diet options (vegetarian, vegan,...) and multiple sittings (first sitting, second sitting,...)
- individual allocation (rooms, sittings,...)
- bulk allocation (with rules to be entered)
- long stays (staff, visitors, residents,...)

- registration features (instant search, filters, modification, cancellation, confirmation,...)
- individual email notifications
- bulk email (newsletters, information letters,...)
- hot target emailing (triggered when specific booking options matched, booking state changes,...)
- multi-stage variable payments (for example: 25% min deposit, then 50% after a cutoff date,...)
- discounts (children, early booking,...)
- detailed statistics

Modality has the potential to support the following:

- event-oriented organisations
- hotels
- shops (time-limited or permanent, with online option)
- cafes/restaurants (time-limited or permanent, with online option)

Why the name?

'Modality' is a fusion of two words, 'modular' and 'hospitality', a nod to both the software's design and purpose.

Development progress

Modality is in the very early stages of active development. Whilst the server and data models are complete, the clients have significant work remaining before they are viewable. Currently only the Back-Office Desktop client (described below) can be compiled and run successfully from the IDE.

Open-source

Modality is licensed under the [Apache License Version 2.0](#), is completely free to use and modify, and is available on [GitHub](#).

The Modality Architecture

Java for the Web

Modality is the first large-scale Java project to use [WebFX](#) - a toolkit that transpiles JavaFX applications into pure JavaScript web apps for direct execution in the browser.

Technologies

Modality is developed using the following technologies:

Technology	Purpose	Version
Java	Codebase	18
JavaFX	Desktop + mobile user interfaces	18
WebFX	Web user interfaces	Latest (Beta)

...consists of four end-user client applications:

Application	Used By	Compilation Toolchain
Back-Office Web	Administrator	WebFX + GWT
Back-Office Mobile	Administrator	Gluon
Front-Office Web	Customer	WebFX + GWT
Front-Office Mobile	Customer	Gluon

...two developer client applications:

Application	Used By	Compilation Toolchain
Back-Office Desktop	Developer	JavaFX
Front-Office Desktop	Developer	JavaFX

...one web server:

Application	Purpose	Version
Vert.x	Interface between client apps and back-end services; serves the SPA	Latest

...and depends on the following services:

Service	Purpose	Version
Postgres	Database	14.2
Redis	Session management	6.2.6
Flyway	Database schema updates	Latest

The services are orchestrated by [Docker](#) when running Modality on development machines (instructions given later in this document).

Layers

Modality is divided into layers of functionality, shown below:

Layer	Repository	Java Modules
Business Logic (top layer)	modality	modality-event, modality-hotel, modality-restaurant, modality-catering
Ecommerce	modality	modality-ecommerce
CRM	modality	modality-crm
Base	modality	modality-base
WebFX Stack	webfx-stack	webfx-stack
WebFX (bottom layer)	webfx	webfx-kit

Business logic

The highest layer of the architecture consists of business-specific modules implementing logic for events, hotels, restaurants etc. This is a customisable layer, and developers can choose to add only the modules they need, as well as provide their own.

Ecommerce

The next layer down is the ecommerce layer. This provides a generic domain model for ecommerce, which models sales, accounts etc. It is the location for payment gateway integration and ecommerce-specific UIs.

CRM

The CRM layer provides the essential CRM features, including customer accounts, integrated mailing system etc.

Base

The Base layer is a fully operational implementation of the WebFX Stack layer beneath, based on the Postgres database. This layer is a pure technical solution that isn't bound to any specific domain, and so is large in application scope.

WebFX Stack

The WebFX Stack layer provides an opinionated framework for developing enterprise applications with WebFX. This layer is responsible for communication between client and server (using a WebSocket bus), UI routing, ORM, push notification, auth, i18n, etc. Interfaces in all cases, but not always full implementations, allowing this layer to be adapted to any kind of system.

Unlike most Java frameworks, this layer works principally on the client side, following the trend initiated by mobile apps where most of the application code has been moved to the client and can run offline.

It is designed to work with JavaFX (for example, i18n provides JavaFX bindings for use with any kind of control; and the authorisation framework automatically enables/disables and shows/hides

controls depending on user access).

WebFX

WebFX is the foundation layer, providing a web port of JavaFX (in the `webfx-kit` module) that can be compiled by GWT together with your application code. It is a Java-based cross-platform solution that can be used in any domain.

All-Layer Aggregation

Modality ships with the `modality-all` module, which aggregates together the full set of modules across all layers, for use by developers right away.

The Modality Apps

Web apps

Modality uses WebFX to transpile its JavaFX codebase into a single-page application for direct execution in the browser. No server-side rendering, and no plugins required.

Mobile apps

Modality uses the [Gluon](#) toolchain to compile the codebase into native, installable apps ready for inclusion into the Google Play and Apple App stores.

Desktop apps

Modality also provides desktop apps, which have exactly the same UI as the web apps generated from the same source. This is useful for developers, allowing Java code to be rapidly developed and tested via the desktop, before subsequent transpilation into JavaScript and mobile (which takes time).

Installation

1. Install Java JDK

Modality is developed entirely in the Java language, and requires at least JDK 17+. Check whether this is installed:

```
java --version
```

If it is not installed, or is an older version, please refer to [this guide](#).

2. Install WebFX CLI

We use the WebFX CLI to compile Modality for the web. Please follow this [guide](#) to install it.

3. Install IntelliJ IDEA

We develop Modality using the free, community edition of [IntelliJ IDEA](#), and recommend you install this if you do not already have an IDE. IntelliJ allows you to easily compile and run the Modality server and clients, for the purpose of local development and testing.



All subsequent IDE-based examples given in this documentation will be based on IntelliJ.

4. Install Docker

During development, Modality uses [Docker](#) for all external services, including the database and the in-memory datastore for sessions.

Please install [Docker Desktop](#) on your local machine if you do not have it already. If using a Mac, the easiest way is to install using [brew](#). Please provide Docker with a minimum of 8GB of RAM, ideally more.



Insufficient RAM may result in `java.lang.OutOfMemoryError` errors when importing the [modality-dev-db](#).

5. Install Git

A git client is needed to retrieve the Modality codebase from GitHub. Check if git is installed:

```
git --version
```

If it is not installed, you may wish to refer to [this guide](#).

