

Configuración de Contenedores:

Para la configuración de contenedores utilizaremos docker compose por facilidad de uso y de configuración, a continuación se muestra el archivo yml con el cual se levantarán los contenedores necesarios para la realización del proyecto , el cual tendra el nombre “proyectocompose.yml” , en el cual estaran contenidos los siguientes contenedores:

- scheduler : Scheduler de Airflow
- webserver: Servidor Airflow
- postgres: servidor Postgres utilizado por airflow
- db: Servidor mysql para almacenar los datos
- rstudios: Servidor de R para levantar el shiny app del dashboard

```
version: '3.3'
services:
  postgres:
    image: postgres:12
    environment:
      - POSTGRES_USER=airflow
      - POSTGRES_PASSWORD=airflow
      - POSTGRES_DB=airflow
    ports:
      - "5433:5432"
```

```
webserver:
  image: apache/airflow
  hostname: webserver
  restart: always
  depends_on:
    - postgres
  env_file:
    - .env
  volumes:
    - ./dags:/opt/airflow/dags
    - ./scripts:/opt/airflow/scripts
    - ./airflow-logs:/opt/airflow/logs
    - ./data:/data
  ports:
    - "8080:8080"
  entrypoint: ./scripts/airflow-entrypoint.sh
  healthcheck:
    test: ["CMD-SHELL", "[ -f /usr/local/airflow/airflow-webserver.pid ]"]
    interval: 30s
    timeout: 30s
    retries: 32
```

```
scheduler:
  image: apache/airflow
  restart: always
  depends_on:
    - postgres
    - webserver
  env_file:
    - .env
  ports:
    - "8793:8793"
  volumes:
    - ./dags:/opt/airflow/dags
    - ./airflow-logs:/opt/airflow/logs
  command: scheduler
  healthcheck:
    test: ["CMD-SHELL", "[ -f /usr/local/airflow/airflow-webserver.pid ]"]
    interval: 30s
    timeout: 30s
    retries: 3
```

```
db:
  image: mysql:5.7
  command: --default-authentication-plugin=mysql_native_password
  restart: always
  environment:
    MYSQL_DATABASE: 'db'
    # Para no usar Root
    MYSQL_USER: 'user'
    # Password para conectarse
    MYSQL_PASSWORD: 'PD2020GALILEO'
    # Password para conectarse como root
    MYSQL_ROOT_PASSWORD: 'PD2020GALILEO'
  ports:
    # <Puerto expuesto> : < puerto de MySQL adentro del container>
    - '3306:3306'
  expose:
    # Abrir puerto 3306 en el contenedor
    - '3306'
    # donde se va a guardar la info
  volumes:
    - my-db:/Users/luisgarcia/Maestria/ProductDevelopment/Parcial1/mysql
```

```

rstudios:
  image: dceoy/rstudio-server
  ports:
    - 8787:8787
  volumes:
    - rstudio-server-data:/Users/luisgarcia/Maestria/ProductDevelopment/Parcial1/mysql
  working_dir: /home/rstudio
  entrypoint:
    - /usr/lib/rstudio-server/bin/rserver
  command:
    - --server-daemonize=0
    - --server-app-armor-enabled=0
  depends_on:
    - db

volumes:
  my-db:
  rstudio-server-data:

```

Para iniciar los contenedores necesitaremos correr el siguiente comando:

