FairDb as Components & QA Database

Tutorial

Evgeny Lavrik
31CBM Collaboration Meeting 22.03.2018

Scope of this training

- Will be concentrated on storing QA and Components data

- There was a tutorial by Denis Bertini and Anton Lymanets on core FairDb ParameterSet storage which is more relevant to the FairTasks
  - see https://indico.gsi.de/event/3620/contribution/4

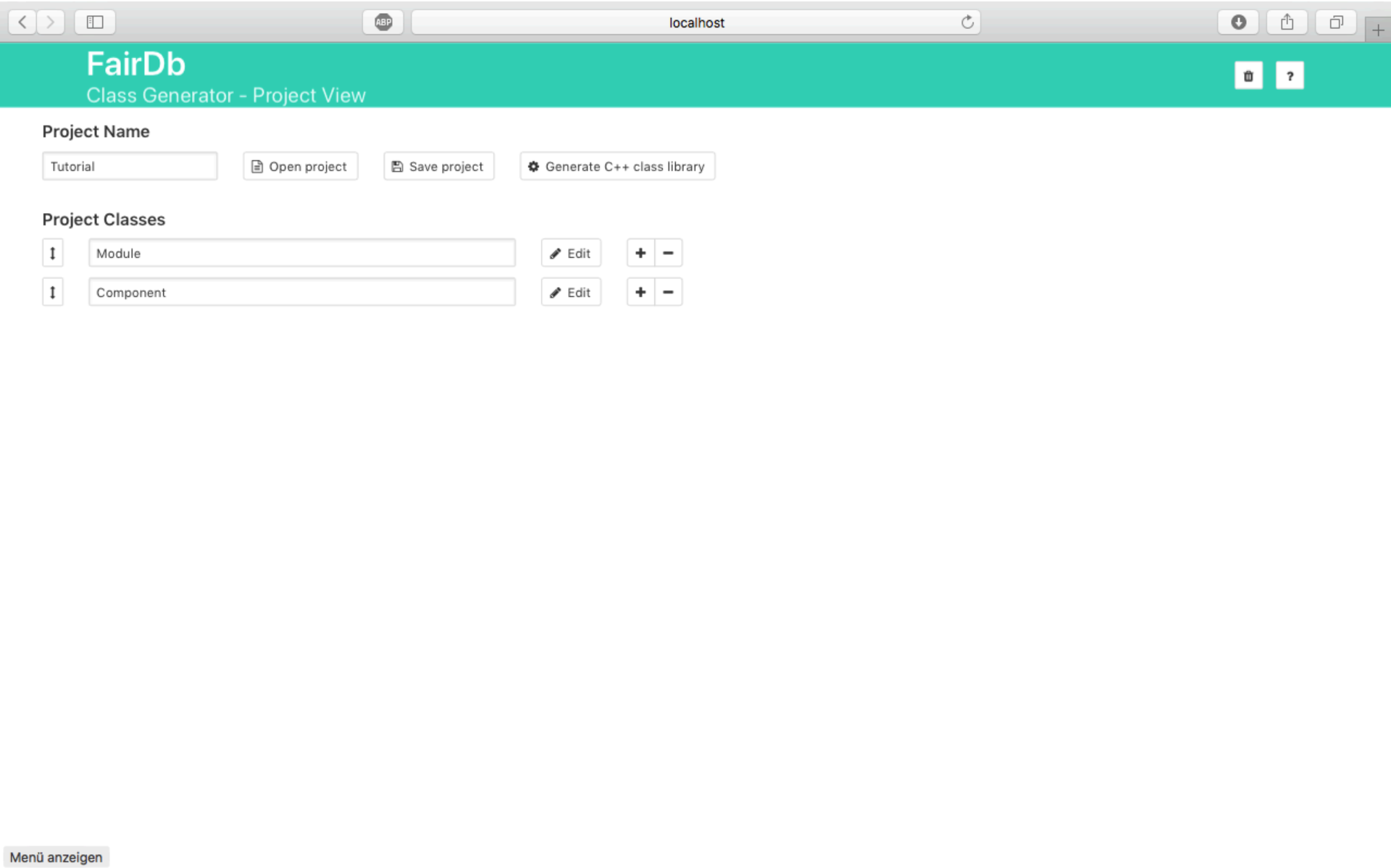
Organizatorial

- You might want to download the slides and follow them locally as well if it is hard to see from afar.

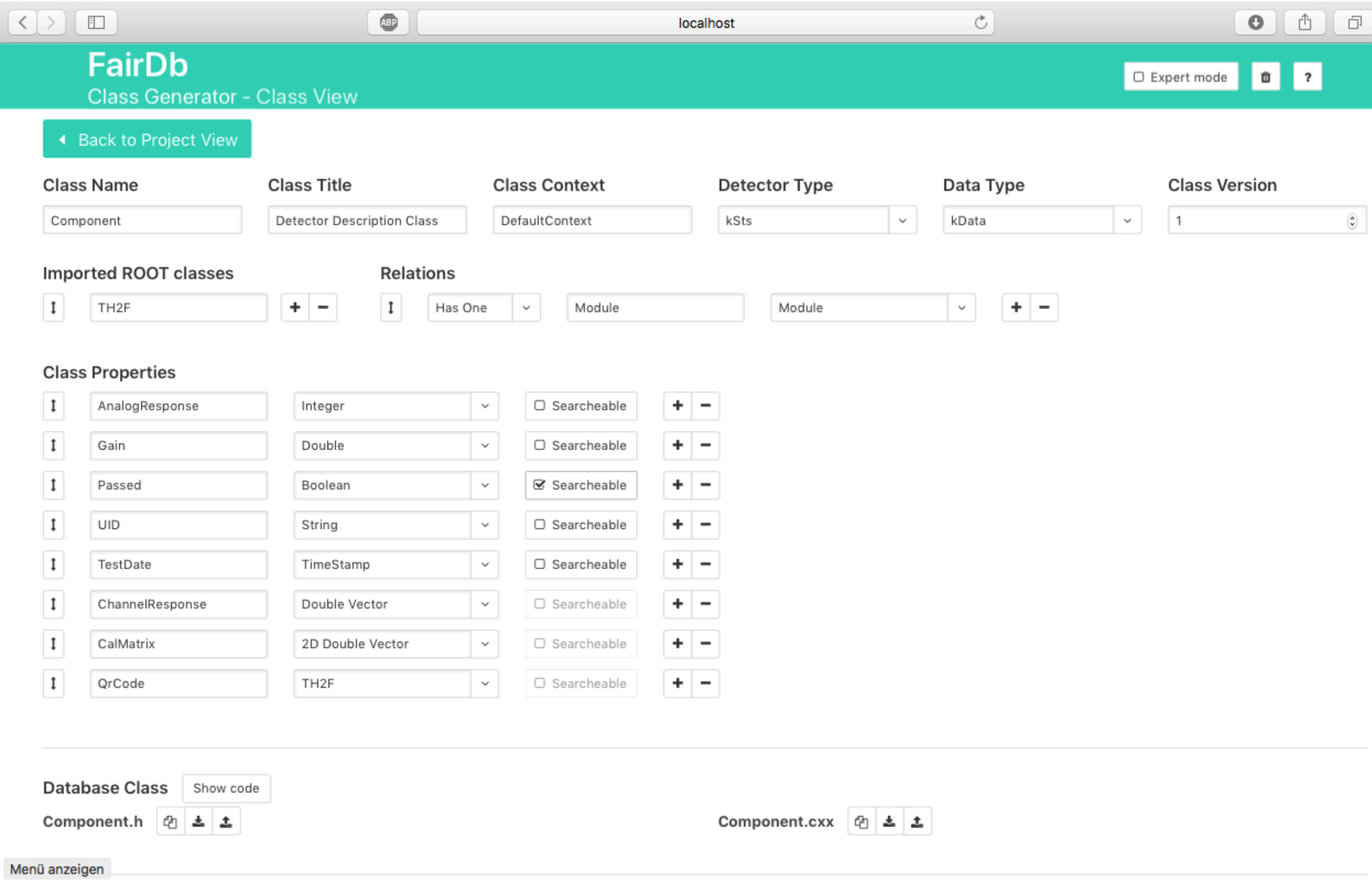
- If you have no gsi linux user account you can use these temporary