6. Create the Modality root

```
mkdir -vp modality
export MODALITY_ROOT=${PWD}/modality
```

7. Clone the codebase

Git clone the Modality codebase via the terminal (or IntelliJ etc):

```
cd $MODALITY_ROOT  
git clone https://github.com/modalityone/modality.git .
```

8. Unzip the development database

The Modality codebase includes a 15Mb zip file of example data, that makes it easier for developers to navigate around the running system. Execute the following commands to unzip the database, ready for later ingestion into Postgres:

```
cd $MODALITY_ROOT/modality-base/modality-base-server-datasource/src/main/resources  
unzip db-dev/V0001__modality_dev_db.sql.zip -x '__MACOSX/*' -d db/
```

9. Prepare Docker environment variables

Environment variables store the Postgres database name, username and password. Defaults are provided in the [.env-template](#). Use this template file as the basis for your Docker-based configuration, by creating an [.env](#) file from it. You may leave the defaults, or provide new values accordingly:

```
cd $MODALITY_ROOT/docker  
cp .env-template .env  
source .env # make the environment variables available to the shell
```

10. Build the Docker containers

```
cd $MODALITY_ROOT/docker  
docker-compose build --no-cache
```

11. Start the containers to build the database

```
cd $MODALITY_ROOT/docker  
docker-compose up
```

Configure Modality for Development

Resolve all Modality Dependencies

Open Modality in IntelliJ and wait for all dependencies to be automatically resolved:

```

1  # Modality
2
3  Modality is a free, open-source, hospitality-oriented booking system, currently in development. It will support significant event complexity, unmatched
4  by any existing open or closed-source solution.
5
6  Modality will support:
7
8  * short events (1 evening, 1 day, 1 weekend,...)
9  * long events (1 month, 6 weeks,...)
10 * recurring events (daily, weekly, monthly,...)
11 * full attendance, or partial attendance
12 * multiple venues
13 * translation option for international events
14 * multiple accommodation locations (including onsite accommodation and partner hotels)
15 * multiple accommodation types (potentially specific to each location)
16 * shuttles (for example: airport shuttles at beginning and end of events)
17 * daily shuttles (transport from hotels to the venues)
18 * parking for different vehicle types
19 * catering options: multiple meals (breakfast, lunch, supper,...) with multiple diet options (vegetarian, vegan,...) and multiple sittings (first
20   sitting, second sitting,...)
21 * individual allocation (rooms, sittings,...)
22 * bulk allocation (with rules to be entered)
23 * long stays (staff, visitors, residents,...)
24 * registration features (instant search, filters, modification, cancellation, confirmation,...)
25 * individual email notifications
26 * bulk email (newsletters, information letters,...)
27 * hot target emailing (triggered when specific booking options matched, booking state changes,...)
28 * multi-stage variable payments (for example: 25% min deposit, then 50% after a cutoff date,...)
29 * discounts (children, early booking,...)
30 * detailed statistics
31
32 Modality has the potential to support the following:
33
34 * event-oriented organisations
35 * hotels
36 * shops (time-limited or permanent, with online option)
37 * cafes/restaurants (time-limited or permanent, with online option)
38
39 The Modality server can support single organisations, multiple different organisations, and parent organisations comprising many child/spin-off
40   organisations (each organizing their own events)
41
42 ## Why the name?
43 'Modality' is a fusion of two words, 'modular' and 'hospitality', a nod to both the software's design and purpose.
44
45 ## Key Technology Features
46 * developed entirely in Java, on both the server-side and client-side
47 * uses the same UI toolkit (JavaFX) across all clients apps
48 * native web, mobile and desktop clients generated from the same codebase
49 * server powered by Vert.x (https://vertx.io/)
50
51 ## Powered by WebFX
52

```

Modality

Modality is a free, open-source, hospitality-oriented booking system, currently in development. It will support significant event complexity, unmatched by any existing open or closed-source solution.

Modality will support:

- short events (1 evening, 1 day, 1 weekend,...)
- long events (1 month, 6 weeks,...)
- recurring events (daily, weekly, monthly,...)
- full attendance, or partial attendance
- multiple venues
- translation option for international events
- multiple accommodation locations (including onsite accommodation and partner hotels)
- multiple accommodation types (potentially specific to each location)
- shuttles (for example: airport shuttles at beginning and end of events)
- parking for different vehicle types
- catering options: multiple meals (breakfast, lunch, supper,...) with multiple diet options (vegetarian, vegan,...) and multiple sittings (first sitting, second sitting,...)
- individual allocation (rooms, sittings,...)
- bulk allocation (with rules to be entered)
- long stays (staff, visitors, residents,...)
- registration features (instant search, filters, modification, cancellation, confirmation,...)
- individual email notifications
- bulk email (newsletters, information letters,...)
- hot target emailing (triggered when specific booking options matched, booking state changes,...)
- multi-stage variable payments (for example: 25% min deposit, then 50% after a cutoff date,...)
- discounts (children, early booking,...)
- detailed statistics

Modality has the potential to support the following:

- event-oriented organisations
- hotels
- shops (time-limited or permanent, with online option)
- cafes/restaurants (time-limited or permanent, with online option)

Modality is entirely modular. Choose all the modules, or just the modules you want (even the events module is optional).

Modality is a fusion of two words, 'modular' and 'hospitality', a nod to both the software's design and purpose.

Modality is entirely modular. Choose all the modules, or just the events module is optional).

Why the name?

