Ticket Booth

Table of Contents

Test & Lint Status
Build & Packaging Status


Please see the README.pdf for the PDF version of this README.

Please see the Acknowledgements at the end of this page.

Welcome to the Ticket Booth!

The goal of the app is to make ticket and volunteer management for community events easier and automated.

Production Deploy, as of 2024

To deploy to production site — https://tickets.fnf.events — the deployer's IP address must be white-listed with EC2. Contact Konstantin Gredeskoul to get white-listed.

The deploy is performed via Capistrano:

```
git checkout main
git pull
bin/dev-tooling # installs local dev dependencies
bin/dev-setup   # installs Ruby, Node, Gems, etc.
bin/deploy      # actually performs the deployment
```

Development Environment Setup

The following walks through a local setup on OS-X M1.

Streamlined Setup

If you installed Homebrew on your laptop, you should be able to boot the app.

Optional VIM and PostgreSQL Local Configuration

We provided a pretty comprehensive VIM configuration with auto-complete, as well as the `psql` configuration with a prompt and additional useful macros.

To install this, run

```
bin/dev-tooling
```

After that, your `vim` sessions will have auto-complete enabled, and your `psql -U postgres` sessions will have rich prompt.

Running App Dependencies Installer

You can run the following setup script, but only on OS-X, to attempt a complete set up of the development environment, as well as the installation of the Rubies, Gems and Database:

```
bin/dev-setup
```

This should install all of the Brew dependencies, start PostgreSQL, memcached, and install Ruby, Node, Yarn, Gems and Node packages.

Starting the Rails Server

```
bin/dev-start
```

This actually starts Foreman via `bundle exec foreman -f Procfile.dev` — this is required to start CSS and JS just-in-time compilcation in addition to the Rails server.

The server will run on port 8080, and in development will hot swap any locally modified files, including CSS and JS.

Running `rails s` is no longer sufficient to start the application.

Running Tests and Linters

To verify that your local environment is working, run the following:

```
make ci
```

This will run DB Migrations, followed by RSpec, Rubocop, and ShellCheck.

Additional Information

We dedicated a separate document to the developer setup, which helps you get the application running locally.

Alternatively, keep reading for step-by-step manual instructions.

Optional Manual Setup

If you prefer to run all the steps manually, then follow the guide below.

Manual 1: Services

Please make sure you have PostgreSQL and running locally, or install it via Homebrew:

```
brew install direnv

brew install postgresql@16
brew services postgresql@16 start

brew install memcached
brew services memcached start
```

Manual 2: Direnv Setup

Before you can start the Ruby Server, you need to configure `direnv` so that the environment in the file `.envrc` is loaded on OS-X.

To do that follow the instructions for setting direnv on bash or zsh depending on what you are running. To find out, run `echo $SHELL`.

After you setup the shell initialization file, restart your terminal or reload the shell configuration.

Once you are back in the project's folder, run:

```
eval "$(direnv hook ${SHELL/*\/})"
direnv allow .
```

the first line above should be copied to your shell RC (aka "dotfiles").

This will load the environment variables from the `.envrc` file.

Manual 3: NodeJS & Votal Setup

Run the following to get Volta Node Manager working:

```
curl https://get.volta.sh | bash
volta install node@lts
volta install yarn
volta pin node yarn
```

Now your Node & Yarn should be installed.

Manual 3: Ruby Setup

```
# install brew from https://brew.sh
brew bundle 2>/dev/null

# ensure the following packages exist
brew install rbenv ruby-build direnv volta

eval "$(rbenv init -)"

rbenv install -s $(cat .ruby-version)
rbenv local $(cat .ruby-version)

bundle install -j 12
rails db:prepare
rails db:test:prepare

# Run Specs at the end
bundle exec rspec

# Run rubocop
bundle exec rubocop

# Run ShellCheck
bin/shchk
```

