Manual de usuario
App Server
75.52 - Taller de programación II

Integrantes:

- Gismondi, Maximiliano Ezequiel - Padrón: 89957
- Lavandeira, Lucas - Padrón: 98042
- Alvarez Avalos, Dylan Gustavo - Padrón: 98225
- Blanco, Sebastian Ezequiel - Padrón: 98539

Ayudante: Calónico, Christian Mario

Año: 2018

Cuatrimestre: 2°

Fecha de entrega: 6 de Diciembre 2018

Introducción

El presente documento informa los procesos necesarios para instalar, configurar, y probar el application server de **Comprame**. Es el backend usado por la aplicación Android del proyecto.

El servidor se encarga del manejo de todas las entidades del sistema, salvo pagos y envíos, que son delegados otro servidor (**shared-server**). Por ejemplo, maneja usuarios, artículos, categorías, compras, ventas, preguntas, respuestas, mensajes de chat.

El stack de tecnologías manejado es: Python como lenguaje de programación, y Flask como servidor web. Se arma un servicio REST sobre Flask, y se persisten los datos en una base de datos NoSQL (MongoDB). El servidor puede ser cualquier entorno UNIX-like que permita instalar correctamente las tecnologías mencionadas.

Opcionalmente se puede utilizar Docker para manejar la instalación de la aplicación automáticamente.

1 - Instalación de dependencias (sin Docker)

A continuación se mencionan las herramientas a instalar necesarias para poder hacer uso del proyecto

1. Python

Para el lenguaje, se recomienda utilizar **pyenv** para instalar y manejar un entorno virtual.

- Instalar Python 3.6.6: pyenv install 3.6.6
- Configurar un entorno virtual para el proyecto: pyenv virtualenv 3.6.6 taller2
- Crear un archivo `.python-version` en el root del directorio del proyecto con contenidos `taller2` para que el entorno se active automáticamente al entrar al directorio.
- Instalar dependencias: `pip install -r requirements.txt`

2. MongoDB

Para MongoDB, se recomienda utilizar **Docker**. El proyecto ya viene preconfigurado con una instancia de Mongo para ser levantada con `docker-compose: docker-compose up -d mongodb`. Si no se cuenta con Docker, referirse a la documentación de mongodb para configurarlo de manera nativa.

2 - Levantar proceso

2.1 - Levantar la aplicación sin docker

- Levantar la aplicación: `sh scripts/run.sh`
- Con gunicorn: `sh scripts/wsgi.sh`

2.2 - Testing

- Flake8 (validador de PEP8 + extras): `sh scripts/flake8.sh`
- Pylint: `sh scripts/pylint.sh`
- Correr tests automáticos: `sh scripts/tests.sh`

2.3 - Levantar la aplicación con docker

Escribimos en una terminal situada en la raíz del repositorio lo siguiente:

- Instalar dependencias: `pip install -r requirements.txt`
- Levantar aplicacion: `$ docker-compose up --build`

3 - Requerimientos mínimos

Para manejar un uso ligero de una baja cantidad de usuarios concurrentes, el servidor puede funcionar correctamente con la siguiente configuración

- Sistema operativo: Alpine Linux o similar con bajo uso de memoria (minimalista)
- Ram: 256 MB
- Cantidad de cores: 1

Para soportar una mayor cantidad de usuarios, se debería modificar la configuración de gunicorn para utilizar más procesos *worker* que 4, aumentar la RAM y la cantidad de núcleos de procesador.

4 - Debug

- Usar `pycharm` y ejecutar la aplicación bajo la configuración `run flask app`. Luego desde el entorno de desarrollo se pueden configurar *breakpoints* y correrse bajo un modo debug.

5 - Recursos usados

La aplicación utiliza el puerto `8000`. Para acceder al servidor, una vez levantado, se puede acceder a través de la url `http://0.0.0.0:8000/`.

Mongodb utiliza el puerto 27017 por defecto.

6 - Entorno productivo

El despliegue de la aplicación está automatizado: el pipeline del mismo lo conforman el repositorio GitHub, el servicio de integración contínua Travis y la plataforma Heroku.

Para realizar un deploy a dicho entorno se debe pushear un nuevo commit a la rama master. El resto del proceso es automático: se validarán las reglas de estilo del código y se correrán los tests. Si el resultado de ambos procesos es exitoso, se realizará el deploy y una vez finalizado podremos acceder a la nueva version en la siguiente direccion: https://taller2-app-server.herokuapp.com

7 - Configuración de la aplicación

La configuración entera puede encontrarse en los módulos dentro del paquete `conf` del proyecto. Existen cuatro variables a considerar:

- DEBUG: Ejecutar flask en modo debug. Booleano. **No correr con True en entornos productivos!**
- MONGO_URI: URI a la base de datos de mongodb. String.
- SKIP_AUTH: No validar la autenticación de usuarios provistas por Firebase Authentication. Booleano. **No correr con True en entornos productivos!**
- FIREBASE_API_KEY: String con el JSON de la API Key de Firebase Cloud Messaging.

Estas variables son seteadas a través de código Python, que para los valores no booleanos se leen desde variables de entorno, con valores default en caso de no existir, útiles para el desarrollo local. En el ambiente productivo (Heroku) se encuentran configuradas en el panel de administrador los valores de las variables de entorno necesarias para el funcionamiento adecuado del sistema.

Para que los request al shared server que requieren autenticación sean exitosos, se debe cargar un app server con nombre y password en shared server, y luego se deben setear las siguientes variables de entorno:

- APP_SERVER_NAME: nombre de la base de datos en shared server
- APP_SERVER_SECRET: password de la base de datos en shared server