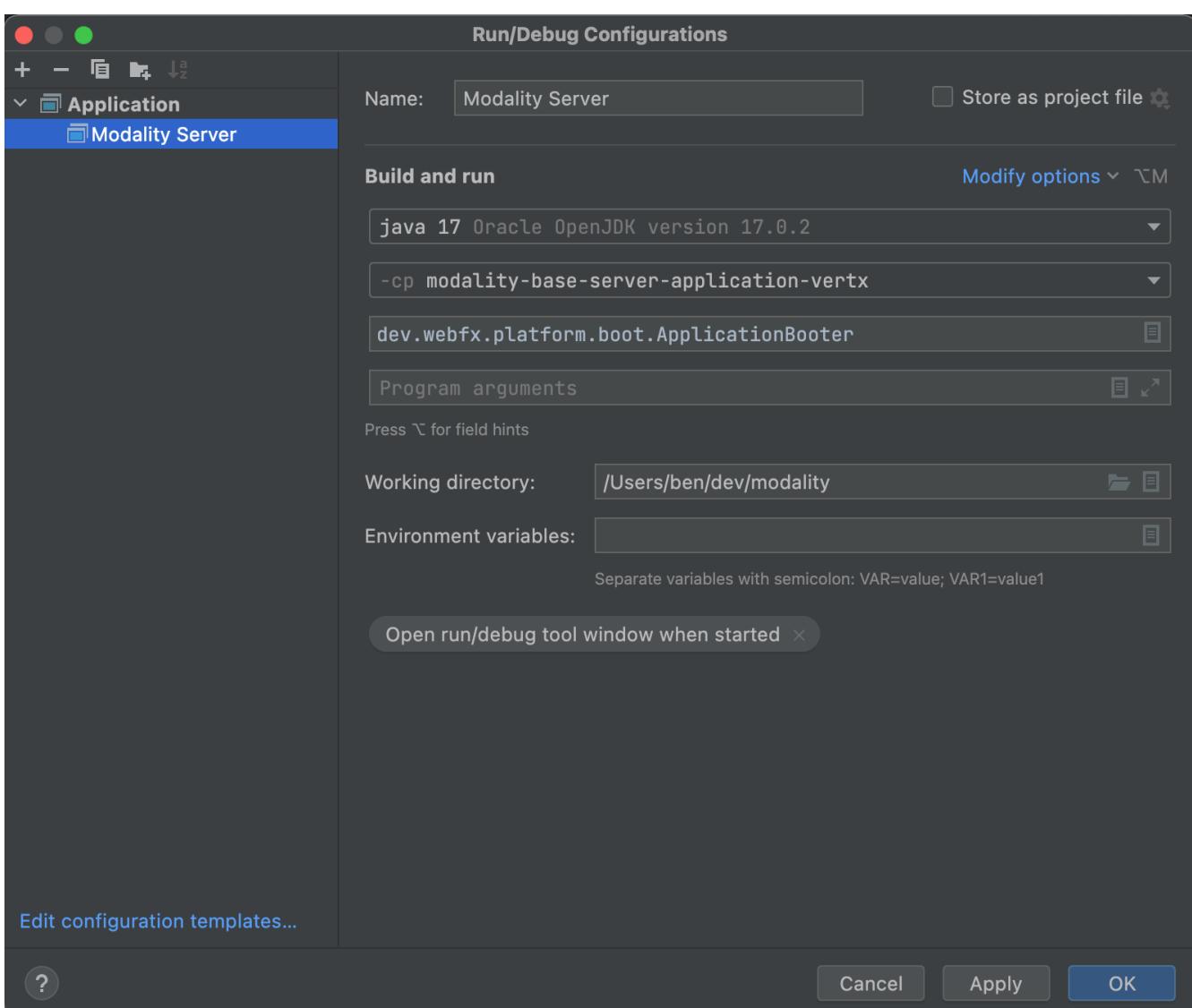
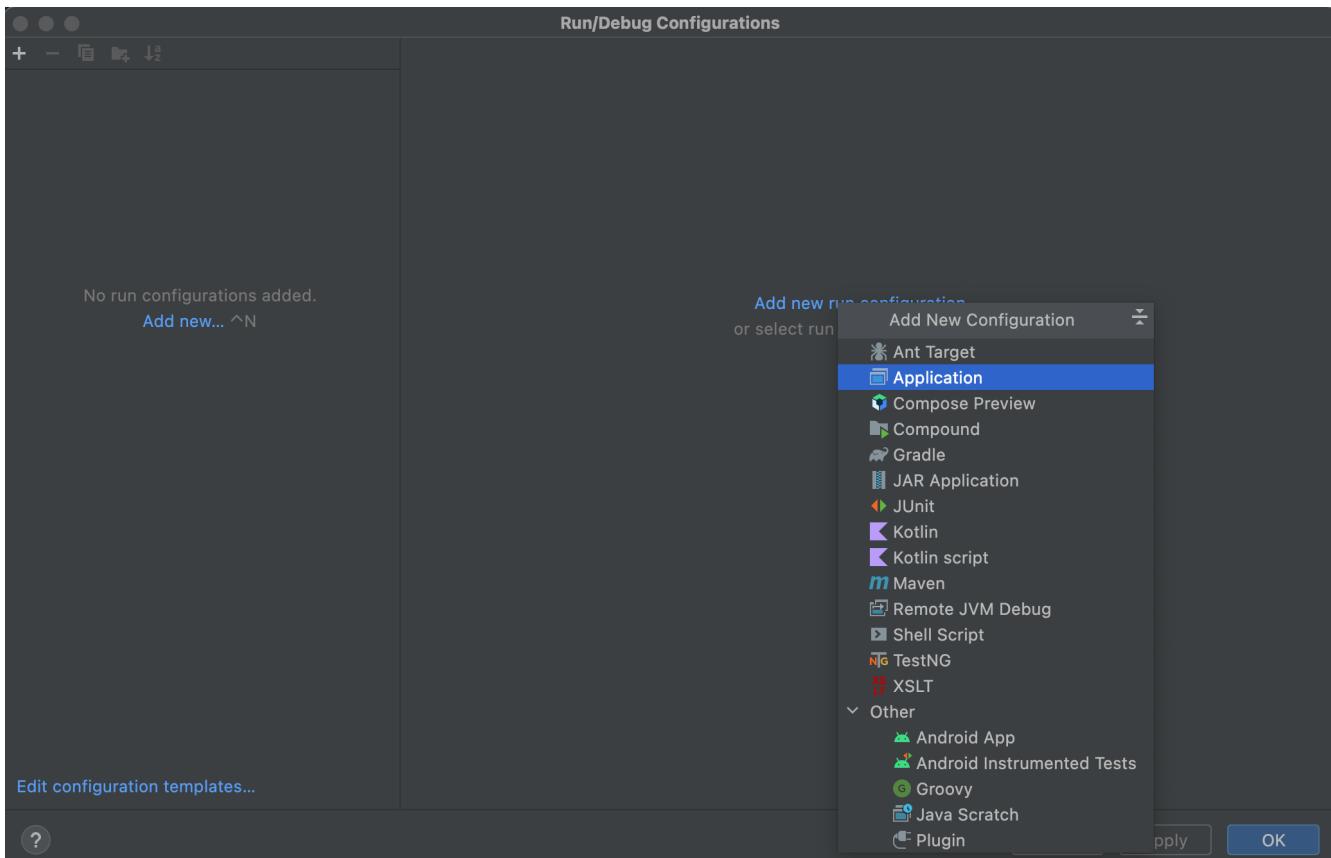
'Modality' is a fusion of two words, 'modular' and 'hospitality', a nod to both the software's design and purpose.

Modality Server

In order to run any of the Modality client applications, the Modality Server should first be running. The Modality Server is a [Vert.x](#) server that proxies requests to the database and is responsible for establishing and maintaining user sessions.

The easiest way to stand up the server locally is to create an application run configuration in the IDE.

In the IntelliJ menu, click [Run → Edit Configurations](#) to display the following dialog, and populate with the same details:

