Tecnicatura Universitaria en Programación

PROGRAMACION I

Trabajo Práctico Integrador

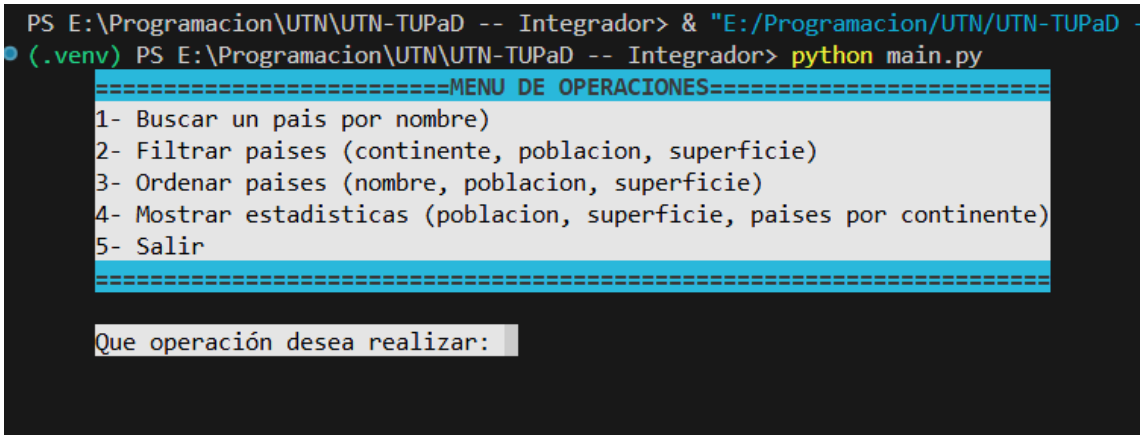
Comisión: 1 prog 2

Integrantes: Morillas Andrés, Ciro Cattaneo

Informe teórico:

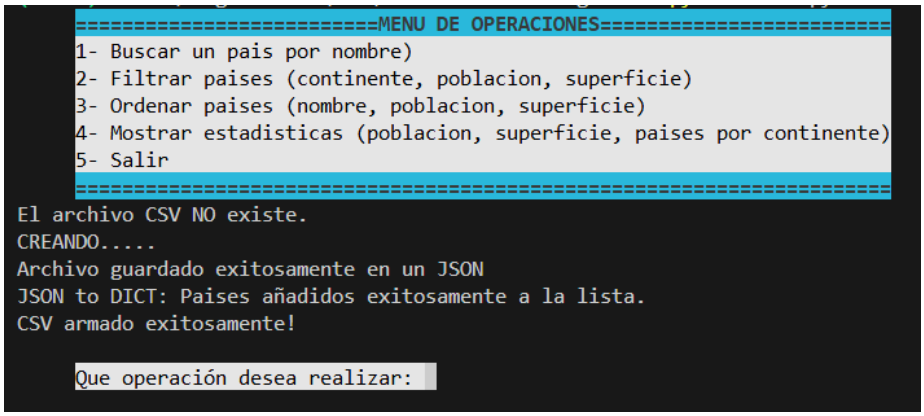
Primer paso – Creación de archivo .CSV:

Al momento de desarrollar el proyecto en cuestion, se implementaron todos los conceptos basicos desarrollados y explicados en la materia hasta el día de la fecha. En una aproximación global e inicial se busco, implementar la persistencia de un menú, a través de bucles con salida por parte del usuario; para implementar 4 funciones principales, encargadas de implementar subfunciones particulares. Cada subfunción, en terminos generales, buscaba hacer un manejo de listas de diccionarios, utilizando en ocasiones las keys de los distintos campos, y en otras pasando los valores de dichas keys a strings, o integers que permitían calculos o verificaciones sencillas.



Vista inicial del menú.

Entrando más en detalle, al requerirse la implementación y uso de un archivo .CSV para almacenar la informacion de paises del mundo; se tuvo que construir una funcion inicial *(api_build.py)* para encargarse de verificar la existencia de un archivo .CSV; y dada la negativa hacer un llamado a la API para que lo cree.



Confirmación de la creación de un archivo .CSV

Tras una breve verificación de la existencia o no de los archivos, mediante la librería OS; la funcion que llamar a una API de http://restcountries.com a través de la funcion .get() proveniente de la librería Requests.