Username	Password
dvgast01	Gsi2018!01
dvgast02	Gsi2018!02
dvgast03	Gsi2018!03
dvgast04	Gsi2018!04
dvgast05	Gsi2018!05
dvgast06	Gsi2018!06
dvgast07	Gsi2018!07
dvgast08	Gsi2018!08

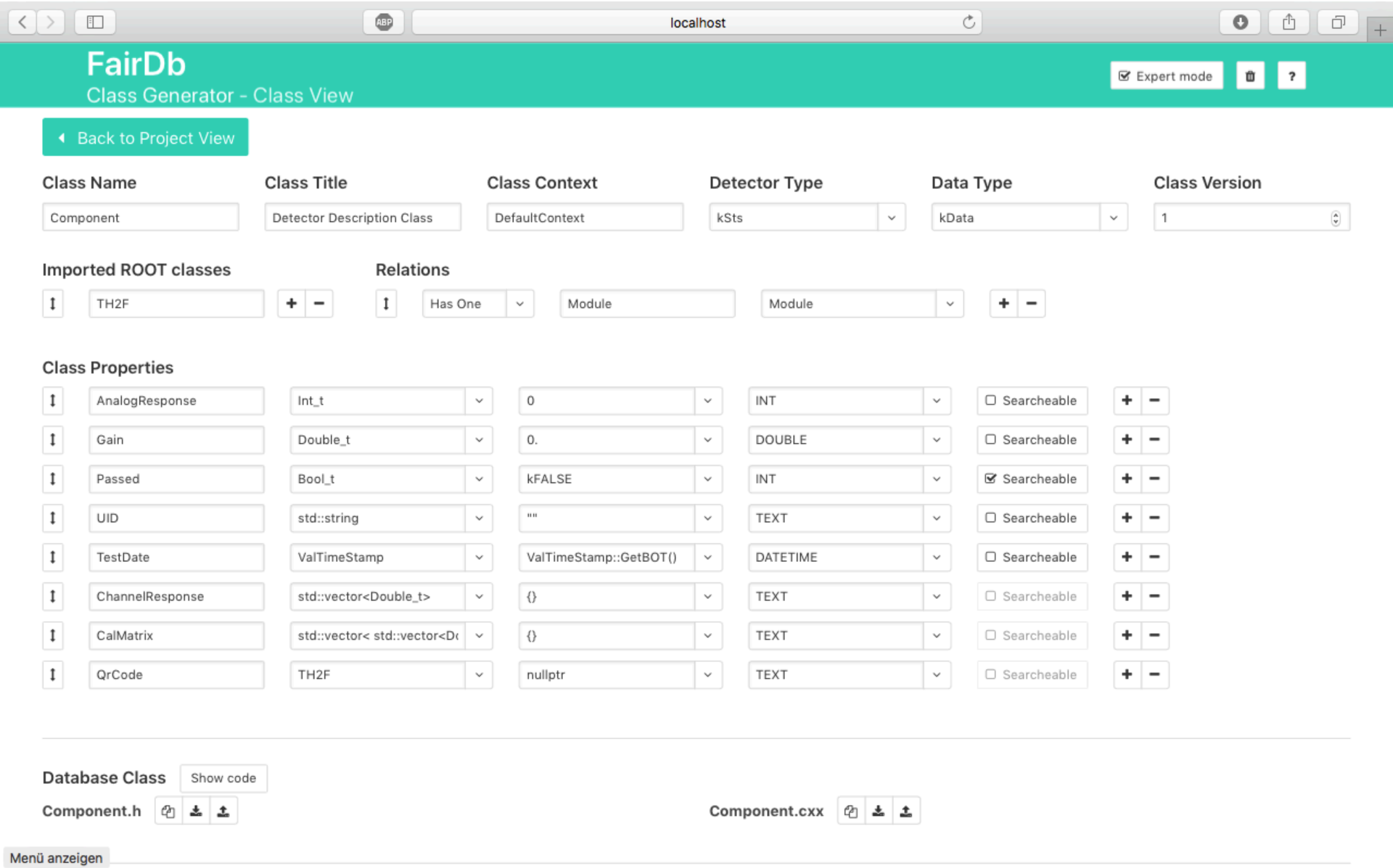
Defining a project



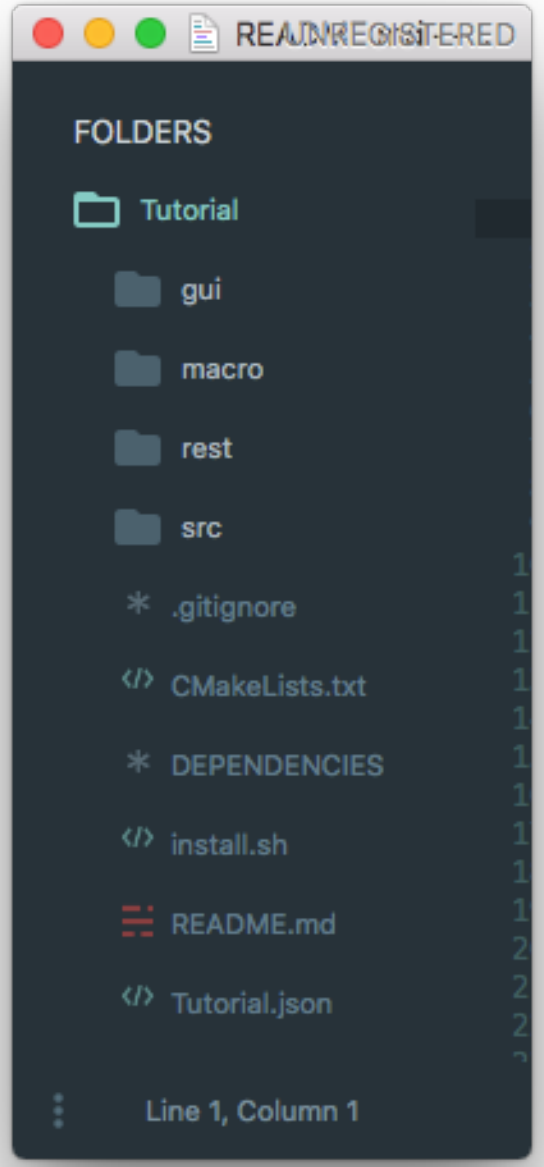
Describing a class



Describing a class, expert mode



Project Structure

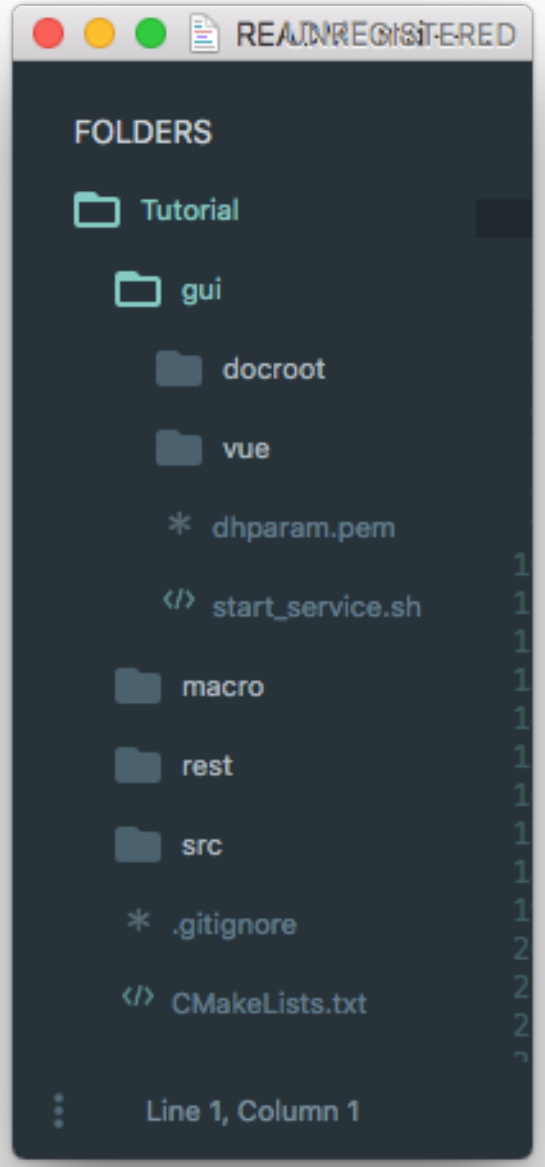


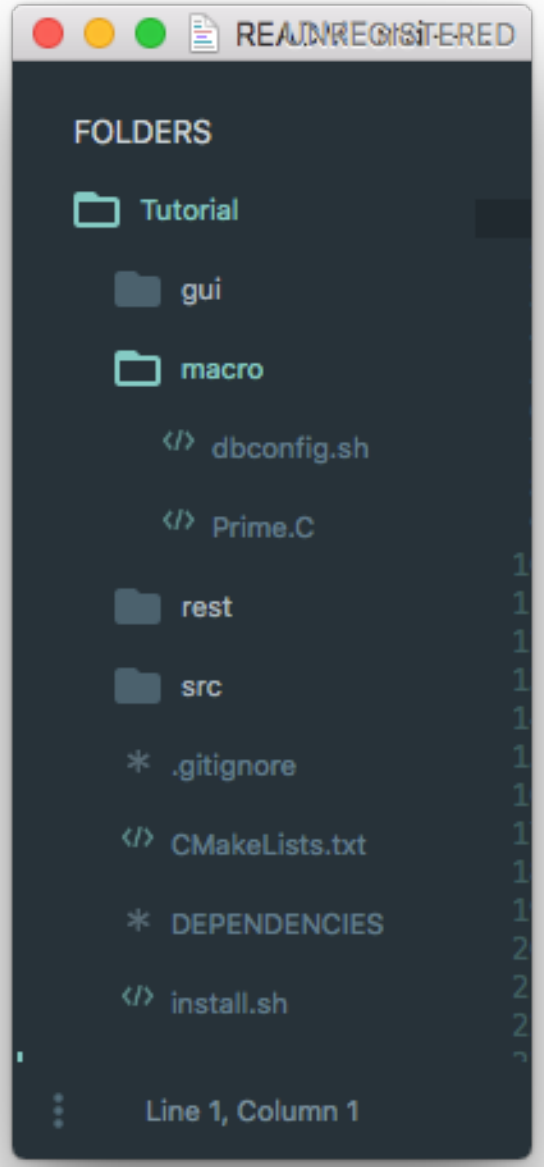
The generated project will contain:

- Short docu (README, DEPENDENCIES)
- src ROOT class sources
- macro DB configuration file, priming macro
- rest sources for RESTful web service
- gui sources for the web-site

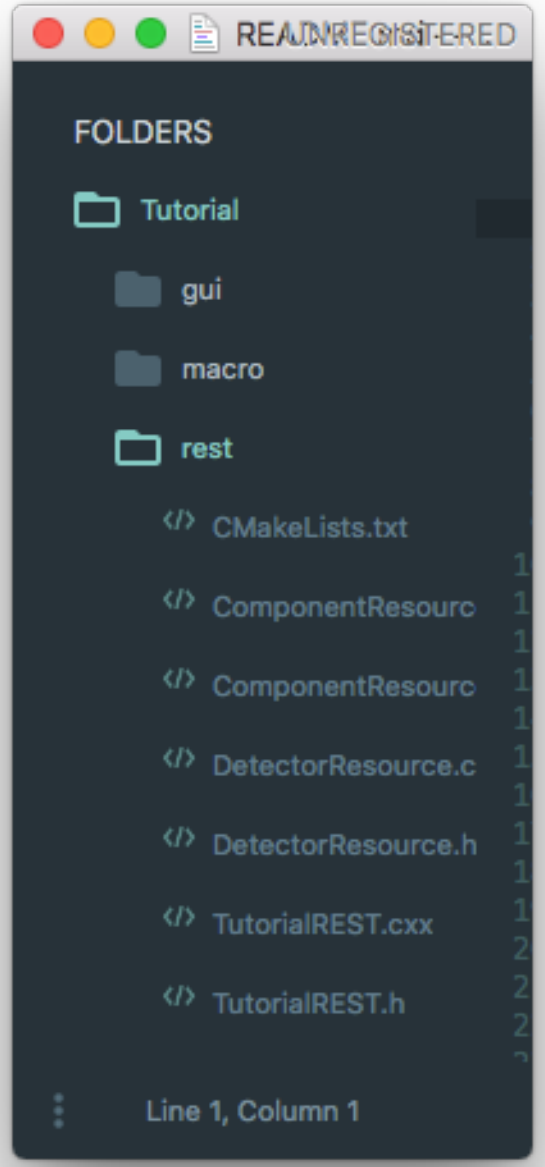
- Install.sh auto installation and deployment script

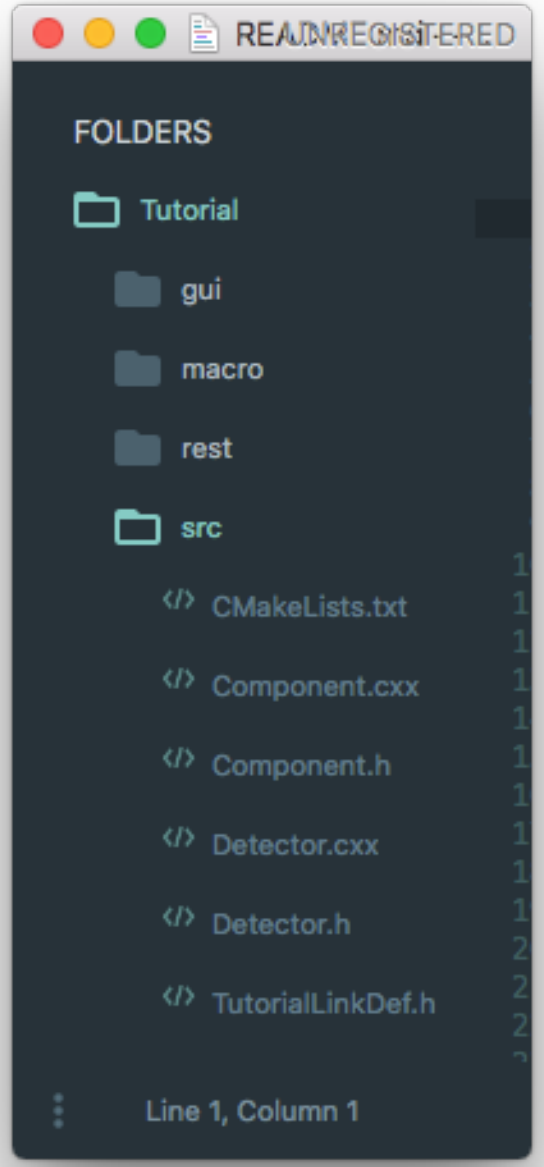
Project Structure, cont.





Project Structure, cont





Project installation

- Local (be sure to install dependencies)

. install.sh

- GSI Linux cluster

. /cvmfs/fairroot.gsi.de/fairroot/v-17.10b_fairsoft-oct17p1/bin/FairRootConfig.sh
export FAIRROOTPATH=/cvmfs/fairroot.gsi.de/fairroot/v-17.10b_fairsoft-oct17p1/
. install.sh

- For this tutorial (from internal GSI network)

. /d/cbm05/dbsoft/FairDb/bin/FairDbConfig.sh
. install.sh

Understanding dbconfig.sh