`docker-compose -f proyectocompose.yml up`

El parametro -f hace referencia al archivo yml que deseamos correr y la palabra up es la acción que deseamos ejecutar la cual es levantar los contenedores y ponerlos a correr. Al finalizar nos mostrara los mensajes de la consola de los contenedores y no deberá mostrar ningún tipo de error.

```

27 - 8787:8787
28 volumes:
29 - rstudio-server-data:/Users/luisgarcia/Maestria/ProductDevelopment/Parcial1/mysql

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE
db_1 2020-11-18T22:09:52.568396Z 0 [Note] Plugin 'FEDERATED' is disabled.
db_1 2020-11-18T22:09:52.574083Z 0 [Note] InnoDB: Buffer pool(s) load completed at 201118 22:09:52
db_1 2020-11-18T22:09:52.582034Z 0 [Note] Found ca.pem, server-cert.pem and server-key.pem in data directory. Trying to enable SSL support using them.
db_1 2020-11-18T22:09:52.583047Z 0 [Note] Skipping generation of SSL certificates as certificate files are present in data directory.
db_1 2020-11-18T22:09:52.584938Z 0 [Warning] CA certificate ca.pem is self signed.
db_1 2020-11-18T22:09:52.585036Z 0 [Note] Skipping generation of RSA key pair as key files are present in data directory.
db_1 2020-11-18T22:09:52.589799Z 0 [Note] Server hostname (bind-address): '*'; port: 3306
db_1 2020-11-18T22:09:52.590156Z 0 [Note] IPv6 is available.
db_1 2020-11-18T22:09:52.590925Z 0 [Note] - '::' resolves to '::';
db_1 2020-11-18T22:09:52.590985Z 0 [Note] Server socket created on IP: '::'.
db_1 2020-11-18T22:09:52.596750Z 0 [Warning] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is accessible to all OS users. Consider c
hoosing a different directory.
db_1 2020-11-18T22:09:52.621907Z 0 [Note] Event Scheduler: Loaded 0 events
db_1 2020-11-18T22:09:52.622219Z 0 [Note] mysqld: ready for connections.
db_1 Version: '5.7.32' socket: '/var/run/mysqld/mysqld.sock' port: 3306 MySQL Community Server (GPL)
^[[1;2c^[1;2c^[1;2c

```

Dentro de la configuración de los contenedores se encuentran las siguientes líneas:

```

ports:
  # <Puerto expuesto> : < puerto de MySQL adentro del container>
  - '3306:3306'
expose:
  # Abrir puerto 3306 en el contenedor
  - '3306'

```

```

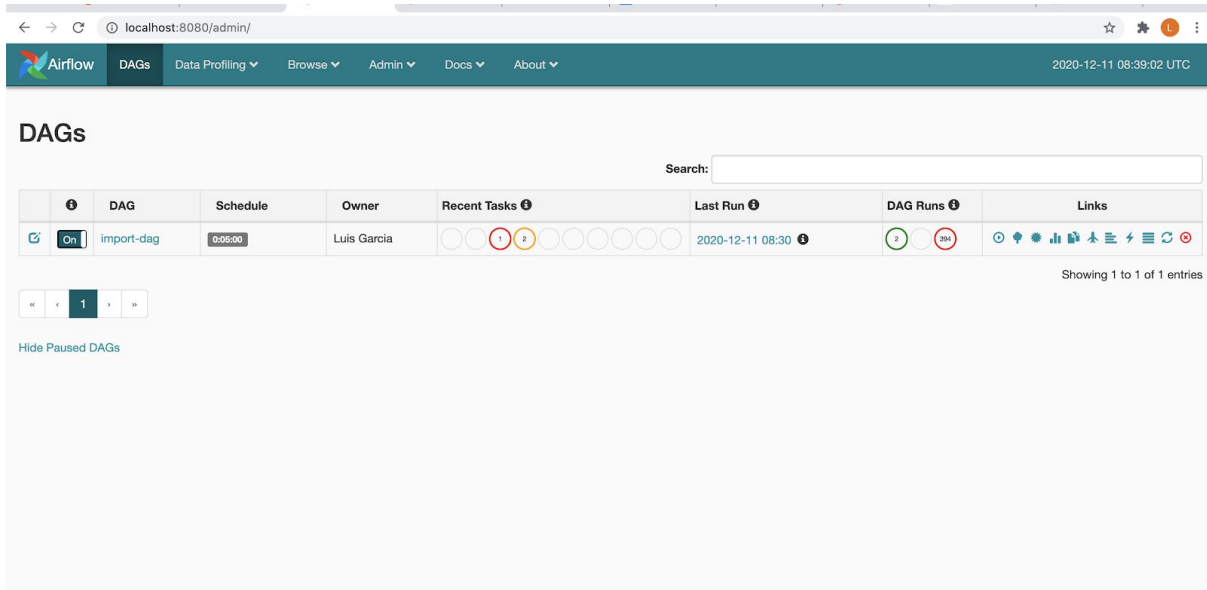
ports:
  - 8787:8787

```

Las cuales son para exponer los puertos del contenedor hacia nuestra máquina local