Tras leer la documentación que ofrecia, se indica que para obtener campos específicos, sin traer toda la informacion completa, se pueden utilizar parámetros denominados “fields”, los cuales limitan a requests.get() a que obtenga unicamente campos seleccionados por nosotros.

Dicho método genera un archivo .JSON, el cual va a brindar todos los datos obtenidos. Sin embargo, al seguir siendo un archivo que no es el apropiado para el manejo de datos dentro de nuestro sistema, se generó una segunda funcion para convertir dicho archivo en una lista, donde cada pais es un diccionario. Es alli donde se hizo la conversión apropiada de datos, para poder generar un archivo .CSV con las condiciones fijadas por el trabajo integrador. Es decir, con nombre, población, superficie y continente.

Finalmente, una tercera funcion, constituiría el archivo .CSV usando los metodos de la librería CSV. Es decir el método csv.DictWriter()

Las tres funciones, serian reunidas dentro de un solo archivo; el cual se denominaría api_build.py, con lo cual conceptualmente todo lo relacionado a la generación de .CSV mediante APIs, existiría unicamente aquí.

Como mencione inicialmente; en el caso de que inicialmente se compruebe la existencia de un archivo .CSV en la raíz del proyecto; esta funcion no se ejecuta. Encontré que no sería necesario imprimir por pantalla la presencia de un archivo .CSV en la raiz, por lo tanto el programa no ofrece mensaje alguno en esta situacion.

Segundo paso – Utilización del archivo .CSV dentro del programa:

Tras haber constituido exitosamente el archivo .CSV a nivel local con toda la informacion necesaria para la constitucion del programa; se necesitaba una funcion aparte la cual cumpliese con dos funciones principales:

- En primer lugar detectar si existía un archivo .CSV a nivel local, y dada la negativa generarlo (a través de la funcion *api_build.py*.
- En segundo lugar, existiendo un archivo .CSV, lo convirtiese en una lista de elementos tipo diccionario, para un apropiado manejo de informacion por parte del programa.

Nuevamente se utilizan metodos de la librería .CSV para el manejo de archivos, permitiendo la lectura de este a través de la funcion csv.DictReader() y la composición de elementos de tipo diccionario, con las limitaciones fijadas por las premisas del trabajo:

- -String para nombre del pais.
- -INT para dato de población y superficie territorial.
- -String para dato - continente.

Leido el archivo mediante una funcion clasica de manejo de archivos, se utiliza un bucle for, para la lectura de "*reader"* y la posterior conversión de dicha informacion en variables locales que, reunidas las cuadro, compondrán un objeto de tipo Diccionario denominado "*pais"*. Dicho elemento, será agregado mediante el método "*.append()"* a la lista principal del menú; la cual será la que se utilizaran en todas las funciones.

Tercer paso: Implementación de funcionalidades, subfunciones, y logica detrás de su implementacion

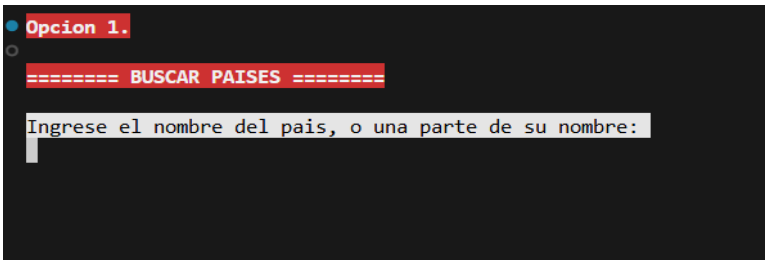
La rúbrica del trabajo practico exije implementadas las siguientes funcionalidades:

- La búsqueda exacta y parcial de paises por información ingresada por el usuario
- El filtrado y procesamiento de informacion de cada pais por distintos criterios (población, continente y superficie territorial)
- El filtrado y ordenamiento de información de cada pais tambien por distintos criterios (alfabético, cantidad de habitantes, y extensión territorial)
- El filtrado y ofrecer estadisticas particulares en el analisis de cada pais (paises con mayor o menor población, el promedio global de población por pais, el promedio de superficie por pais, y cuantos paises existen en cada continente)

