# Contributing

Here are some short guidelines to guide you if you want to contribute to the development of the Full Stack FastAPI PostgreSQL project generator itself.

After you clone the project, there are several scripts that can help during development.

\* `./scripts/dev-fsfp.sh`:

Generate a new default project `dev-fsfp`.

Call it from one level above the project directory. So, if the project is at `~/code/full-stack-fastapi-postgresql/`, call it from `~/code/`, like:

```console

$ cd ~/code/

$ bash ./full-stack-fastapi-postgresql/scripts/dev-fsfp.sh

```

It will generate a new project with all the defaults at `~/code/dev-fsfp/`.

You can go to that directory with a full new project, edit files and test things, for example:

```console

$ cd ./dev-fsfp/

$ docker-compose up -d

```

It is outside of the project generator directory to let you add Git to it and compare versions and changes.

\* `./scripts/dev-fsfp-back.sh`:

Move the changes from a project `dev-fsfp` back to the project generator.

You would call it after calling `./scripts/dev-fsfp.sh` and adding some modifications to `dev-fsfp`.

Call it from one level above the project directory. So, if the project is at `~/code/full-stack-fastapi-postgresql/`, call it from `~/code/`, like:

```console

$ cd ~/code/

$ bash ./full-stack-fastapi-postgresql/scripts/dev-fsfp-back.sh

```

That will also contain all the generated files with the generated variables, but it will let you compare the changes in `dev-fsfp` and the source in the project generator with git, and see what to commit.

\* `./scripts/discard-dev-files.sh`:

After using `./scripts/dev-fsfp-back.sh`, there will be a bunch of generated files with the variables for the generated project that you don't want to commit, like `README.md` and `.gitlab-ci.yml`.

To discard all those changes at once, run `discard-dev-files.sh` from the root of the project, e.g.:

```console

$ cd ~/code/full-stack-fastapi-postgresql/

$ bash ./scripts/dev-fsfp-back.sh

```

\* `./scripts/test.sh`:

Run the tests. It creates a project `testing-project` \*inside\* of the project generator and runs its tests.

Call it from the root of the project, e.g.:

```console

$ cd ~/code/full-stack-fastapi-postgresql/

$ bash ./scripts/test.sh

```

\* `./scripts/dev-link.sh`:

Set up a local directory with links to the files for live development with the source files.

This script generates a project `dev-link` \*inside\* the project generator, just to generate the `.env` and `./frontend/.env` files.

Then it removes everything except those 2 files.

Then it creates links for each of the source files, and adds those 2 files back.

The end result is that you can go into the `dev-link` directory and develop locally with it as if it was a generated project, with all the variables set. But all the changes are actually done directly in the source files.

This is probably a lot faster to iterate than using `./scripts/dev-fsfp.sh`. But it's tested only in Linux, it might not work in other systems.