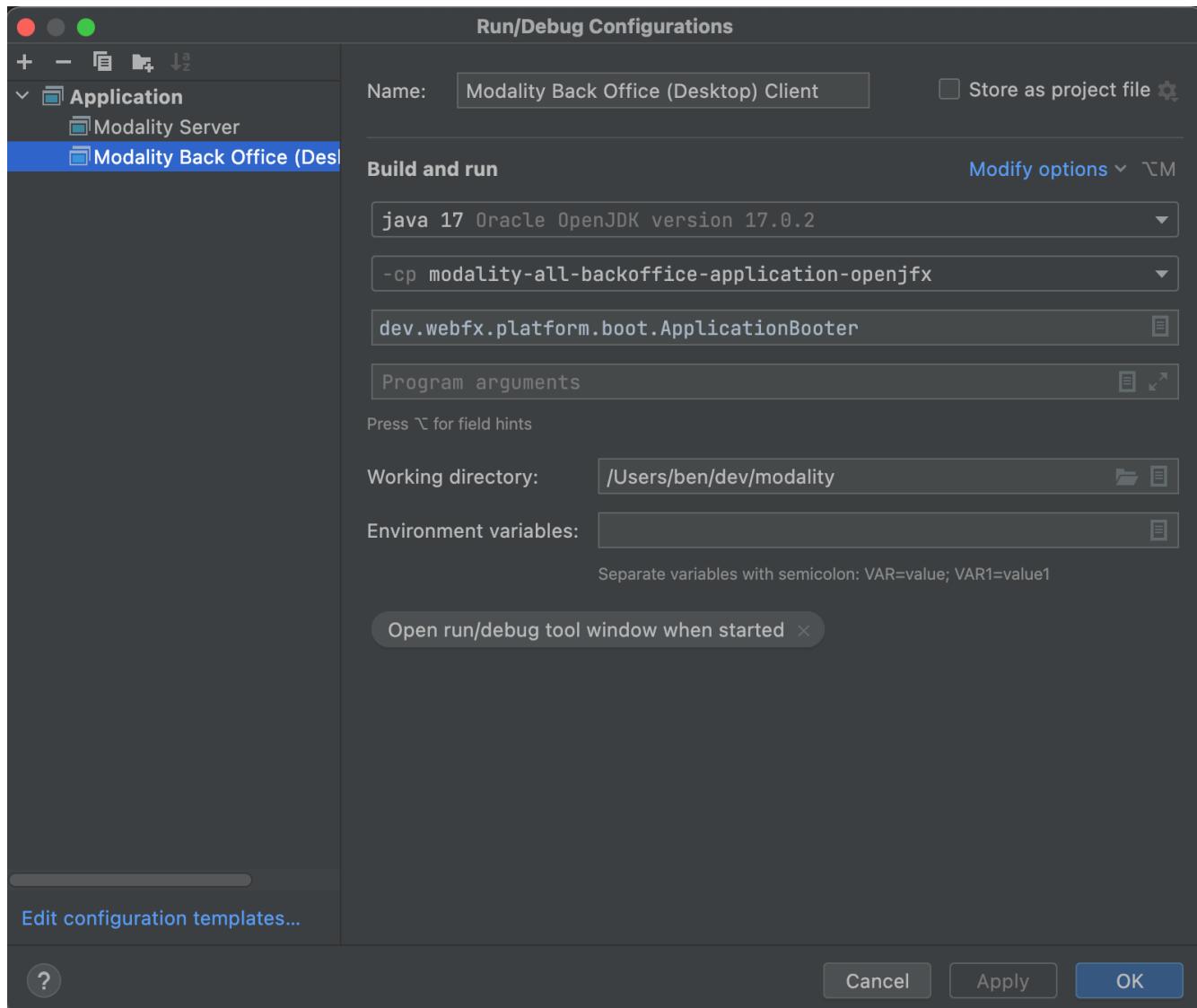


Click 'OK' to save the configuration and close the dialog.

Back-Office Desktop

The Back-Office Desktop client is an application used by developers of Modality, and emulates the web user interface used by administrators of the system.

Create another run configuration and populate it with the details given in the screenshot below:



Click 'OK' to save the configuration and close the dialog.

Front-Office Desktop



The Front-Office Desktop client is not yet implemented.

Build & Run Modality on Development

The Modality clients run independently of each other, but all require the Modality Server to be running, which in turn requires Docker to be running the service containers described above.

Therefore, the first two steps below are mandatory before running one or more of the Modality clients locally.

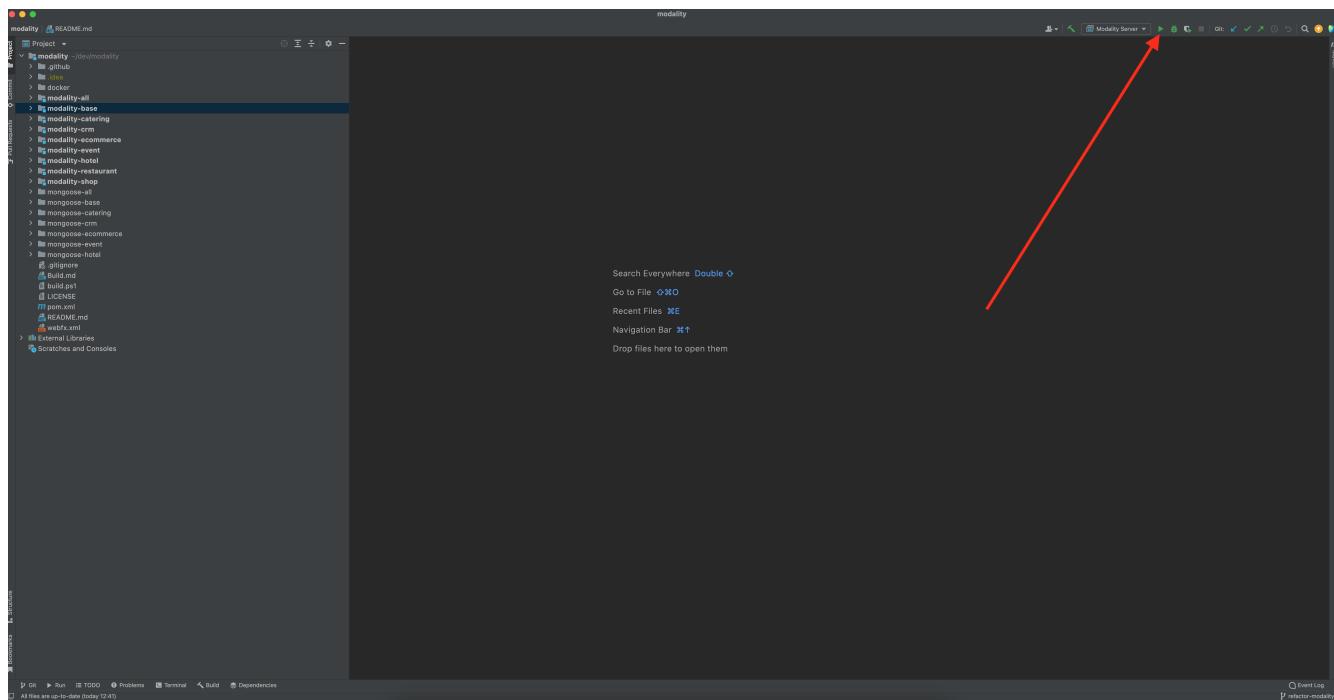
Docker *

Ensure that Docker is running:

```
cd $MODALITY_ROOT/docker  
docker-compose up
```