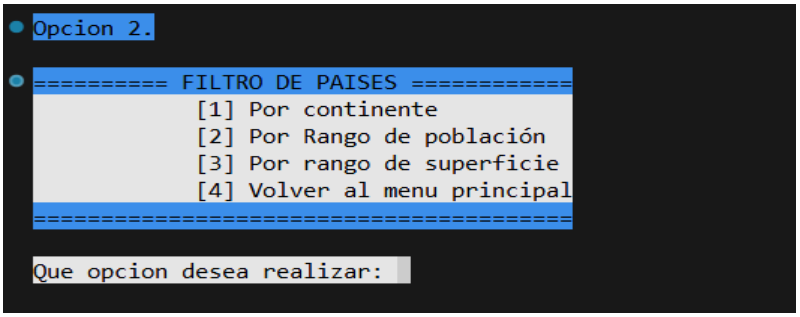
Para implementar las mismas, la mejor manera de realizarlo era a través de una implementación combinada de funcionalidades, que persistían en la consola, gracias a la implementación de bucles. Los bucles que permitían la persistencia de datos se manejan con un input del usuario, quien voluntariamente va a decidir si seguia utilizando otras funcionalidades, o salía completamente del programa.

Asi se crean 4 funcionalidades principales, las cuales, cumpliendo los puntos mencionados, se denominarían:

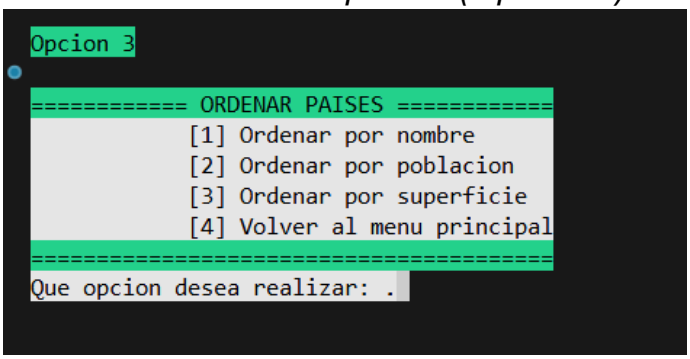
- -menu_1_buscar.py
- -menu_2_filtrar.py
- -menu_3_ordenar.py
- -menu_4_estadisticas.py



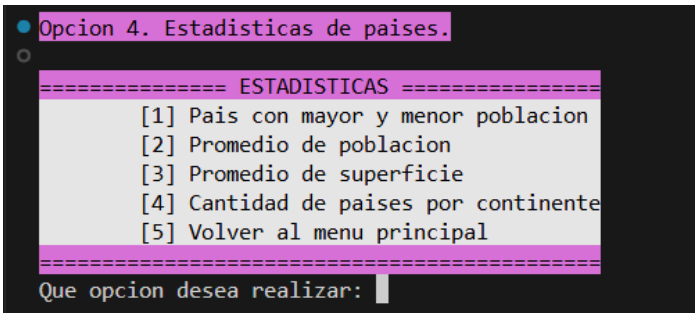
Funcion de buscar países (Opcion 1)



Funcion de filtrar paises (Opcion 2)

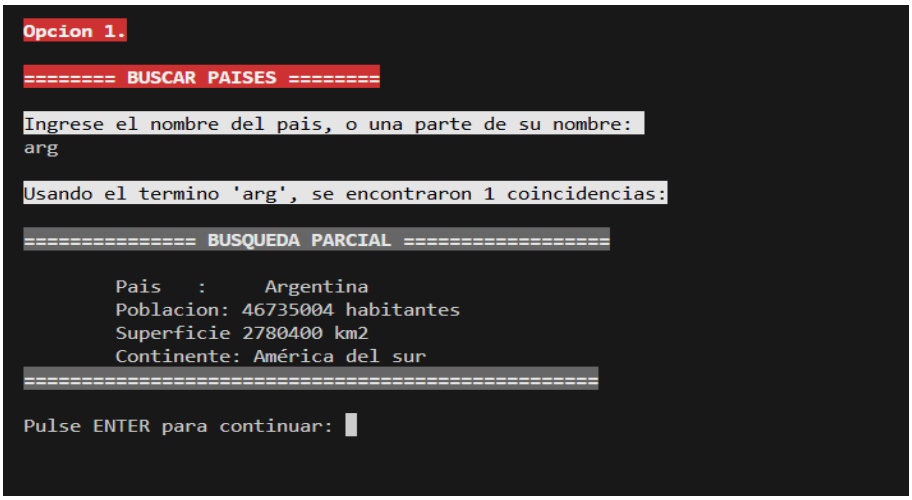


Funcion de ordenar paises (Opcion 3)