Makes FairDb know to which database it should talk, by which means, what is the authorization

For local applications, just call it by:
. dbconfig.sh local_sqlite

This will read from or write to the file test.sqlite in the directory where you have invoked ROOT from

```
# SQLite
export FAIRDB_TSQL_URL="sqlite://test.sqlite"
export FAIRDB_TSQL_USER="test"
export FAIRDB_TSQL_PSWD="test"
```

Importing data

- After compilation:

Open file macro/Prime.C and define your data (uncomment the lines in the vector initialization in all classes)

root -l macro/Prime.C

Check out the **Prime_FairDbUser()** function, there you can set your user credentials

```
std::vector<Component_t> components = {
// define your data here
// {-1, 0, 0, 0., kFALSE, "", ValTimeStamp::GetBOT(), {}, {}, nullptr },
};
```

Data query interfaces

For every class there are static methods available:

vector<**Component**> **Component**::GetAll(UInt_t rid = ValTimeStamp())

unique_ptr<**Component**> **Component** ::GetById(Int_t id, UInt_t rid = ValTimeStamp())

vector<**Component**> **Component**::GetByIds(vector<Int_t> ids, UInt_t rid = ValTimeStamp())

vector<**Component**> **Component**::GetBy(function<bool(const **Component**&)> condition, UInt_t rid = ValTimeStamp())