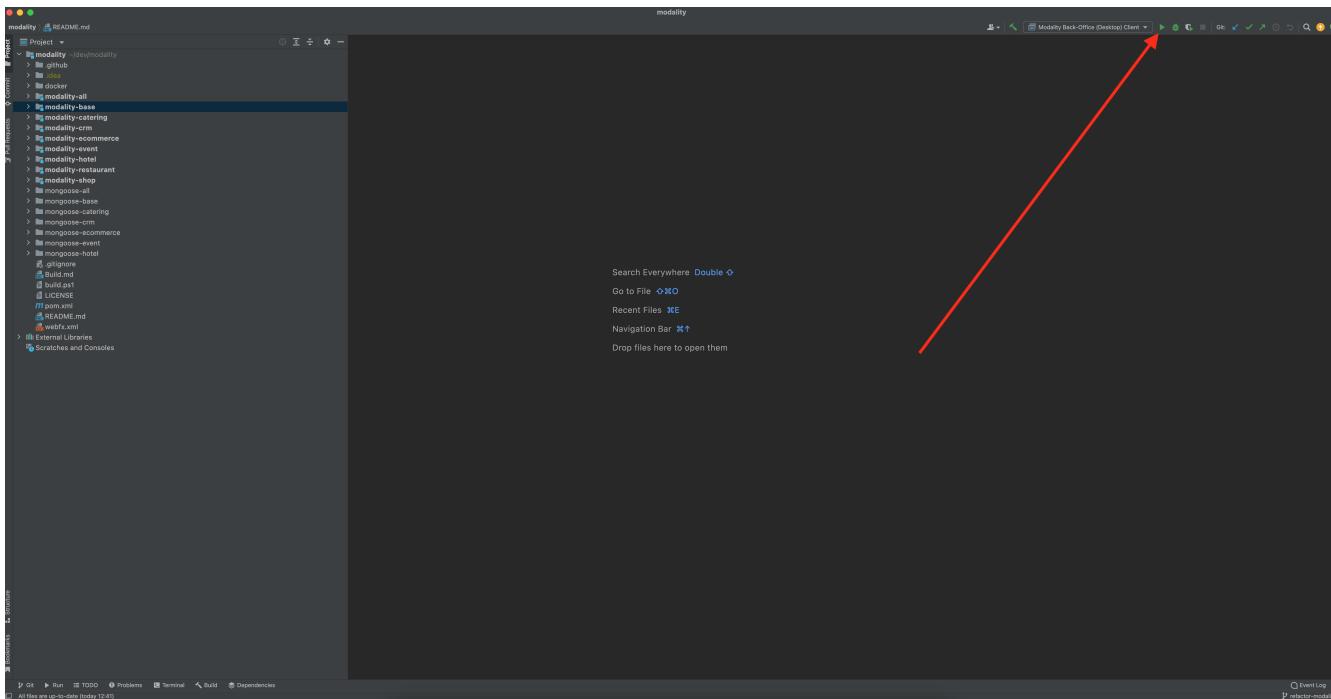
Modality Server *

Build and run the server by executing its run configuration:

