TEAM TEAM DEVELOPER'S MANUAL

DEPLOYMENT OF THE FRONTEND

- Deployment client application:
  - Go to **/src/services/backend-service.js**, at the top of the file.
    Change the baseURL property to the location of the server application.
    (Currently using a custom domain getnova.app)
  - On command line, run **npm install**, **npm run build** to build the client application into static file in /dist dictionary
  - Statically host the contents of the /dist dictionary. (Currently using **Netlify**)
  - Ensure that the hosting provider is set up with a catch-all fallback route that redirects to index.html to allow Vue.js history mode routing. This is achieved in our current hosting provider (Netlify) by placing a _redirects file in the root with the content "**/* /index.html 200**" (included in the project source by default under /public).

DEPLOYMENT OF THE BACKEND

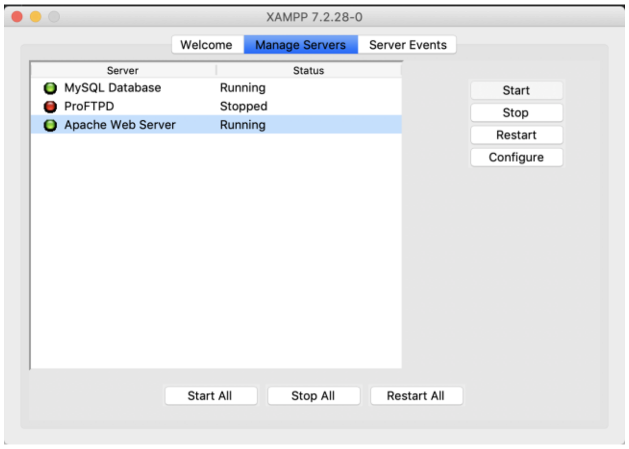
- Create the tar file by running the Gradle command:
  - **gradle distTar**
- Transfer the created tar file to the VPS e.g. through FileZilla. The server needs an SSL certificate – this can be obtained from Certbot.
- Export the tar file into a desired directory
- Transfer the even-kite file from main/resources
- Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable according to the path of the even-kite file
- Extract the tar by running the command:
  - **tar -xf back-end.tar**
- Set up the hibernate configs for database: this includes the following files:
  - backend/src/main/resources/application.yml
  - backend/src/main/resources/hibernate.cfg.xml
- Run ./back-end

Relevant tables for the database will be created as they're invoked by the Controller along with a user account.

SETTING UP LOCALLY

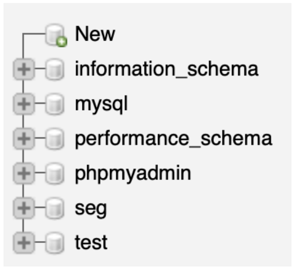
- Download necessary components
  - Gradle (**https://gradle.org/install/**)
  - Npm & Node (**https://www.npmjs.com/get-npm**)
  - Vue.js (**https://vuejs.org/v2/guide/installation.html**)
  - Xampp [For localhost use] (**https://www.apachefriends.org/index.html**)

1. Open Xampp
2. Start Mysql and Apache

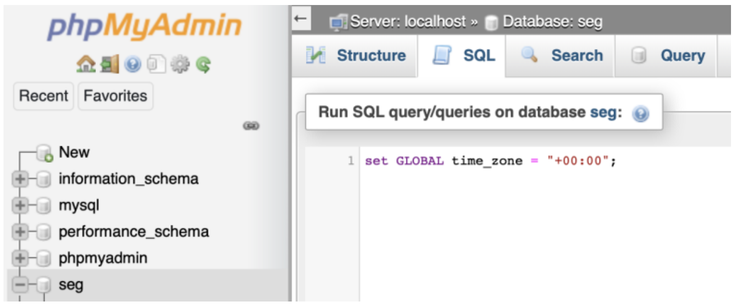


Server	Status
MySQL Database	Running
ProFTPD	Stopped
Apache Web Server	Running

3. Open Browser and go to (**http://localhost/phpmyadmin/**)
4. Create a database which can be connected to; e.g. testunittest with no password. Relevant tables will be automatically created with constraints enforced once ‘gradle test’ or the relevant controllers/managers are invoked (through POSTMAN from ‘gradle run’)



5. Click the created table and click SQL button on the top nav bar and run command:
   set GLOBAL time_zone = "+00:00";



```
set GLOBAL time_zone = "+00:00";
```

6. Do the same with the created test table

7. Go to **SEGMajorProject/back-end/src/main/resources**

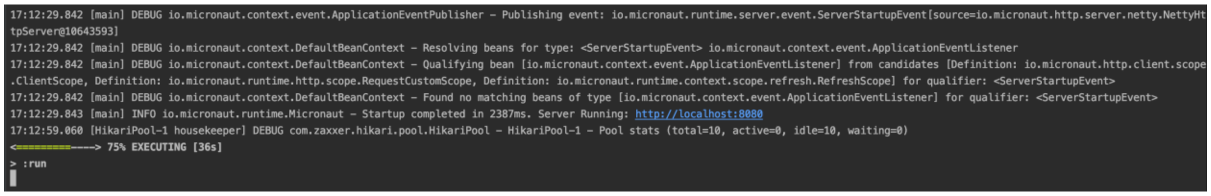
   Add correct connection string and credentials in **application.yml**, **hibernate.cfg.xml** and **hibernate.properties** according to the database config. This will be the database for the main running.

