01 - Installation

Edit New page

Jeff Caldwell edited this page 3 hours ago · 1 revision

> Important
>
> These installation instructions cover installing and running a **development version** of the ISS Analysis Tool. This software is a demonstration and should not yet be considered "production-ready" and further development is required to make it suitable to use in a production environment. For further development, please contact the members of Team Noname

> Important
>
> You must have Docker and Docker Compose installed on the host computer for everything to work correctly. Both can be obtained by installing Docker Desktop. If you don't want to install Docker Desktop, follow the installation instructions for installing the Docker Engine and Docker Compose separately.

Installing the ISS Analysis Tool

First, clone or download the project repository. Navigate to a suitable directory on the host machine and run one of the following commands:

```
# Clone via http
git clone https://github.com/4306-team-noname/barrios

# or clone via ssh
git clone git@github.com:4306-team-noname/barrios.git
```

Once the repository has been cloned or downloaded and extracted onto the host machine, navigate to the `barrios` directory to install the project's dependencies and initialize the tool's seed data.

```
cd <installation_dir>/barrios
```

> Note
>
> The top-level `barrios/` project directory contains three subdirectories. Only one of these directories is

+ Add a custom sidebar

Clone this wiki locally

https://github.com/4306-te

> necessary to run the project — the `server/` directory. The `archive/` directory is included in the repository as evidence of the team's exploratory work, and the `assets/` directory is included to serve images and diagrams to the project wiki. You can safely investigate or delete either directory without affecting the application or its ability to run. **All other files in the top-level `barrios/` directory are necessary**.

```
barrios/
├── archive/  # Exploratory code and prototypes
├── assets/   # Documentation images and diagrams
└── server/   # Real project code (do not delete)
```

Create a virtual environment and install dependencies

Run these three commands from the top-level `barrios/` directory:

```
# create a virtual environment
python -m venv .venv

# Activate the virtual environment
source .venv/Scripts/activate #  On Windows
source .venv/bin/activate #  On Mac/Linux

# Install project dependencies
pip install -r requirements.txt
```

Install and initialize the development database

Edit postgres user and password in `docker-compose` file

Open `barrios/docker-compose.yml` in a text editor and edit line `9` to add your preferred postgres admin password. You can also leave it as it is if you're only evaluating a development version of this app.

```
environment:
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD:-changem
```

The postgres password will be referred to as `<POSTGRES_PASSWORD>` for the remainder of this guide.

While still in the `barrios/` directory, run the following to create a containerized Postgres database:

```
docker-compose up
```

This will pull the necessary docker images and set them up so you have a PostgreSQL server and a PGAdmin server running.

Set up the database through PGAdmin

1. **Connect to the database server with PGAdmin**
    - i. a. In a browser, navigate to `localhost:5050` (or `http://127.0.0.1:5050`, if you're feeling verbose). This will open a setup page for PGAdmin. It will ask you to set a master password. Set it to whatever you want.
    - ii. In `Quick Links` on the front page, click `Add New Server`.
    - iii. On the tab labeled `General`, set the `Name` to `postgres`.
    - iv. In the `Connection` tab, set the `Host name/address` to `postgres`, the username to `postgres`, and the password to `<POSTGRES_PASSWORD>`. You can toggle `Save password` if you want.
    - v. Click `Save`
2. **Set up the database user**
    - i. This will add the `postgres` server you loaded from the docker container. In the left sidebar, click `Servers`, then `postgres` to expand the menu.
    - ii. Right-click on `Login/Group Roles`, then choose `Create > Login/Group Role...`
    - iii. In the `General` tab, set the name to `barrios`.

    iv. In the `Definition` tab, set the password to `barrios123`
    v. In the `Privileges` tab, turn on `Can log in?`
    vi. Click `Save`
3. **Add the database**
    i. In the left sidebar, under `postgres`, right-click on `Databases` and select `Create > Database...`
    ii. In the `General` tab, set the `Database` to `barrios`, and the `Owner` to `barrios`
    iii. Click `Save`

Create a superuser and seed the database

1. Navigate to the `barrios/server/barrios` directory. You're going to need to run one script and one Django command. Before running the script, you should make sure that you have the correct permissions to do so. Run this command to change the script's permissions:

```
chmod 775 resetdb.sh
```

2. Open the script in a text editor and change the `ADMIN_EMAIL` and `ADMIN_USERNAME` variables on lines `22` and `23`. If you don't change it, the admin username (for testing purposes only) will be `admin`, and the email will be `testadmin@example.com.

3. Run the script with the command `./resetdb.sh`. It will ask you if you want to proceed. Type `yes`. The script will then attempt to drop all of the data from an existing `barrios` database and reset all the tables to match the application's models. When it's done, it will prompt you for a password. Enter a password.

(OPTIONAL) Seed the database

1. Clone the seed data: A collection of seed data is located in a private repository. If you do not have access to this repository, contact Jeff Caldwell fo access

repository, contact Jeff Caldwell to access.

```
git clone https://github.com/4306-team-noname/se a
```

2. From the `barrios/server/barrios` directory, run the following command to seed the database with the Barrios data:

```
python manage.py init_data
```

It will list all of the files being loaded. If there are any errors, or any files don't match expected data types, it will tell you. If everything went well, it will output:

```
n of n files were saved to the database
successfully.
All data loaded successfully!
🚀 Good luck out there!
```

> **Important**
>
> Files uploaded to or seeded to the IMS Analysis Tool database should conform to specifications defined in the CSV Data Guidelines

Start the server

Now that the setup is all out of the way, it's time to start the server:

```
python manage.py runserver
```

Refer to the User Manual for instructions on running the application.

+ Add a custom footer