Instance method: vector<Component> GetAllVersions()
Returns you previous versions of that class with a certain Id

Try it in the console:
root [0] **Component**::GetAll**()**
(std::vector<Component>) { @0x7fc1e168b1b0 }

root [1] **Component**::GetById**(**0**)**
(std::unique_ptr<Component>) @0x7fc1e2b01bb0

Data query interfaces, cont

For the classes generated you will receive the methods:
To look up relations:

- **One to one**

root [0] auto component = **Component**::GetById(0)
(std::unique_ptr<Component, default_delete<Component> > &) @0x10fc2c360

root [1] component->GetModule**()**
(std::unique_ptr<Module>) @0x7fcfed26c270

- **One to many**

root [0] auto module = **Module**::GetById(0)
(std::unique_ptr<Module, default_delete<Module> > &) @0x10e455360
root [1] module->GetComponents**()**
(std::vector<Component>) { @0x7ff4db2f9e50 }

To look up “searcheable” properties:
root [0] **Component**::GetByPassed**(**true**)**
(std::vector<Component>) {}

Data storage interfaces

For every class there are storage methods available:

Static method:
void StoreArray(vector<**Component**>, UInt_t rid = 0, string logMessage = "")

Instance method:
void store(UInt_t rid = 0)

Try it in console:

root [0] **Component** c
root [1] c.store**()**
root [2] c.GetId**()**
(int) 1

Notice that new record gets new Id

Rule is: if you try to store a class with Id == -1, the next sequential Id will be allocated
If you save with a certain Id, the record will be created with that particular Id. This might **overwrite** your data.

Data access interfaces, versioning

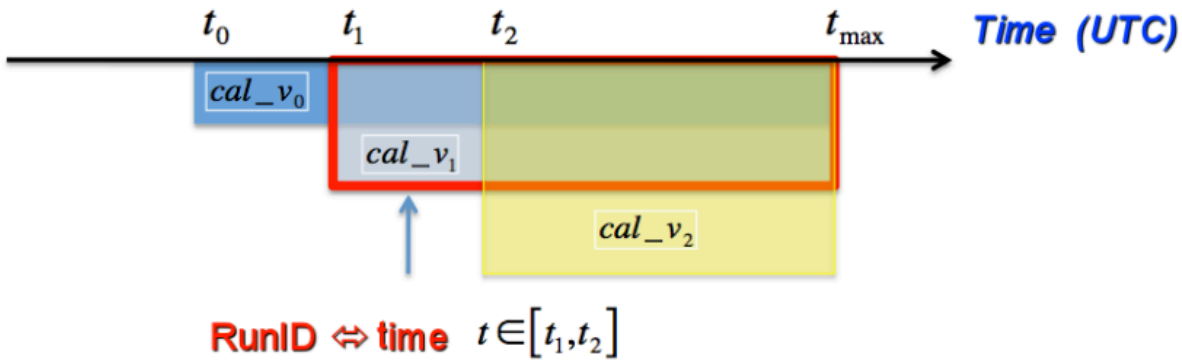
Every class can be stored with a runId, which is a UNIX timestamp

This concept leads to versioning, where you can query your historical data.

Rule is, when storing, the default timestamp is 0 which is 1.1.1970
When reading, it is always current timestamp, meaning you will get the latest version

root [0] **Component**::GetAll**()**
(std::vector<Component>) { @0x7fc053ecc420, @0x7fc053ecc5f8 }

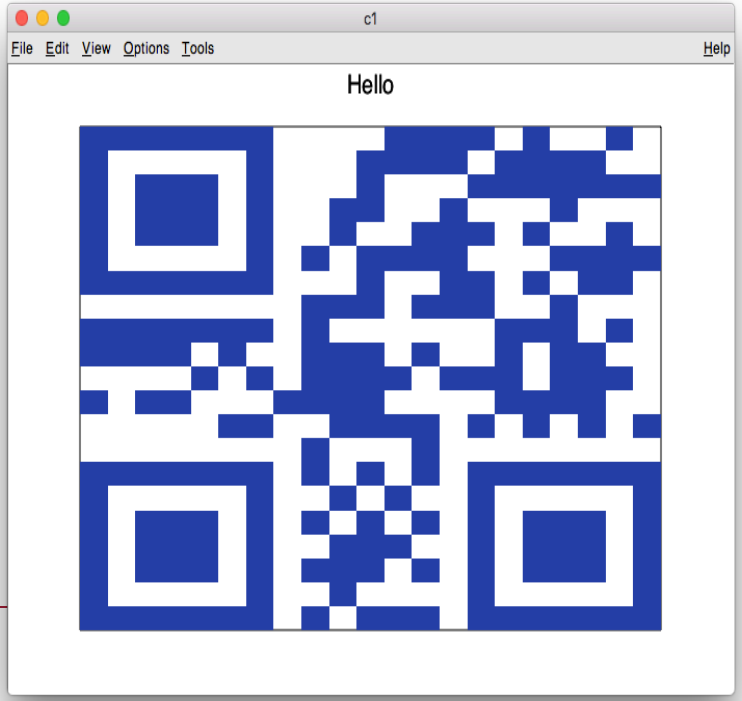
- root [1] **Component**::GetAll**(**0**)**
- (std::vector<Component>) { @0x7fc0553dd980 }



Working with objects

```
root [0] auto c = Component::GetById(0)
root [1] TH2F *qr = FairDbQr::EncodeString("Hello")
root [2] c->SetQrCode(*qr)
root [3] c->SetChannelResponse({1.1, 1.2, 1.3, 1.4, 1.5})
root [4] c->SetCalMatrix({{1.1, 1.2, 1.3}, {2.1, 2.2, 2.3}, {3.1 , 3.2, 3.3}})
root [5] c->store(ValTimeStamp())

root [6] qr->Draw("cola")
```



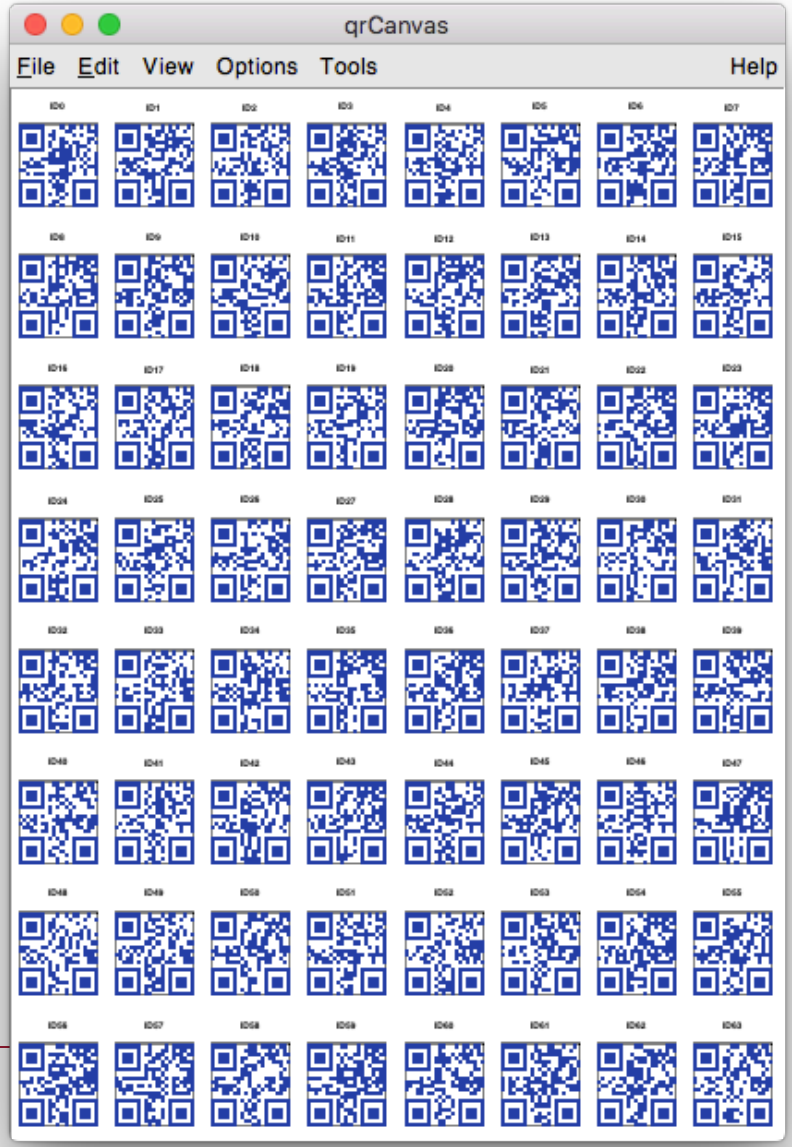
Generating QR codes en-masse

Check out a macro at
$FAIRDBPATH/share/FairDb/macro/Qr/QrPage.C

Adapted to print the codes on a A4 paper format
Can be used to print out stickers