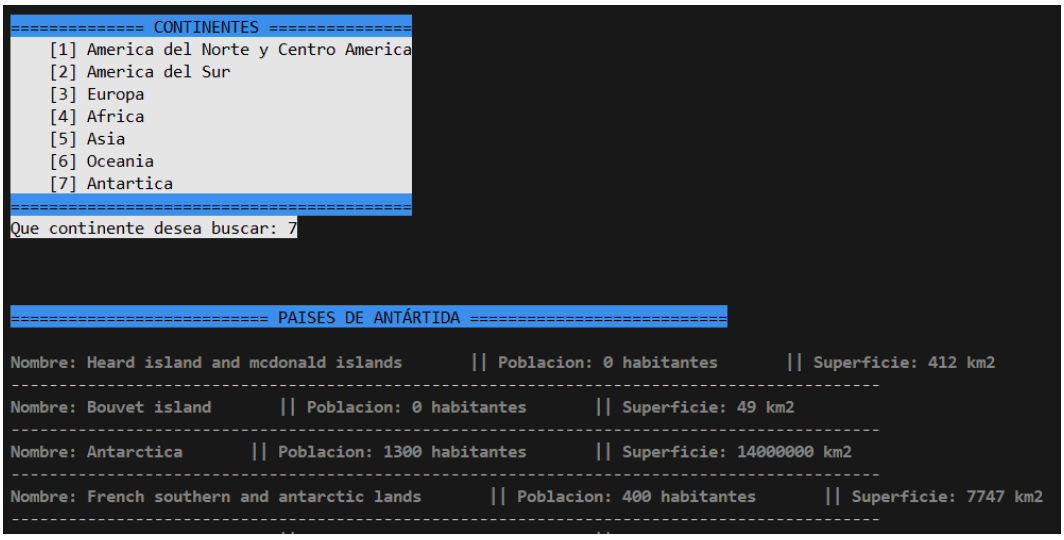


Funcion de obtención de estadísticas (Opción 4)

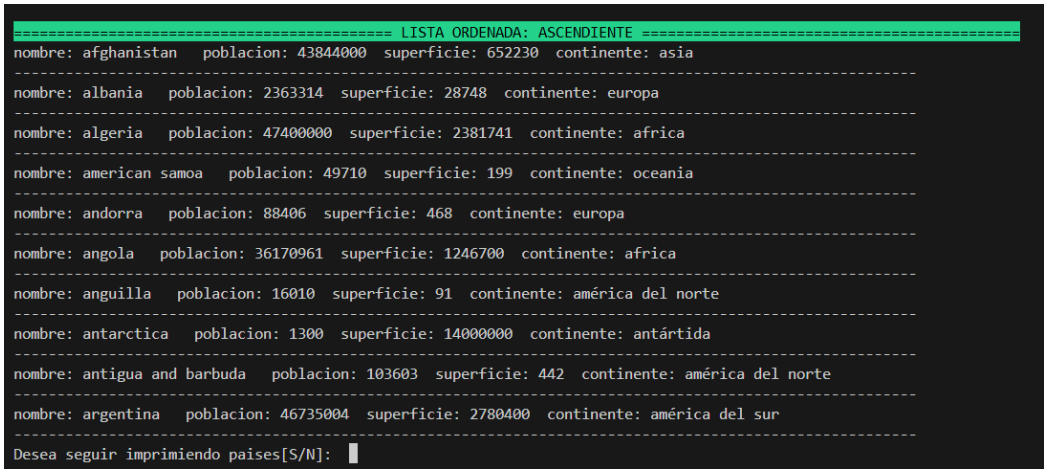
El usuario tendría por pantalla impresa la opcion de ingresar en cada funcion, por una opcion distinta, el cual a su vez tendría subfunciones encargadas de cumplir cada funcion por separado. Cada funcion, por una cuestion de claridad y simpleza, tendría creado un archivo, con una nomenclatura sencilla de identificar de a cuál menú corresponde cada script de funciones. A continuación algunas de las funcionalidades que están presentes en los distintos menús.