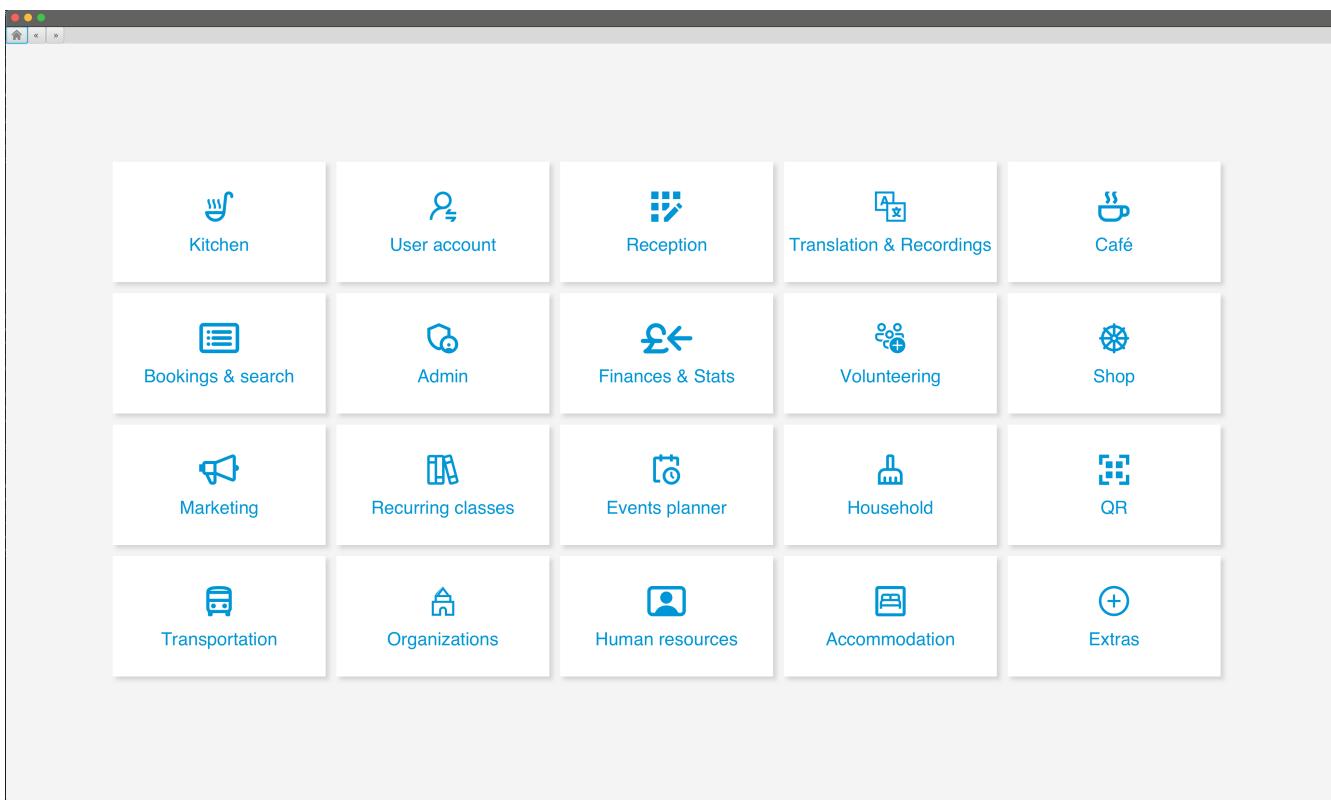


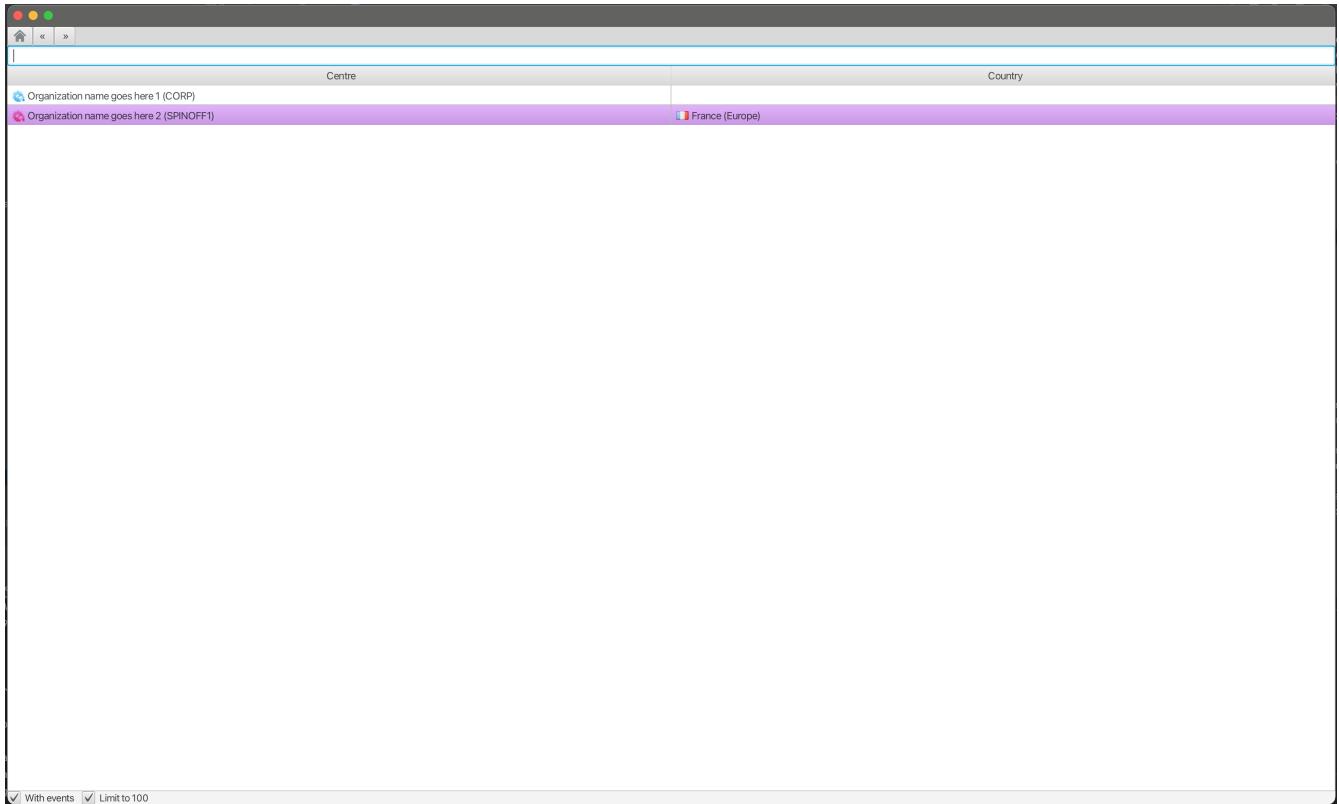
Back-Office Desktop

Build and run the Back-Office Desktop client by executing its configuration:



The Back-Office Desktop client will then be ready to use:





Event	type
Event name goes here 933 ~ 10-16 june	Event type 35
Event name goes here 928 ~ 10-16 june	Event type 35
Event name goes here 589 ~ 26 november - 2 december 2021	Event type 37
Event name goes here 821 ~ 22-30 october 2021	Event type 36
Event name goes here 820 ~ 22-28 october 2021	Event type 36
Event name goes here 735 ~ 30 july - 16 august 2021	Event type 36
Event name goes here 734 ~ 30 july - 14 august 2021	Event type 36

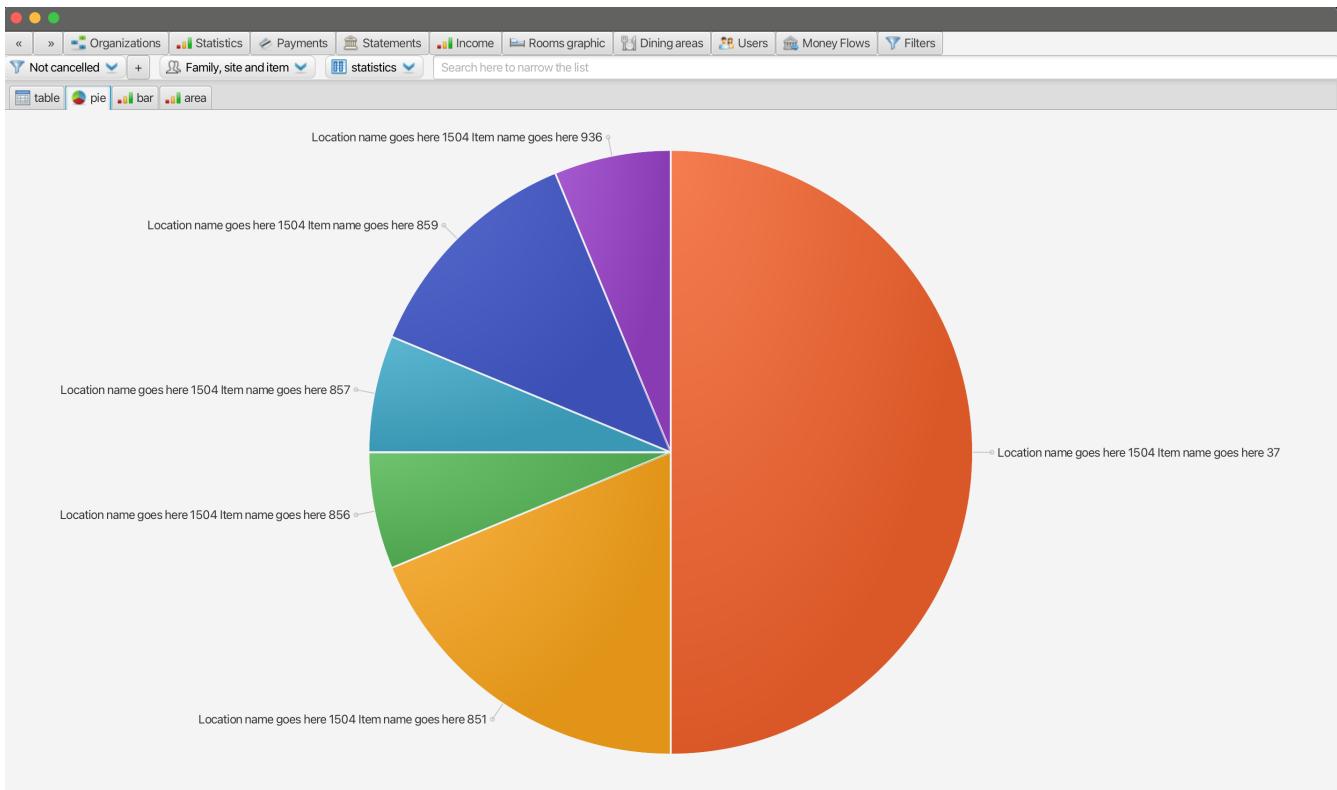
Not cancelled + Search here to narrow the list