Reads input from file

pushd
$FAIRDBPATH/share/FairDb/macro/Qr/QrPage.C
root –l QrPage.C



Visualizing data

For the data described in the generator, the web-service and a graphical web interface is created as well

It resides in directory *gui* and configured in *start_service.sh*
There you can configre http address and port, where the data will be hosted

There will be SSL certificate generated to enable the secure data transfer and authorization

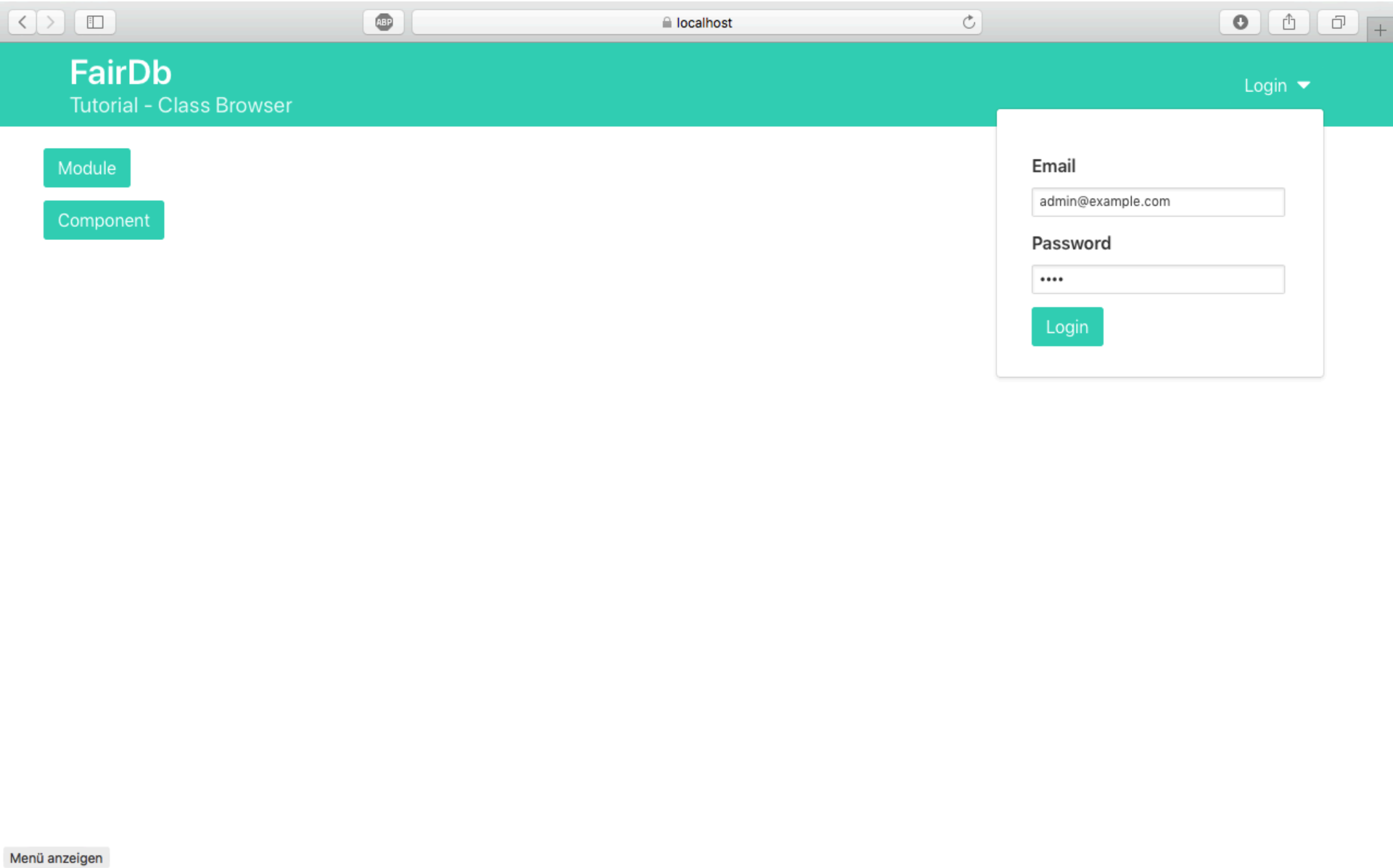
Since we are working with sqlite, be sure to:
mv test.sqlite gui/

Start service by:
. start_service.sh

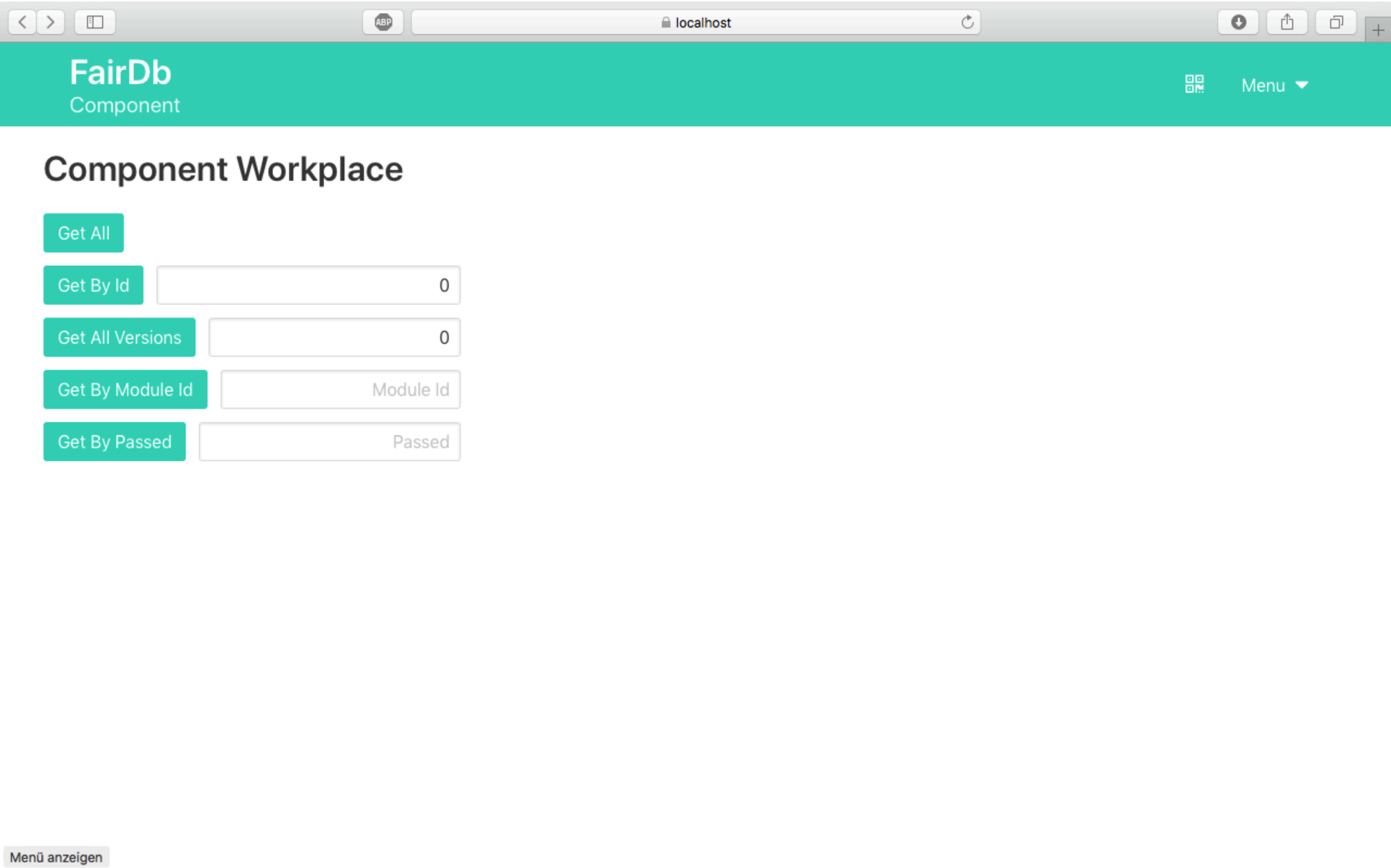
Navigate to https://localhost:8008

```
$REST \
    --docroot ./docroot/ \
    --https-address=0.0.0.0 \
    --https-port=8008 \
    --ssl-certificate=server.pem \
    --ssl-private-key=server.key \
    --ssl-tmp-dh=dhparam.pem \
    --ssl-cipherlist='ECDHE-RSA-AES128
```