Busqueda de paises por exactitud parcial

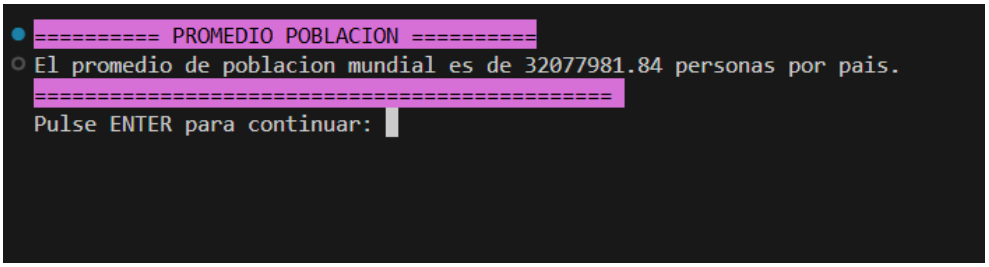


Filtración de paises existentes en Antártida



Ordenamiento de paises por criterio alfabético ascendente

(se imprime de a 10 paises, permitiendo al usuario observarlos a su ritmo)



Estadisticas y calculos, por ej promedio poblacional por pais

A su vez, existirían funciones generales que serian utilizadas indistintamente en los cuatro archivos, las cuales no tendrían numeración vinculante, dando a entender su uso general en todo el programa. Principalmente f_validaciones reunirían especificamente las que busquen validar el ingreso de informacion por parte del usuario, y f_varias reunirían las que no tengan propósito especifico.

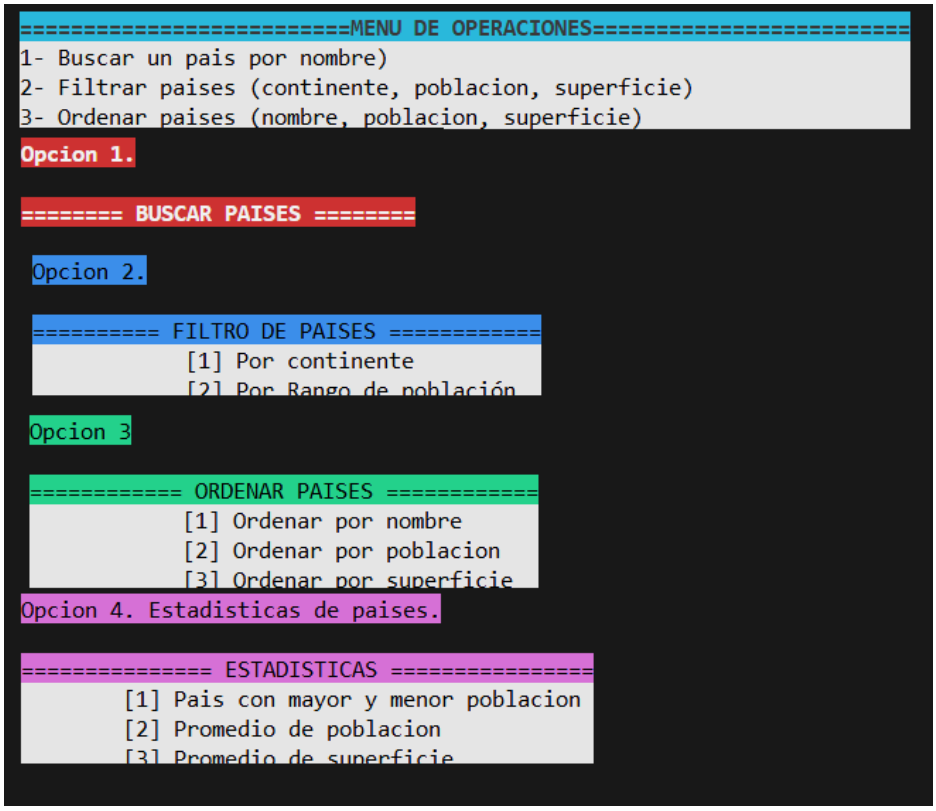
De esta manera, organizando la estructura de archivos, se podía facilitar la eventual necesidad de agregar o removar funciones, como asi testear o debugear las mismas.

Cuarto paso – Empleo de Colorama como embellecimiento y funcionalidad:

Al momento de integrar funcionalidades de accesibilidad a la consola; se consideró que la utilización de librerías como Colorama, podria

brindarle algun aporte, no solamente desde lo estético, sino tambien de la propia funcionalidad.

