Heroku deployment

We'll work with a sample application (https://github.com/carlesm/minidjangoapp/), we can either download AS ZIP the repo, or fork it to have our own cloned repo.

Once we have the web application, we must ensure that the requirements.txt file is right, we will need, at least: pycopg2, dj-database-url, and gunicorn (besides Django, obviously).

So we check our requirements.txt to have at least:

```
Django==1.11.10
psycopg2==2.7.3.1
dj-database-url
gunicorn
```

We then add a new file, on the project root (where manage.py resides), called Procfile (careful with capitals), with:

```
web: gunicorn tapapp.wsgi --log-file -
```

This will tell heroku how to run our program, in this case, by using gunicorn with the wsgi package inside our project directory (tapapp). And to send all logs to standard output ("-"), so heroku can capture it.

We also add a file to tell Heroku that our project is a python one, and to tell it which python version to use, in our case, to be coherent with the labs and with docker deployment, 2.7.14. The file should be called Pipfile, careful with capitalization, containing:

```
[requires]
python_full_version = "2.7.14"
```

Only if starting with a ZIP not a git fork:

We will need our code to be in a git repository, so we'll create one:

```
git init .
git add *
git commit -a -m "Initial commit"
```

Then, or if we forked the github repo, we can create a heroku application:

```
heroku create
```

That will add a new git remote repository, we can check with:

```
git remote -v
```

We can deploy to heroku, then, with:

```
git push heroku master
```

We can run into an error concerning static files, we will fix it correctly later, for now, we can "skip" the error with:

```
heroku config:set DISABLE_COLLECTSTATIC=1
```

We can then connect, if all goes ok to our app with:

```
heroku open
```

If we run into errors with hosts not in ALLOWED_HOSTS, you can fix it by editing settings.py, and adding the suggested host in the error page to ALLOWED_HOSTS variable.

```
ALLOWED_HOSTS = ['*****']
```

We the do the:

```
git add tapapp/settings.py
git commit -m "Fixed"
git push heroku master
```

If the connection is ok we'll probably get an error related to missing tables. We should correct it as we'd do when locally developing with django, but running the migrations in heroku not locally:

```
heroku run python manage.py migrate
heroku run python manage.py createsuperuser
heroku open
```

Static files the right way

Add to MIDDLEWARE on settings.py:

```
'whitenoise.middleware.WhiteNoiseMiddleware',
```

On the STATIC_URL on settings.py substitute it by:

```
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/1.11/howto/static-files/

STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')

STATIC_URL = '/static/'
# Extra places for collectstatic to find static files.
STATICFILES_DIRS = (
  os.path.join(BASE_DIR, 'static'),
  )
  STATICFILES_STORAGE = 'whitenoise.django.GzipManifestStaticFilesStorage'
```

And to requirements.txt add:

```
whitenoise
```

Next we send the files to heroku again:

```
git add tapapp/settings.py
git add requirements.txt
git commit -m "Added whitenoise"
git push heroku master
```