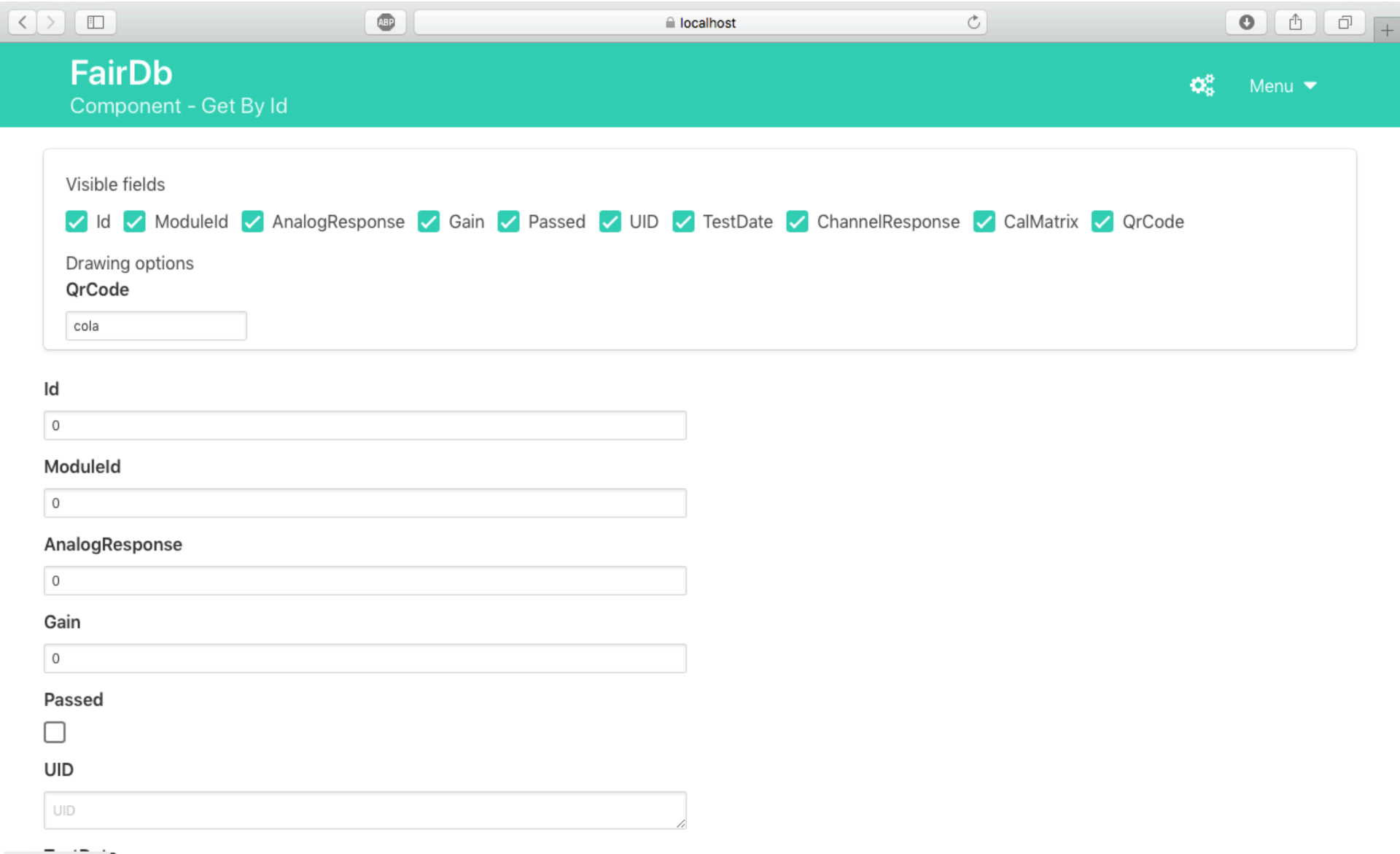
Visualizing data, main view



Visualizing data, class view



Visualizing data, instance view



Visualizing data, instance view

localhost

FairDb

Component - Get By Id

Menu

Id

0

ModuleId

0

AnalogResponse

0

Gain

0

Passed

UID

UID

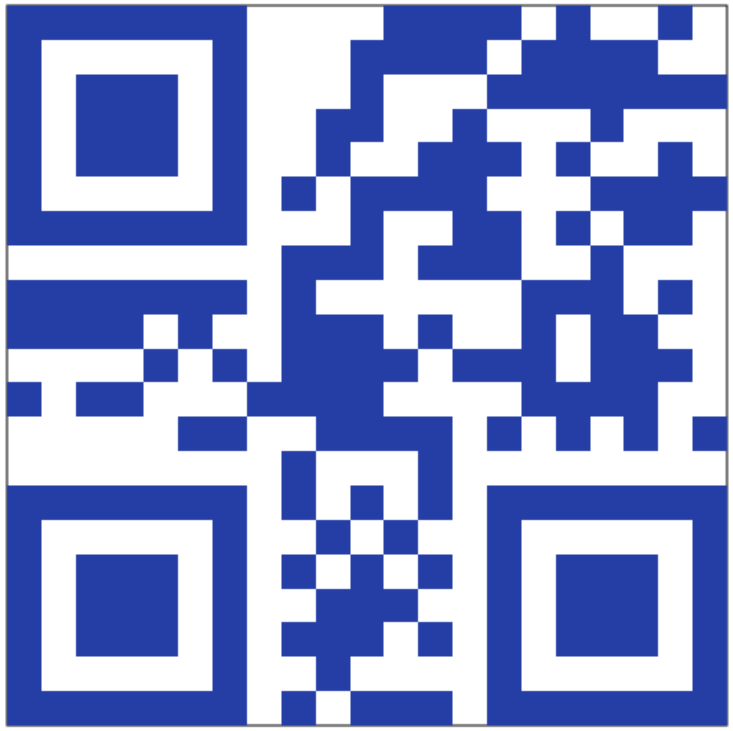
TestDate

1.1.1970

ChannelResponse

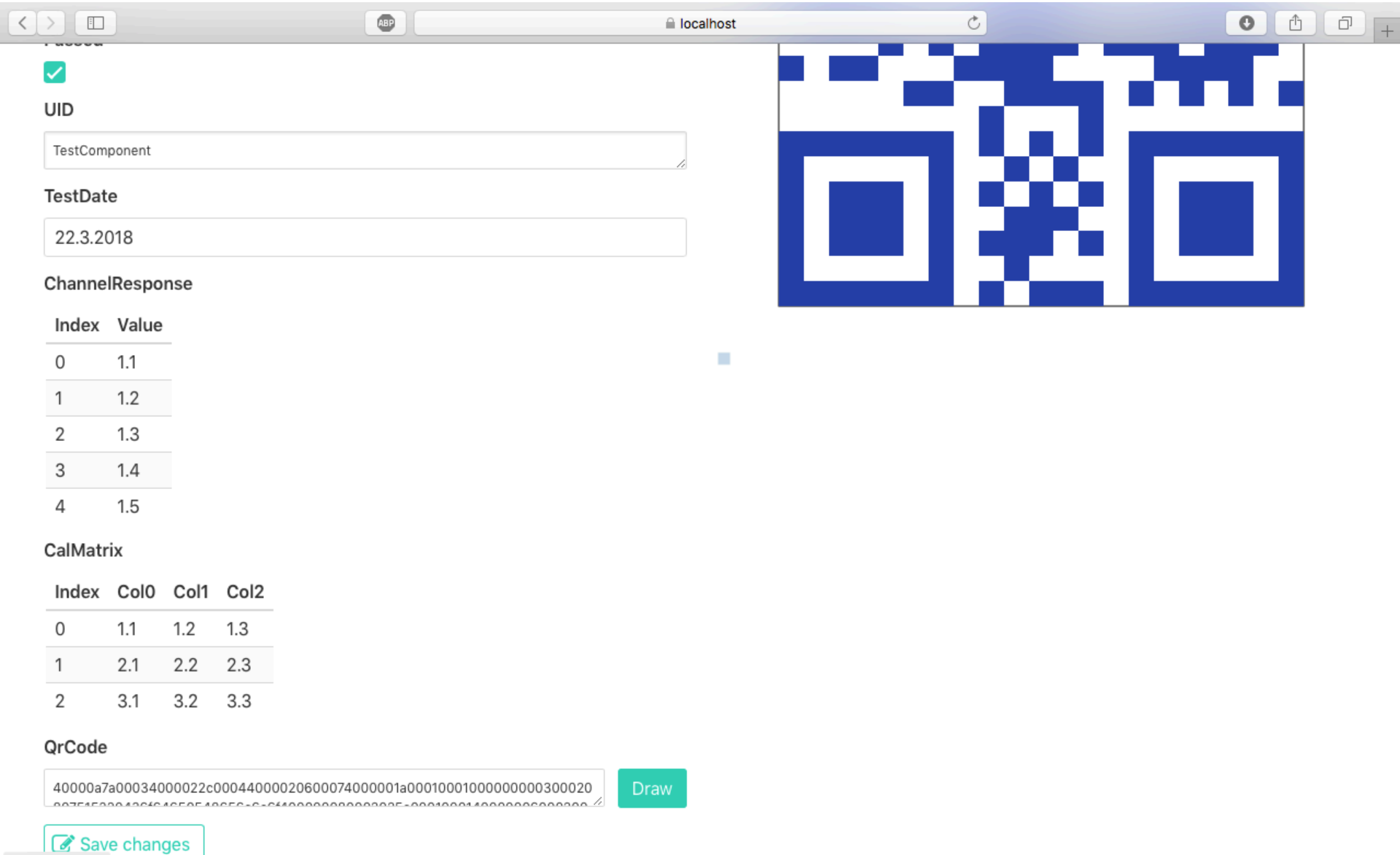
Index	Value
0	1.1
1	1.2
2	1.3

Hello

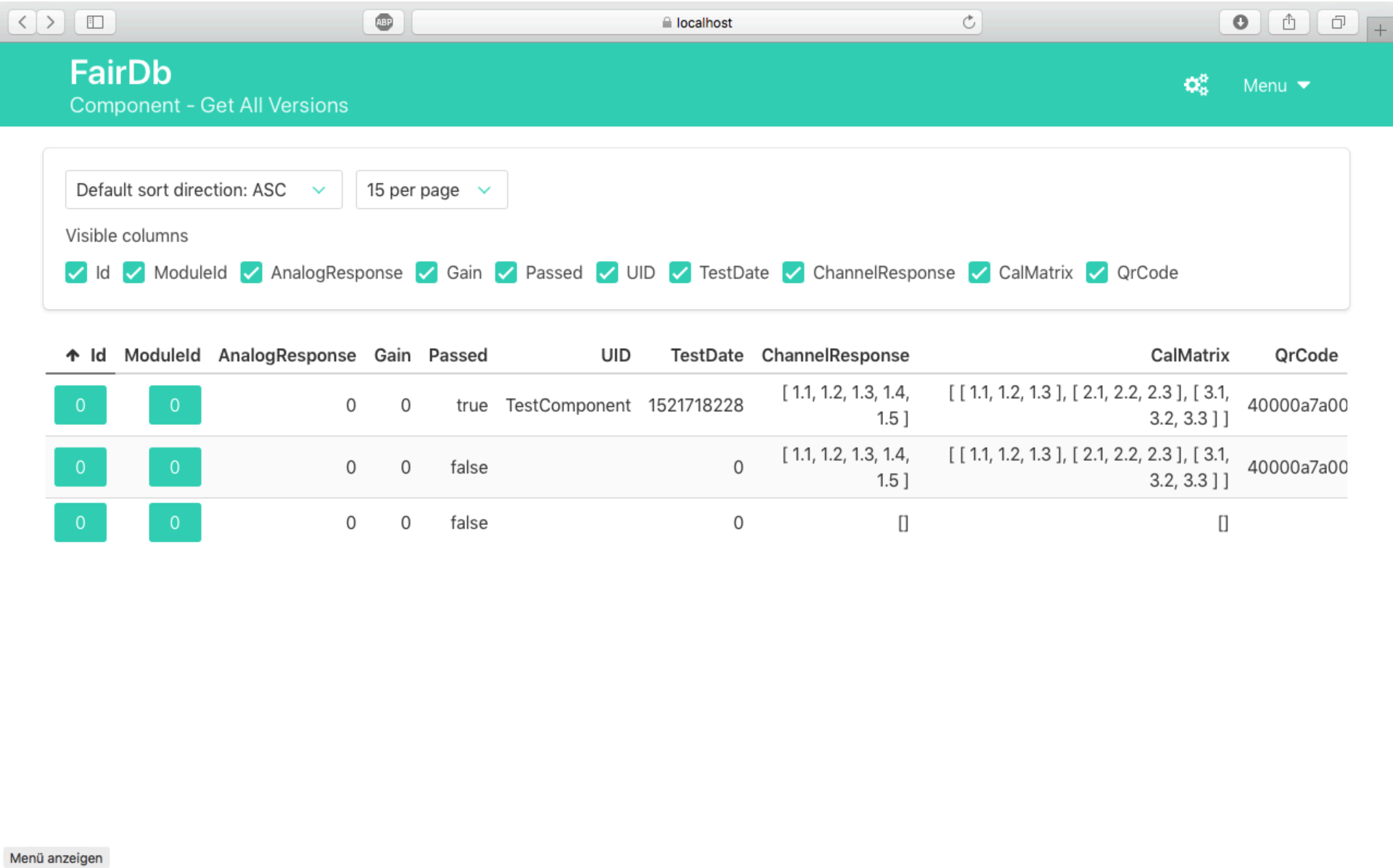


Menü anzeigen

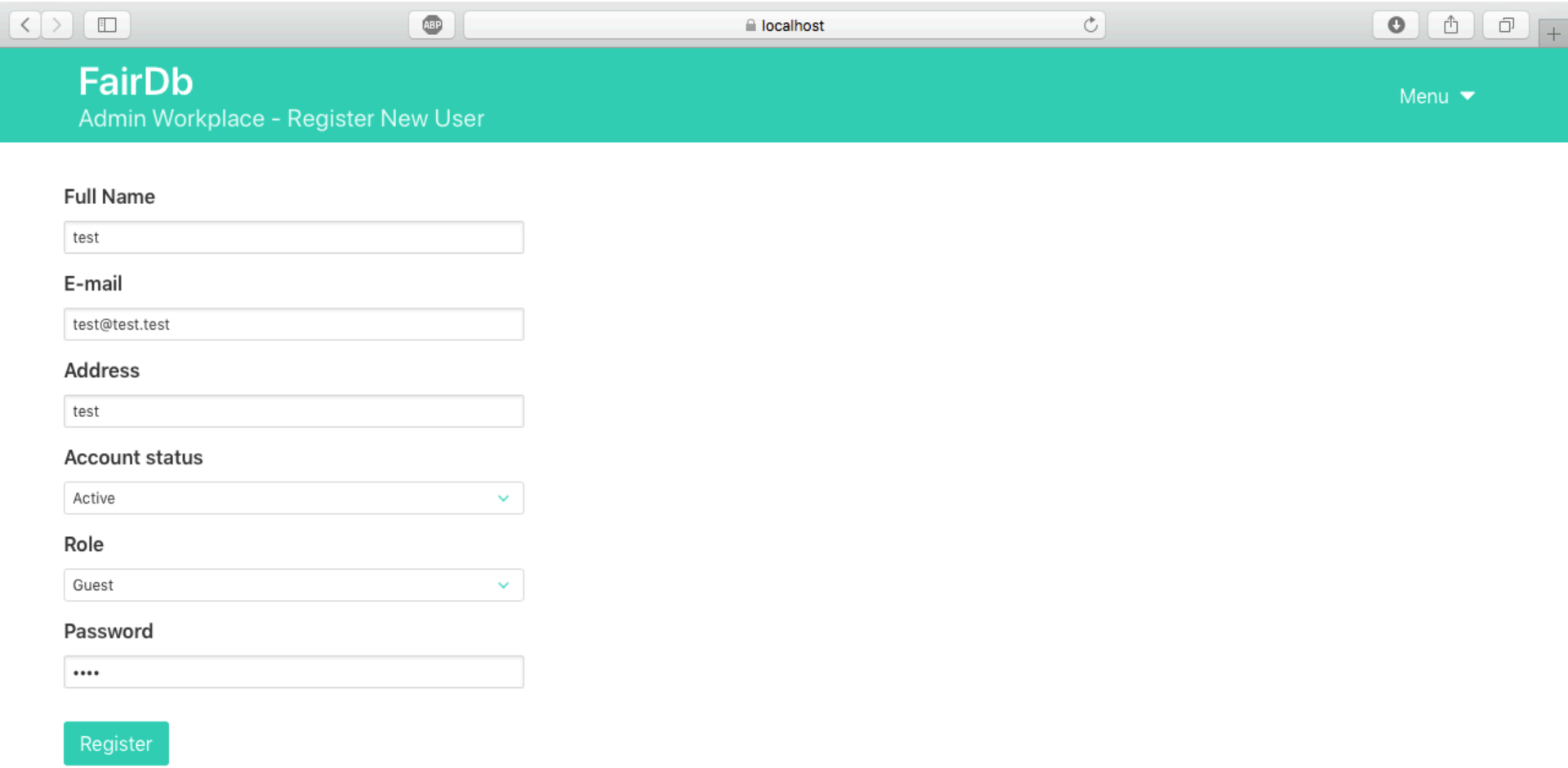
Visualizing data, editing



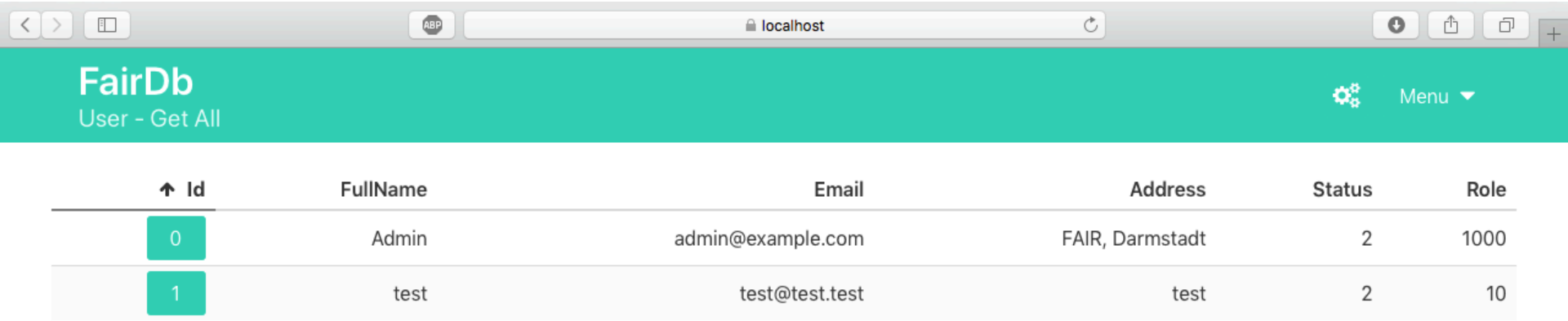
Visualizing data, listing all versions



Registering users



Registering users



