AACE Platform

Project diary

This document will contain all the steps and processes involved in building the AACE platform. This is intended to be used as a handbook that will serve as an entry point for future references. More documents will be attached for a complete overview.

In addition to reporting on the steps taken for this particular project, this document also seeks to be a guide for further projects. For AACE we have tried to generalize the process of building an API based backend for any web or mobile application.

# Project scaffolding

The scaffolding will include documentation on all steps made that directly or indirectly include project layout, structure or any other architectural decisions.

## 1.1 Scaffolding API Flask application

The first step to take when dealing with a new project that will require Flask programming, is to open a repository in BitBucket or any other code management tool. Since we use BitBucket, we open a new project.



Fig 1.1

AACE project repository in Bitbucket

For this kind of job, it makes sense to open a project repository for two reasons:

1. All team members can have access to the code and therefore can perform operations from their own accounts
2. The project might require different repositories for different aspects of the work (Mobile, Web, Landing Page, etc)

The first repository is named ‘aace-platform’. The repository is cloned locally and the first commit is a folder containing, among other things, this document. It is advisable that every project repository is accompanied by a report of some sort that serves as a guide to the code.

The second addition to the project that is suggested at the time of writing is to include a .gitignore file. This file needs to be added before scaffolding a Flask project to avoid complications with caching and other issues that follow mid-project .gitignore tentatives. This file will be copied from the Flusk repository where we will adapt the general architecture.

The third addition is to add a virtual environment for python. Having an isolated environment protects the application from different requirements among projects residing in the local development machines.

The resulting folder should look similar to Fig 1.2.

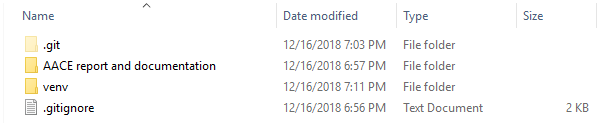


Fig 1.2

Documentation, venv and gitignore

Having prepared a clean environment, we are ready to proceed with the architecture of the application itself. For further reference, most of the steps taken in this report are directly or indirectly influenced by the famous Grinberg tutorial on Flask. This point is also appropriate for a commit with a clean description, to easier track changes and progress.

Now activate the virtual environment and install only the necessary packages, to avoid conflicts and bloated space requirements. So far we only need the flask package, done with ‘pip install flask’. This will install 6 required packages for a basic flask application. These packages need to be saved in a requirements.txt file, so we execute ‘pip freeze > requirements.txt’. The result should look similar to Fig 1.3.

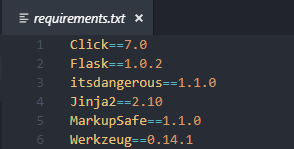


Fig 1.3

Initial pip freeze requirements

Flask applications exist in packages. A package is a sub-directory that includes an ‘\_\_init\_\_.py’ file. A package may contain many modules, which are usually python files. The init file defines what inside the package is exposed to outer uses. Our package directory will be called ‘aace’. We create this directory and the init file in it.

*from* flask *import* Flask

app = Flask(\_\_name\_\_)

What needs to be considered here is that putting ‘\_\_name\_\_’ as the argument to Flask() usually configures the application instance right. The created app object now holds our Flask application.

To complete the basic layout, we need to create some environmental variables. We first install the dotenv python package using: pip install python-dotenv. Afterwards update the requirement file with pip freeze.

Create a .flaskenv file where we will store our env variables to configure our application. Create a run.py file in the main directory. This file needs to contain only one line for a basic functioning:

*from* aace *import* app

Write down the following on the .flaskenv file:

FLASK\_APP=run.py

You can now execute ‘flask run’ on the terminal. The application is working. This is another good milestone that needs to be clearly committed.