   (Our setting is **'127.0.0.1:3306/seg'**, the format may change according to your own db settings.)

8. Go to **SEGMajorProject/back-end/src/test/resources**
   Adjust application.yml, testhibernate.cfg.xml and hibernate.properties to match a database for testing. This database will automatically have all its tables created and wiped (or just wiped) once tests are complete via 'gradle test'.

9. Go to **SEGMajorProject/back-end/** in command line
   Run command:
   - **gradle test** (to see the test results)

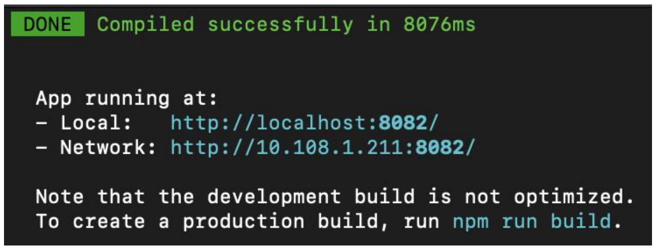
10. Go to **SEGMajorProject/back-end/** in command line
    Run command:
    - **gradle build**
    - **gradle run** (to run the backend server)



```
17:12:29.842 [main] DEBUG io.micronaut.context.event.ApplicationEventPublisher - Publishing event: io.micronaut.runtime.server.event.ServerStartupEvent[source=io.micronaut.http.server.netty.NettyHt
tpServer@10643593]
17:12:29.842 [main] DEBUG io.micronaut.context.DefaultBeanContext - Resolving beans for type: <ServerStartupEvent> io.micronaut.context.event.ApplicationEventListener
17:12:29.842 [main] DEBUG io.micronaut.context.DefaultBeanContext - Qualifying bean [io.micronaut.context.event.ApplicationEventListener] from candidates [Definition: io.micronaut.http.client.scope
.ClientScope, Definition: io.micronaut.runtime.http.scope.RequestCustomScope, Definition: io.micronaut.runtime.context.scope.refresh.RefreshScope] for qualifier: <ServerStartupEvent>
17:12:29.842 [main] DEBUG io.micronaut.context.DefaultBeanContext - Found no matching beans of type [io.micronaut.context.event.ApplicationEventListener] for qualifier: <ServerStartupEvent>
17:12:29.843 [main] INFO io.micronaut.runtime.Micronaut - Startup completed in 2387ms. Server Running: http://localhost:8080
17:12:59.060 [HikariPool-1 housekeeper] DEBUG com.zaxxer.hikari.pool.HikariPool - HikariPool-1 - Pool stats (total=10, active=0, idle=10, waiting=0)
<=========----> 75% EXECUTING [36s]
> :run
```

11. Go to **SEGMajorProject/front-end/seg-major-frontend/** in command line
    You may have to change BaseURL in **/src/services/backend-service.js** to the host address you received in the last step.
    Run command **npm install** first, then run:

- **npm run serve** (to start the frontend)



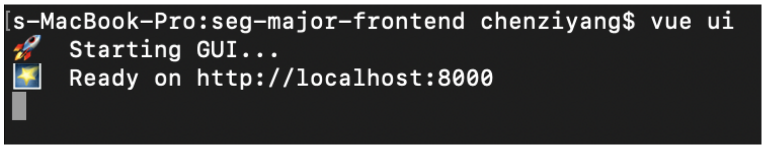
```
 DONE  Compiled successfully in 8076ms


  App running at:
  - Local:   http://localhost:8082/
  - Network: http://10.108.1.211:8082/

  Note that the development build is not optimized.
  To create a production build, run npm run build.
```

Or

- **vue ui** (to start the vue.js user interface)



```
[s-MacBook-Pro:seg-major-frontend chenziyang$ vue ui
🚀  Starting GUI...
🌠  Ready on http://localhost:8000
```

Then start the project from the user interface e.g. heading to the npm host site on a web browser.

ACCESSING LIVE TESTING DATABASE

1. On the terminal swap to branch ‘test_db_setup’ via command ‘git checkout test_db_setup’.

2. Ensure your **GOOGLE_APPLICATION_CREDENTIALS** environment variable still points to the even-kite JSON file path.

YOUR FIRST ADMIN ACCOUNT

- Once running, if your user table is empty, a default account is created: if your database doesn’t contain any tables, running gradle test or gradle run will create relevant tables and add your first user account:
  - Email: admin@admin.com
  - Password: admin
  - Name: Administrator

- Post requests to [host]/user/login/ (e.g. getnova.app/ or localhost:8080/) can take a similarly made JSON body consisting of: in order to obtain a valid X-API Key.

```
{
        "name": "Administrator",
        "password": "admin",
        "email": "admin@admin.com"
}
```