Access control

If you need to redefine the default permissions to the functions you should:

For web service:

visit the generated files in **rest** directory with *Resource.cxx and edit AddEndpoint(…) or override the permissions with SetEndpointRequiredRole(…)

For web graphical interface

visit the gui/vue/src/router/*Routes.js and edit the roles

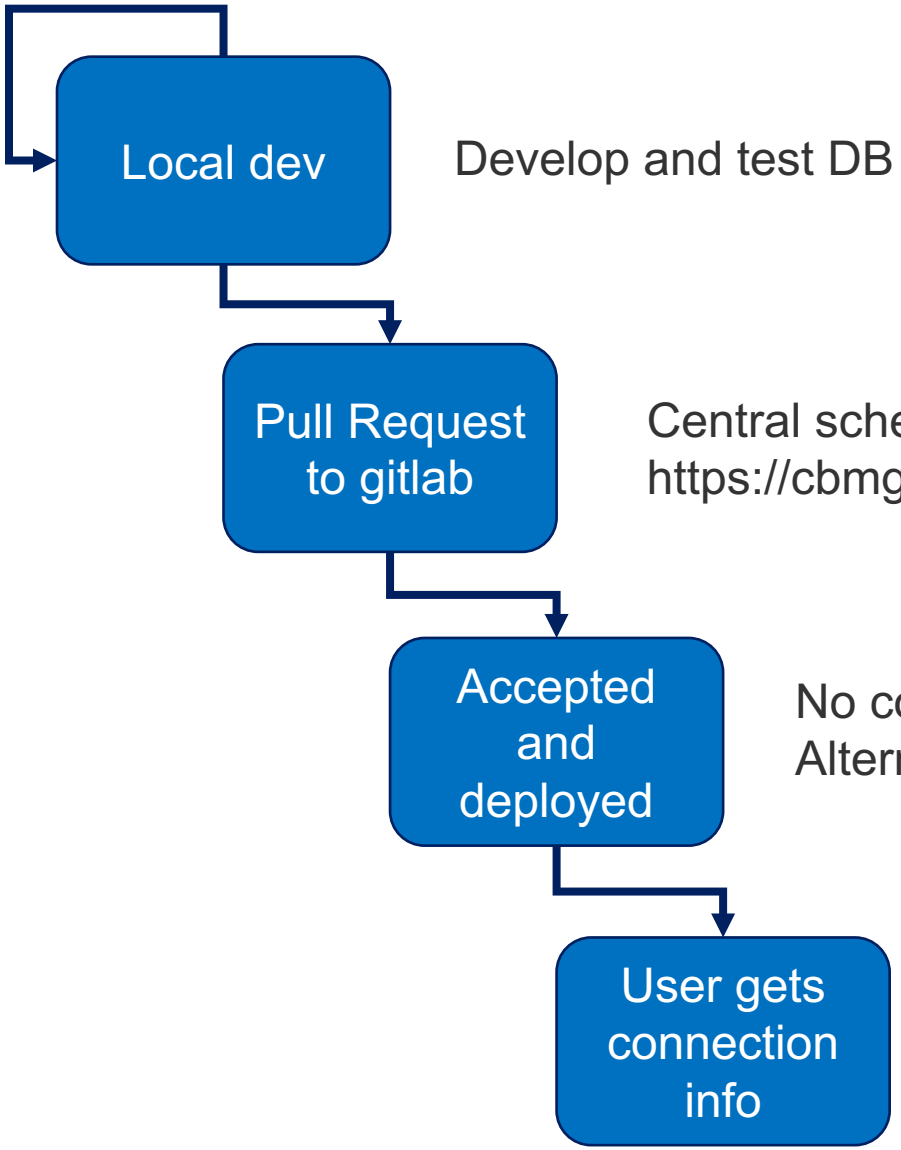
```
ComponentResource::ComponentResource(Wt::WServer& server, Wt::WObject *parent)
  : FairDbWtGenericResource<Component>(server, parent)
{
  AddEndpoint("/Component/GetByModuleId", FairDbUserRole::kGuest, boost::bind(&Co
  AddEndpoint("/Component/GetByPassed", FairDbUserRole::kGuest, boost::bind(&Comp

  AddEndpoint("/Component/GetModule", FairDbUserRole::kGuest, boost::bind(&Compor
}
```

Then you need to recompile the project by typing
. install.sh once again

```
{
  path: '/Component/GetByModuleId/:ModuleId',
  name: 'component-getbymoduleid',
  component: ComponentGetByModuleId,
  meta: {
    access: AccessLevel.Guest,
  },
  props: (route) => ({ Rid: Number(route.query.Ri
},
```

Database deployment workflow



Develop and test DB schema, data export scripts, etc.

Central schema store. Currently at https://cbmgsi.githost.io/evgeny.lavrik/dbSchemaStore

No connection from outside to DB server atm
Alternative: DB server in Uni Tübingen

e.g. WebInterface URI,
Postgresql endpoint URI and creds.