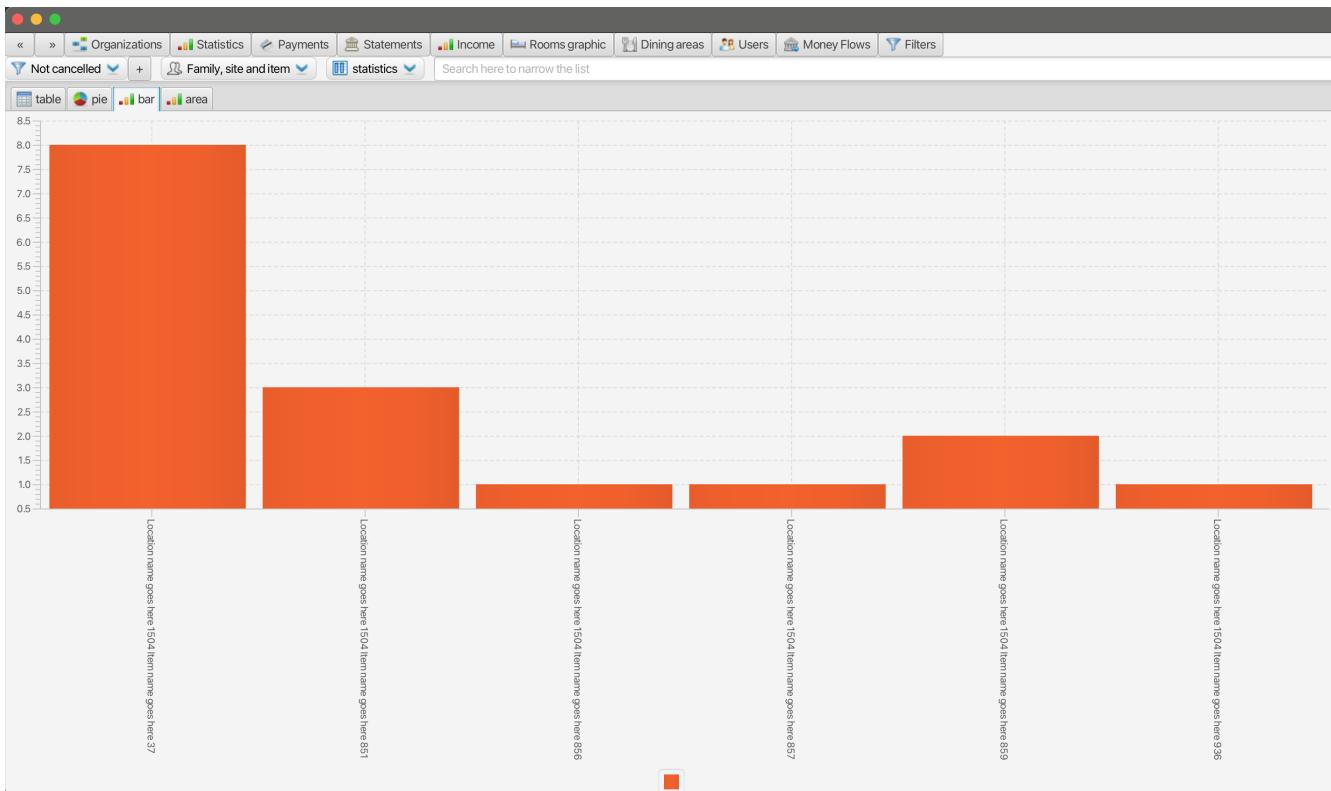
ref	Firstname	Lastname	Age	Invoiced	min deposit	Deposit	Balance
3842	Firstname	Lastname		115.00	115.00	115.00	0.00
3841	Firstname	Lastname		133.00	133.00	133.00	0.00
3840	Firstname	Lastname		115.00	115.00	115.00	0.00
3839	Firstname	Lastname		133.00	133.00	133.00	0.00
3838	Firstname	Lastname		133.00	133.00	133.00	0.00
3837	Firstname	Lastname		115.00	115.00	115.00	0.00
3836	Firstname	Lastname		133.00	133.00	133.00	0.00
3835	Firstname	Lastname		115.00	115.00	115.00	0.00
3834	Firstname	Lastname		115.00	115.00	115.00	0.00
3833	Firstname	Lastname		115.00	115.00	115.00	0.00
3832	Firstname	Lastname		133.00	133.00	133.00	0.00
3831	Firstname	Lastname		133.00	133.00	133.00	0.00
3830	Firstname	Lastname		115.00	115.00	115.00	0.00
3829	Firstname	Lastname		115.00	115.00	115.00	0.00

✓ Limit to 100

Personal details Options Payments Comments Cart Multiple bookings Family Mails History

Site	Item	Dates	Allocation	Special Request	Net	Non refundable	min deposit	Deposit
Location name goes ...	Item name goes here ...	24/10-30/10			115.00	115.00	115.00	115.00
Location name goes ...	Item name goes here ...				0.00	0.00	0.00	0.00
Location name goes ...	Item name goes here ...	24/10-30/10			0.00	0.00	0.00	0.00
Location name goes ...	Item name goes here ...	24/10-28/10			18.00	18.00	18.00	18.00





Flat payments Pending or successful

Date	Booking	Transaction ref	Gateway status	Special Request	Amount	P...	S...
20/03/2022 16:00:21	36 🇫🇷 🚪 Firtname Lastname	123456789			98.00	🌐	⚠️
17/03/2022 20:43:20		123456789			196.00	🌐	⚠️
17/03/2022 20:43:20	30 🇫🇷 🚪 Firtname Lastname	123456789			126.00	🌐	⚠️
17/03/2022 20:43:20	31 🇫🇷 🚪 Firtname Lastname	123456789			70.00	🌐	⚠️
16/03/2022 17:34:38	23 🇫🇷 🚪 Firtname Lastrame	123456789	Status value added here		98.00	🌐	✅

