# Contributing guidelines

## Before you send a pull request

1. \*\*Did you add or changed some functionality?\*\*

Add (or modify) tests!

2. \*\*Check if the automated tests pass\*\*

```bash

yarn ci:check

```

3. \*\*Format the files you changed\*\*

```bash

yarn prettier

```

4. \*\*Mark your changes in CHANGELOG\*\*

Put a one-line description of your change under Added/Changed section. See [Keep a Changelog](https://keepachangelog.com/en/1.0.0/).

## Running Watermelon in development

### Download source and dependencies

```bash

git clone https://github.com/Nozbe/WatermelonDB.git

cd WatermelonDB

yarn

```

### Developing Watermelon alongside your app

To work on Watermelon code in the sandbox of your app:

```bash

yarn dev

```

This will create a `dev/` folder in Watermelon and observe changes to source files (only JavaScript files) and recompile them as needed.

Then in your app:

```bash

cd node\_modules/@nozbe

rm -fr watermelondb

ln -s path-to-watermelondb/dev watermelondb

```

\*\*This will work in Webpack but not in Metro\*\* (React Native). Metro doesn't follow symlinks. Instead, you can compile WatermelonDB directly to your project:

```bash

DEV\_PATH="/path/to/your/app/node\_modules/@nozbe/watermelondb" yarn dev

```

### Running tests

This runs Jest, ESLint and Flow:

```bash

yarn ci:check

```

You can also run them separately:

```bash

yarn test

yarn eslint

yarn flow

```

### Editing files

We recommend VS Code with ESLint, Flow, and Prettier (with prettier-eslint enabled) plugins for best development experience. (To see lint/type issues inline + have automatic reformatting of code)

## Editing native code

In `native/ios` and `native/android` you'll find the native bridge code for React Native.

It's recommended to use the latest stable version of Xcode / Android Studio to work on that code.

### Integration tests

If you change native bridge code or `adapter/sqlite` code, it's recommended to run integration tests that run the entire Watermelon code with SQLite and React Native in the loop:

```bash

yarn test:ios

yarn test:android

```

### Running tests manualy

- For iOS open the `native/iosTest/WatermelonTester.xcworkspace` project and hit Cmd+U.

- For Android open `native/androidTest` in AndroidStudio navigate to `app/src/androidTest/java/com.nozbe.watermelonTest/BridgeTest` and click green arrow near `class BridgeTest`

### Native linting

Make sure the native code you're editing conforms to Watermelon standards:

```bash

yarn swiftlint

yarn ktlint

```

### Native code troubleshooting

1. If `test:ios` fails in terminal:

- Run tests in Xcode first before running from terminal

- Make sure you have the right version of Xcode CLI tools set in Preferences -> Locations

1. Make sure you're on the most recent stable version of Xcode / Android Studio

1. Remove native caches:

- Xcode: `~/Library/Developer/Xcode/DerivedData`:

- Android: `.gradle` and `build` folders in `native/android` and `native/androidTest`

- `node\_modules` (because of React Native precompiled third party libraries)