Es por ello, que se tomó como criterio principal, darle un color en particular a la utilización de las funcionalidades de cada una de las opciones. Por lo tanto, si nos encontrábamos buscando realizar funciones de ordenamiento de paises, siempre predominaría el color verde; mientras que por otro lado, si se buscaba observar estadísticas, el color magenta estaria presente. De esta manera, como se menciono no solo se buscaba embellecer, sino darle una funcionalidad útil, de cara a la presentación frente al usuario.



Presentación final

Quinto paso: Composición de imagen y contenedor en Docker

Finalmente, una vez finalizado el proyecto en su totalidad, se realizó la composición de una imagen de Docker.

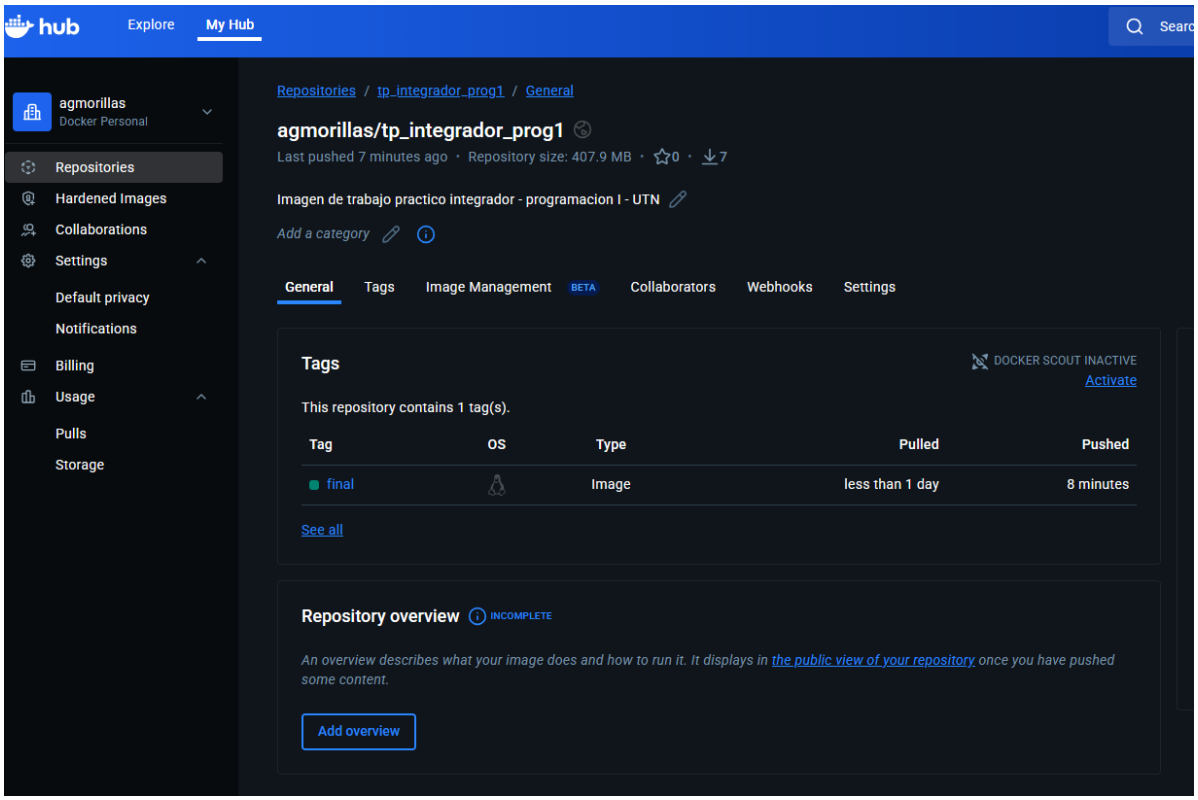
Presentación del contenedor donde existe el proyecto integrador final

Para poder construir el contenedor con los archivos necesarios y correspondientes, se construyó una imagen utilizando Python:3, y al momento de componer dicho archivo, se generó un "requirements.txt" agregando las dependencias necesarias a instalar. En este caso, unicamente dos las consideramos necesarias: Colorama y Operator.

Una vez reunidos los archivos, se generó una imagen ("integrador:1.2") ; y a través de dicha imagen se creó un conteiner ("integrador_contenedor").

Actualmente se encuentra hosteado en hub.docker, en el siguiente enlace:

https://hub.docker.com/repository/docker/agmorillas/tp_integrador_prog1/general



Vista de repositorio actualmente online