Manual 4: Starting the Server

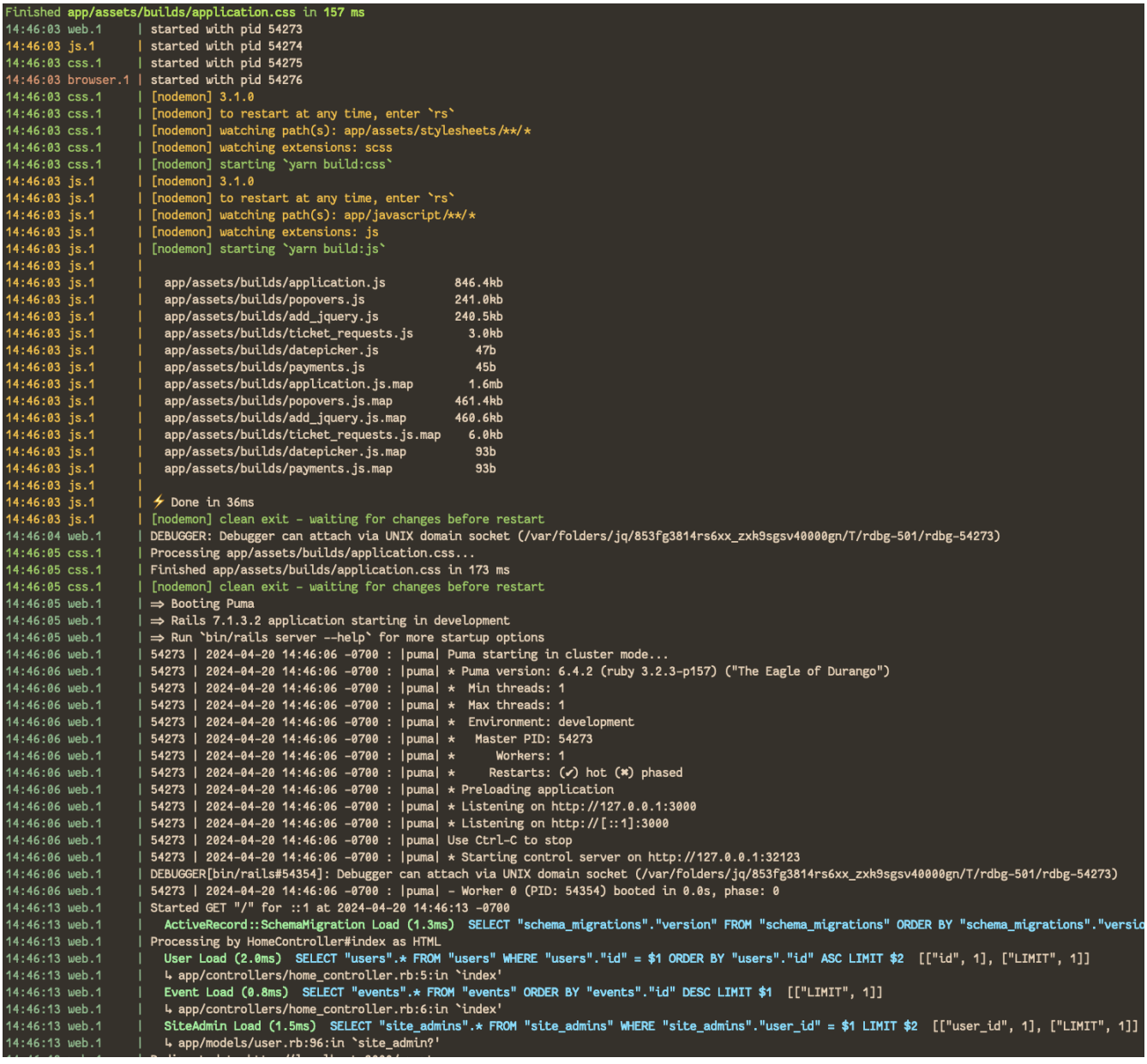
To start the server post-setup, run the following (NOTE: you must start the server via Foreman, since it also starts `yarn` tasks that monitor and dynamically recompile CSS and JS assets)

```
bin/dev-start
```

Or manually:

```
bundle exec foreman -f Procfile.dev
```

Here is an example:



API Testing

HTTP API Specs use the VCR Gem to mock calls to external APIs using a record and replay model. https://github.com/vcr/vcr

Usage: https://benoittgt.github.io/vcr/#/

Cassettes are stored in spec/fixtures/vcr_cassettes If an API changes due to version, response, etc...

you will need to rebuild cassettes for those specs. Delete the directory and/or files for the specs that have changed. It is ok to delete all cassettes and regenerate everything. This can be done in your local development environment.

