## Table Of Contents

[What should I know before I get started?](#what-should-i-know-before-i-get-started)

\* [Docs](https://github.com/rolling-scopes/rsschool-docs)

[How Can I Contribute?](#how-can-i-contribute)

\* [Local Development](#local-development)

\* [Pull Requests](#pull-requests)

[Styleguides](#styleguides)

\* [Git Commit Messages](#git-commit-messages)

\* [TypeScript Styleguide](#typescript-styleguide)

\* [Specs Styleguide](#specs-styleguide)

## How To Contribute

### Local development

#### Prerequisites

- [Git 2.10+](https://git-scm.com/downloads)

- [NodeJS LTS](https://nodejs.org/en/)

- [Docker](https://docs.docker.com/install/)

- [Docker Compose](https://docs.docker.com/compose/install/)

#### Steps

1. Fork the repository (https://help.github.com/articles/fork-a-repo/)

2. Clone the repository to your local machine (https://help.github.com/articles/cloning-a-repository/)

``` command-line

$ git clone git@github.com:[username]/rsschool-app.git

```

3. Navigate into the directory where you've cloned the source code and install NPM dependencies

``` command-line

$ cd rsschool-app

$ npm install

```

4. If you plan to change the server part, please create a branch for your feature

``` command-line

$ git checkout -b feature-x master

```

5. The application requires a connection to a Postgres database. Here is how to get test database running locally:

Run a Postgres Database locally using Docker & Docker Compose

``` command-line

$ npm run db:up

```

Restore a test database snapshot

``` command-line

$ npm run db:restore

```

If you are done with development, stop the database;

``` command-line

$ npm run db:down

```

6. Run the application in development mode with live reload:

``` command-line

$ npm start

```

7. Do hacking ??????

8. You could specify any environment variable during development using `.env` file. Make a copy of `server/.env.example` and rename it to `server/.env`. We support it via `dotenv` package. More information about usage here: https://github.com/motdotla/dotenv.

9. By default locally, you will be logged with `admin` access. If you want to change it, need to set `RSSHCOOL\_DEV\_ADMIN` to `false` in `.env` file

\*\*IMPORTANT:\*\* Never commit changes to `.env` file

10. Do not forget to write [Jest](https://facebook.github.io/jest/) specs for your feature following [Specs Styleguide](#specs-styleguide)

11. Make sure specs, lints pass and code formatted properly (they'll be run on a git pre-commit hook too)

``` command-line

$ npm test

$ npm run lint

$ npm run pretty

```

12. Commit your changes using a descriptive commit message that follows our [commit message conventions](#git-commit-messages)

``` command-line

$ git commit -m "feat: implement feature X"

```

13. Push your branch to GitHub:

``` command-line

$ git push origin feature-x

```

14. Create a pull request

### Pull Requests

\* Check how to create a [pull request](https://help.github.com/articles/creating-a-pull-request/)

\* Send a pull request to `master` branch

\* Write a meaningful description

\* Include screenshots and animated GIFs in your pull request whenever possible

## Styleguides

### Git Commit Messages

\* Use [Conventional Commits](https://conventionalcommits.org/) format

\* Allowed Types:

\* build: - \*changes that affect the build system or external dependencies (example scopes: npm, webpack)\*

\* ci: - \*changes to our CI configuration files and scripts (example scopes: drone)\*

\* docs: - \*documentation only changes\*

\* feat: - \*a new feature\*

\* fix: - \*a bug fix\*

\* perf: - \*a code change that improves performance\*

\* refactor: - \*a code change that neither fixes a bug nor adds a feature\*

\* style: - \*?hanges that do not affect the meaning of the code (white-space, formatting, missing semi-colons, etc)\*

\* test: - \*adding missing tests or correcting existing tests\*

\* Use the present tense ("add feature" not "added feature")

\* Use the imperative mood ("move cursor to..." not "moves cursor to...")

\* Limit the first line to 72 characters or less

\* Reference issues and pull requests liberally after the first line

### TypeScript Styleguide

We use Prettier for TypeScript formatting. Please run the following command before your commit:

``` command-line

npm run pretty

```

For your convience, you can integrate Prettier into your favorite IDE (https://prettier.io/docs/en/editors.html)

### Specs Styleguide

- Name spec file by adding `.spec` to the name of tested file.

Example:

```

foo.ts

foo.spec.ts // spec file for foo.ts

```

- Treat `describe` as a noun or situation.

- Treat `it` as a statement about state or how an operation changes state.

Example:

```javascript

describe('Header', () => {

it('shows username', () => {

//...

})

it('shows logout button', () => {

//...

})

})

```