Limit to 100

No content in table

Back-Office Web

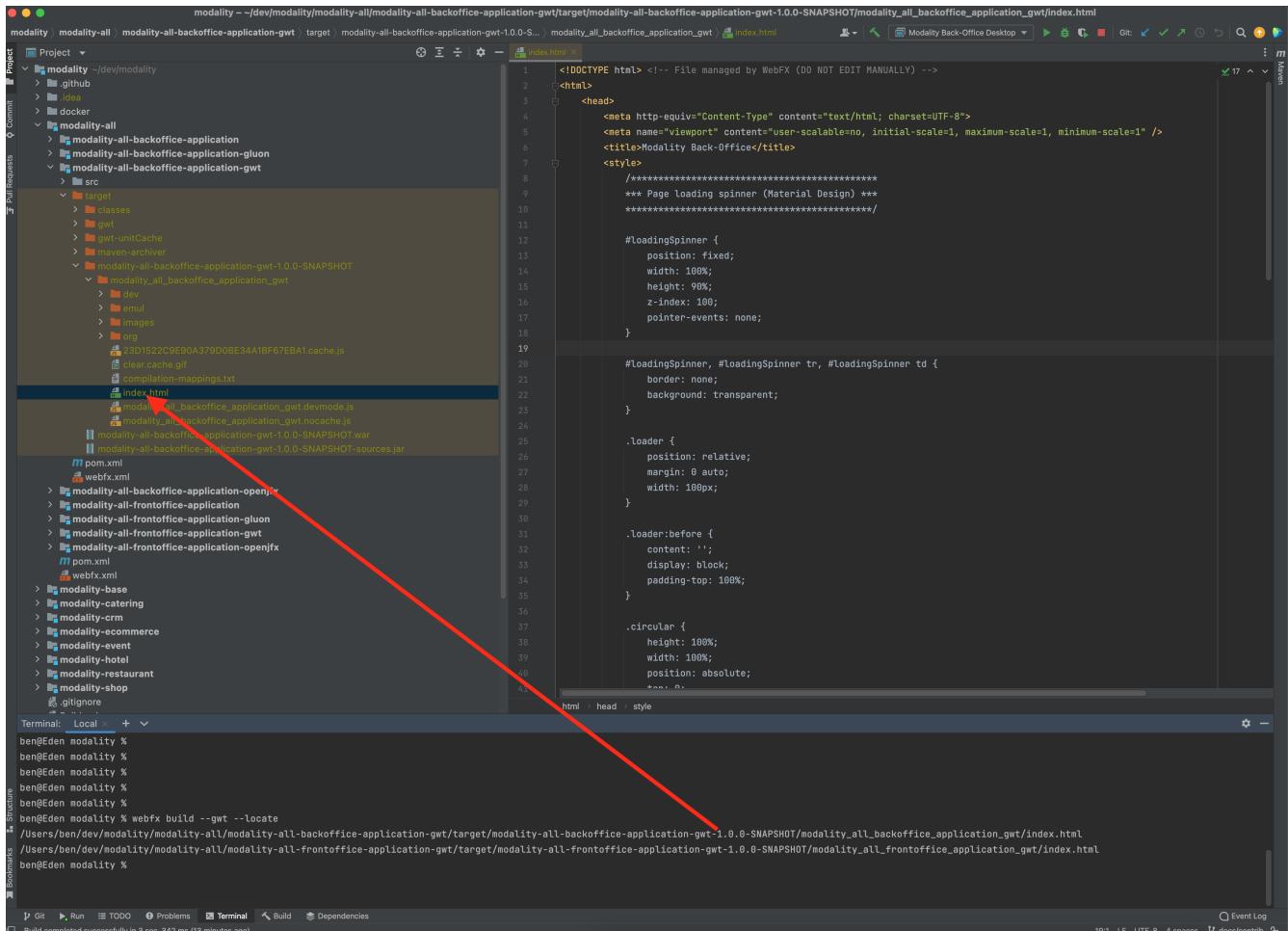
- ① First **build** the index.html file:

```
cd $MODALITY_ROOT
webfx build --gwt
```

- ② Then **locate** the resultant index.html file on the filesystem:

```
webfx build --gwt --locate
```

- ③ In the IntelliJ Project window, navigate to the index file and double click.



- ① Hover the mouse over the index.html source code to display the browser options.

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the project structure under 'modality_all' including 'modality-all-backoffice-application-gwt' and its sub-directories like 'src', 'target', and 'modality-all-backoffice-application-gwt-1.0.0-SNAPSHOT'.
- Code Editor:** Displays the 'index.html' file content. The code includes HTML, CSS, and JavaScript. A red arrow points to the browser icon in the top right corner of the code editor window.
- Terminal:** Shows the command line output of 'mvn webfx:build --gwt --locate'.
- Toolbar:** Includes icons for Git, Run, TODO, Problems, Terminal, Build, and Dependencies.

① Click on one of the browser icons to run the index.html in a browser.

Back-Office Mobile

Full instructions (and limitations) for compiling the codebase to native Android and iOS apps are available on the WebFX documentation site [here](#).

Front-Office Desktop



The Front-Office Desktop client is not yet implemented.

Front-Office Web



The Front-Office Web client is not yet implemented.

Front-Office Mobile



The Front-Office Mobile client is not yet implemented.

Modality Database

All database setup scripts are stored in the `$MODALITY_ROOT/modality-base/modality-base-server-datasource/src/main/resources/db-pristine/` folder, and are numbered in order of execution. They should be copied into the `$MODALITY_ROOT/modality-base/modality-base-server-datasource/src/main/resources/db/` deployment folder where they will then be executed by the Flyway container, which runs on Docker startup. All the data will then be stored on the host, in directory:

```
$MODALITY_ROOT/docker/data/postgres/*
```

This provides persistence, and the container can be safely shut down and restarted without losing data.

Any new database scripts must be:

- ① added to the `$MODALITY_ROOT/modality-base/modality-base-server-datasource/src/main/resources/db-pristine/` folder
- ② named according to the convention used in the folder: `V{latest-number+1}__{desc}.sql`
- ③ copied into the `$MODALITY_ROOT/modality-base/modality-base-server-datasource/src/main/resources/db/` deployment folder
- ④ executed by Flyway by restarting Docker:

```
cd $MODALITY_ROOT/modality-base/modality-base-server-datasource/src/main/resources/  
.  
# Create the new SQL file and copy to the deployment directory  
touch db-pristine/V0100__new_table.sql  
cp db-pristine/V0100__new_table.sql db/V0100__new_table.sql  
.  
# Now restart Docker  
cd $MODALITY_ROOT/docker  
docker-compose down  
docker-compose up
```

Modality Session

The session data is controlled by the docker-based Redis container and is not persisted locally. The data persists only as long as the container is running.

Using Docker

Connect to the Docker database container

Connection is easily made via any Postgres client (e.g. DBeaver). Use the following credentials (contained within the `docker/.env-template` file):

- Server: 127.0.0.1
- Port: 5432
- Database: modality
- User: modality
- Password: modality

Connect to the Docker session container

Connection can be made through the Docker terminal:

```
cd $MODALITY_ROOT/docker
docker exec -ti session /bin/sh
redis-cli
keys *
```

Shut down Docker

```
cd $MODALITY_ROOT/docker
docker-compose down
```

Prune Docker content

Sometimes you will want a pristine Docker environment. The simplest way to do this is:

```
cd $MODALITY_ROOT/docker
docker-compose down
docker ps -a # Lists all Docker containers
docker rm <container-id> # Remove any docker containers listed
docker images # Lists all Docker images
docker image rm <image-id> # Remove any docker images listed
docker volume ls # Lists all Docker volumes
docker volume rm <volume-id> # Remove all docker volumes listed
docker system prune # Removes build cache, networks and dangling images
rm -rf data # Removes locally stored database tables
```

You can now rebuild the Docker containers:

```
docker-compose build --no-cache  
docker-compose up
```

Deploy Modality to Heroku



Procedures for this coming soon!