# gdaps-frontend-vue

**Christian González**

**Jan 10, 2021**

# TABLE OF CONTENTS

This is a GDAPS plugin, which provides a Vue.js frontend support to any GDAPS/Django application

CHAPTER

ONE

# INSTALLATION

For GDAPS install see https://gdaps.readthedocs.org.

Install **gdaps-frontend-vue** in your Python virtual environment:

```
pip install gdaps-frontend-vue
```

This will automatically install the package, including necessary dependencies, like webpack_loader.

Next, `webpack_loader` must be added to `INSTALLED_PLUGINS`:

```
# settings.py

from gdaps.pluginmanager import PluginManager

INSTALLED_APPS = [
    # ... standard Django apps and GDAPS
    # this must be placed *before* gdaps as it overrides gdaps' managemant commands:
    "gdaps.frontend"
    "gdaps",
    "webpack_loader",
]
INSTALLED_APPS += PluginManager.find_plugins("myproject.plugins")
```

Now, to satisfy webpack-loader, add a section to settings.py. Django(-webpack-loader) needs to find the info file webpack creates at each compile (webpack-stats.json), so that files can be recognized by Django's reloading mechanism.

**Note:** `webpack-bundle-tracker >=1.0.0` changed the webpack-stats.json format, so django-webpack-loader is currently not compatible with it, so you have to use a compatibility LOADER_CLASS.

```
WEBPACK_LOADER = {
    'DEFAULT': {
        'STATS_FILE': os.path.join(BASE_DIR, "frontend", 'webpack-stats.json'),
        'LOADER_CLASS': 'gdaps_frontend_vue.webpack.CompatibilityWebpackLoader',
    }
}
```

You should already have added GDAPS' URL path:

```
# urls.py
from gdaps.pluginmanager import PluginManager

urlpatterns = PluginManager.urlpatterns() + [
```

(continues on next page)

(continued from previous page)

```
    # ... add your fixed URL patterns here, like "admin/", etc.
]
```

Now you can initialize the frontend with

```
./manage.py initfrontend
```

This creates a basic boilerplate (previously created with 'vue create' and calls *yarn install* to install the needed javascript packages.

CHAPTER

TWO

# USAGE

## 2.1 Initializing the frontend

```
./manage.py initfrontend
```

This creates a /frontend/ directory in the project root, and installs a Vue.js frontend application there. The type of frontend depends on what you have selected in *GDAPS["FRONTEND_ENGINE"]*: Vue, PySide are available ATM.

**Vue.js** It is recommended to install vue globally, you can do that with `yarn global add @vue/cli @vue/cli-service-global`. GDAPS tries to do that for you when you call `./manage.py initfrontend`.

Now you can start `yarn serve` (or `npm run serve`, depending on your choice) in the frontend directory. This starts a development web server that bundles the frontend app using webpack automatically. You then need to start Django using `./manage.py runserver` as usual to enable the Django backend. GDAPS manages all the needed background tasks to transparently enable hot-reloading when you change anything in the frontend source code now.

## 2.2 Frontend plugins

Django itself provides a template engine, so you could use templates in your GDAPS apps to build the frontend parts too. But templates are not always the desired way to go. Since a few years, Javascript SPAs (Single Page Applications) have come up and promise fast, responsive software.

But: a SPA mostly is written as monolithic block. All tutorials that describe Django as backend recommend building the Django server modular, but it should serve only as API, namely REST or GraphQL. This API then should be consumed by a monolithic Javascript frontend, built by webpack etc. At least I didn't find anything else on the internet. So I created my own solution:

GDAPS is a plugin system. It provides backend plugins (Django apps). But using `gdaps.frontend`, each GDAPS app can use a *frontend* directory which contains an installable module, that is automatically installed when the app is added to the system.

When the `gdaps.frontend` app is activated in `INSTALLED_APPS`, the `startplugin` management command is extended by a frontend part: When a new plugin is created, a *frontend/myproject-plugin-fooplugin* directory wth some boilerplate files in that plugin is created. The `index.js` file is the plugin entry point for the frontend.

So all you have to do is:

1. Add `gdaps.frontend` to `INSTALLED_APPS` (before `gdaps`)
2. Call `./manage.py initfrontend`, if you haven't already
3. Call `./manage.py startplugin fooplugin`
4. start `yarn serve` in the *frontend* directory

5. start Django server using `./manage.py runserver`

To remove a plugin from the frontend, just remove the backend part (remove it from INSTALLED_APPS or uninstall it using pip) and call `manage.py syncplugins` afterwords. It will take care of the database models, and the npm/yarn uninstallation of the frontend part.

## 2.3 Creating plugins

When `gdaps-frontend-vue` is installed, you can create plugins using a Django management command as usual:

```
./manage.py startplugin fooplugin
```

This command asks a few questions, creates a basic Django app in the plugin path chosen in `PluginManager.find_plugins()`. It provides useful defaults as well as a setup.py/setup.cfg file. Additionally, it creates a frontend directory which then can be consumed by your frontend.

CHAPTER

THREE

# API

## 3.1 Vue.js Frontend

CHAPTER

FOUR

# CONTRIBUTING

This is an Open Source project. Any help, ideas, and Code are welcome. If you want to contribute, feel free and write a PR, or contact me.

## 4.1 Code style

No compromises. Format your code using Black before committing.

CHAPTER

FIVE

# LICENSE

I'd like to give back what I received from many Open Source software packages, and keep this library as open as possible, and it should stay this way. GDAPS is licensed under the General Public License, version 3.