Overview

Builder.AI applications are developed with a common architecture and infrastructure. This document describes the standard procedures necessary to install your project and provides an overview of code design. Installation, administration, and maintenance should be performed by qualified professionals.

Platforms

Our Front End projects are written in React.js for websites and React Native for mobile apps like iOS and Android. For Back End projects, we use Ruby on Rails (RoR). If any data needs to be stored, we use a PostgreSQL database.

- **Ruby on Rails:** We use the Ruby on Rails framework as a back-end platform for all RESTful services. This is exposed as an API which is consumed by the front-end Mobile/Web applications. The Rails application also serves an admin console that provides control and administration features for your system.
- **React Native:** We build our web and mobile applications on the React/React Native frameworks.

Building Block Approach

Builder.ai follows the "Building Block" or the "Block Approach" towards software development.

Installation

Installation, administration and maintenance should be performed by qualified professionals.

Back-end Ruby Installation

You might need to install following packages to the system before starting the application:

- bash
- build-base
- libxml2-dev
- libxslt-dev
- postgresql
- postgresql-dev
- nodejs
- vim
- yarn
- libc6-compat
- curl
- git
- which
- wkhtmltopdf
- ttf-ubuntu-font-family
- imagemagick

See the `Dockerfile` in the ruby project for further run-time environment details.

Setup the Ruby environment

Install rvm

Install Ruby 2.7.5:

```
rvm install "ruby-2.7.5" /bin/bash --login rvm
use 2.7.5 --default
```

Install bundler

```
gem install bundler
```

Install Postgresql, create pgsql user, create database:

And change the settings in `template-app/config/database.yml` accordingly.

Install minio and change the settings in `template-app/config/storage.yml`

Install Nginx:

```
sudo apt install nginx
```

Then change the config in `/etc/nginx/nginx.conf`. (See here for further details).

We need 3 subdomains: www, api and minio.

For `www`, set the webroot to point to the folder which includes the react-native build.
For `api`, set the webroot to point to the ruby on rails server port: default 3000

For `minio`, set the webroot to point to the minio server port: default 9000

After completing these steps, ensure that nginx has restarted: `sudo systemctl restart nginx`

Running the ruby application:

From the project repository cd into template-app

Install the gems with:

```
bundle install
```

To initialise the database:

```
bundle exec rails db:migrate
```

Run the ruby on rails server

```
bundle exec rails server
```

For Sidekiq

```
bundle exec sidekiq
```

Front-end (FE) installation

Please make sure, NodeJS LTS version along with yarn node module is installed on the system before performing the below steps.

1. Update backend URL in `/packages/framework/src/config.js` by replacing `__MARKER_FOR_BACKEND_URL_REPLACEMENT` with the backend URL.
2. Go to `<root folder>/src` and run

   ```
   yarn install
   ```

3. To run the application, run the following command from the `src` folder

   ```
   yarn workspace web start
   ```

   There are 2 ways to deploy the application on the server

4. Using project Build (Recommended)
   a. Build the project by running

   ```
   yarn workspace web build
   ```

   b. Now you can deploy the `/packages/web/build` folder to the server.

5. PM2 (alternative)
   a. Install pm2 as global node module to the system

```
yarn global add pm2
```

Or

```
npm install -g pm2
```

b. Run the following command from the root folder of the project

```
pm2 start --name <app name> yarn workspace web start
```

Troubleshooting

Code has been shipped with node_module libraries included. Depending on your local OS and tool versions, (e.g. if you encounter a build error `ERR_OSSL_EVP_UNSUPPORTED` you may need to try the following to ensure a smooth front-end installation:

```
cd src
yarn cache clean
rm -rf node_modules
export NODE_OPTIONS=--openssl-legacy-provider
```

Followed by the usual

```
yarn install
yarn workspace web build
```

You may also need to downgrade your version of node to v18.x.x, although code should even run on v16.x.x.

Development: Getting Started

React Native

The first step to starting block development is to make sure you have the following prerequisite tools.

Prerequisites

Homebrew

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/ins
```

Git

```
brew install git
```

Node

```
brew install node
```

Yarn

```
brew install -g yarn
```

Watchman

```
brew install watchman
```

ReactNative

```
$ sudo npm install -g react-native-cli
```

JDK 13 & JDK 11

```
$ brew tap homebrew/cask-versions
$ brew cask install java
$ brew cask install java11
$ brew cask install java13
```

Android Studio

https://developer.android.com/studio/install.html

XCode

https://itunes.apple.com/us/app/xcode/id497799835?mt=12

Android SDK

```
$ brew cask install android-sdk
```

XCode Command Line Tools

```
$ xcode-select --install
$ sudo gem install cocoapods
```

Recommended Tools

SourceTree

https://www.sourcetreeapp.com/

Postman

https://www.postman.com/

Visual Studio Code (or your IDE of choice)

https://code.visualstudio.com/

Third Party Inclusions (Libraries)

Ruby on Rails

Gems

Please check the ruby project's `template-app/Gemfile` and `template-app/Gemfile.lock' for required gems and dependencies

React Native

Please check the `src/package.json` and `src/package-lock.json` files for required packages and dependencies. Individual blocks have further package dependency definitions. Block structure is discussed further below.

Supplemental Materials

A video describing the front-end react/react-native code and project
A walkthrough document for our back-end Ruby on Rails code
A video describing the back-end code and project