VCR is configured in spec/spec_helper.rb

You must filter any API keys before you check in cassettes to prevent keys in GitHub https://benoittgt.github.io/vcr/#/configuration/filter_sensitive_data

To enable vcr recording on a given spec, add a vcr hook to the spec as follows

```
it 'does not change payment intent', :vcr do
    expect { payment_intent }.not_to(change(payment, :payment_intent))
end
```

To turn off VCR HTTP request interception for a given spec or block, add

```
VCR.turned_off do
    make_request "In VCR.turned_off block"
end
```

```
make_request "Outside of VCR.turned_off block"
```

```
VCR.turn_off!
make_request "After calling VCR.turn_off!"
```

```
VCR.turned_on do
    make_request "In VCR.turned_on block"
end
```

```
VCR.turn_on!
make_request "After calling VCR.turn_on!"
```

https://benoittgt.github.io/vcr/#/cassettes/no_cassette

To turn off VCR by default for http requests see: https://benoittgt.github.io/vcr/#/configuration/allow_http_connections_when_no_cassette

Adding Site Admin

When the database is completely blank, the first step is to create the initial account. Lets say you registered as 'kig@fnf.org':

The second step is to make that person a site admin:

```
RAILS_ENV=production
bin/site-admin add kig@fnf.org

# Or, to remove site admin from a given user:
bin/site-admin remove kig@fnf.org
```

Generating Music Submissions List

The repo contains a convenient script for generating HTML to embed into the Wordpress site, using a CSV generated out of Google Spreadsheet collected using Google Forms.

The CSV must contain three columns and a header row:

- DJ Name
- Full Name
- Set URL

To generate the HTML (we'll use the CSV file checked into the fixtures):

```
# eg, using the fixture file:
$ bin/music-submission-links spec/fixtures/chill_sets.csv > chill_set.html

# or, to include the simple CSS into the header:
$ bin/music-submission-links spec/fixtures/chill_sets.csv --simple-css >
chill_set.html
open chill_set.html
```

If you add `--simple-css` to the arguments, the generated HTML will include `<head>` element with the Simple CSS Stylesheet. Do not use this flag if you plan to paste the output into the WordPress text box. Use this flag if you simply want to verify the resulting HTML in a browser by running `open chill_set.html`.

To verify that the script is working and generating correct HTML, you might want to install a handy tool called `bat`, eg using Homebrew on Mac OS-X:

```
$ brew install bat
$ bin/music-submission-links spec/fixtures/chill_sets.csv | bat
```

Adding Submissions to WordPress

Now you can open WordPress, create a two-column layout on the submissions page and paste the

contents into one of the two columns, typically:

1. Night time / Peak Hour
2. Chill / Daytime

First, let's copy the resulting HTML into clipboard:

```
$ bin/music-submission-links chill_sets.csv | pbcopy
```

Now we can paste it into WordPress directly.

API Documentation

Yard-generated documentation is available via running:

```
$ bundle exec rake doc
# this will automatically open the index.html
```

Acknowledgements

This app is formerly known as **Helping Culture**, which in turn was originally conceived and inspired by Tracy Page.

This project was originally written by Shane de Silva.

It is currently maintained by the FnF org, and within it specifically

- Konstantin Gredeskoul for any Rails, Ruby, or application issues,
- Ryan Shatford and Mike Matera for any issues related to deployment to the Google Public Cloud.
- Matt Levy for development, coordination and project management.

Please use labels to tag any reported issues.