Airflow y DAGS

Una vez el contenedor de Airflow este corriendo podremos entrar al servidor por medio del puerto definido



The screenshot shows the Apache Airflow web interface in a browser at localhost:8080/admin/. The top navigation bar includes links for DAGs, Data Profiling, Browse, Admin, Docs, and About. The main heading is "DAGs". Below it is a search bar and a table listing DAGs. The table has columns for DAG, Schedule, Owner, Recent Tasks, Last Run, DAG Runs, and Links. One DAG is listed: "import-dag" with a schedule of "00:05:00", owner "Luis Garcia", and a status of "On". The "Recent Tasks" column shows a series of circles, with one red circle containing a minus sign and one yellow circle containing a plus sign. The "Last Run" column shows "2020-12-11 08:30". The "DAG Runs" column shows a green circle with a plus sign and a red circle with a minus sign. The "Links" column contains various icons for monitoring and managing the DAG. At the bottom of the table, it says "Showing 1 to 1 of 1 entries".

En el cual podremos ver los dags que se definieron en el mismo , en el caso actual se definió únicamente un DAG el cual estará encargado de importar los 3 archivos csv hacia nuestro servidor de base de datos, el cual está definido de la siguiente manera

```
import codecs
import logging
from datetime import timedelta
from airflow import DAG
from airflow.operators.python_operator import PythonOperator
from airflow.utils import dates
from airflow.models import Variable
from sqlalchemy import create_engine

logging.basicConfig(format="%(name)s-%(levelname)s-%(asctime)s-%(message)s", level=logging.INFO)
logger = logging.getLogger(__name__)
logger.setLevel(logging.INFO)

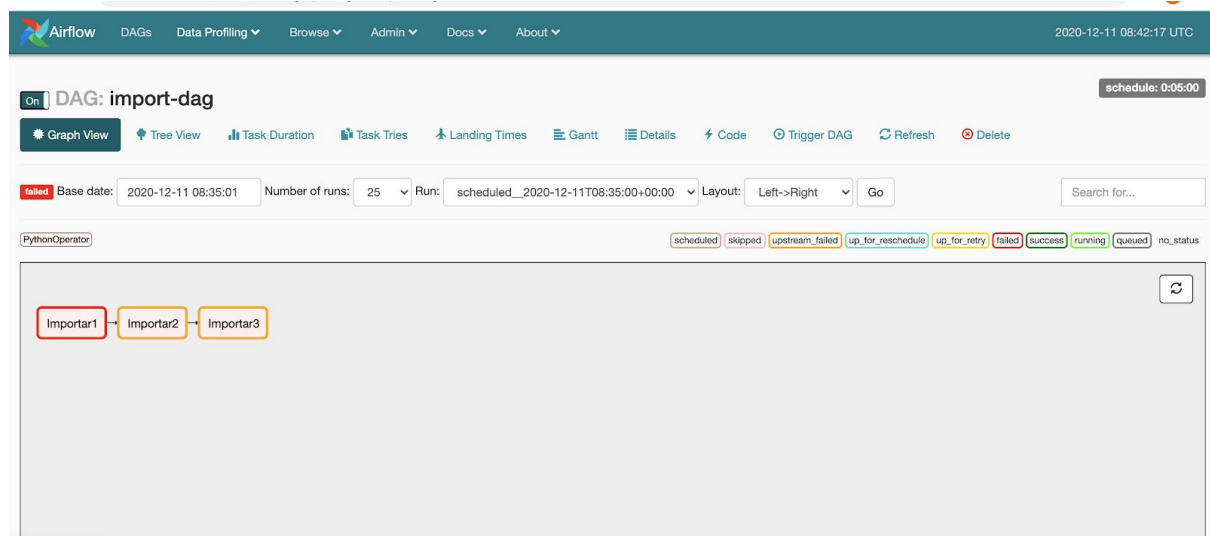
def create_dag(dag_id):
    default_args = {
        "owner": "Luis Garcia",
        "description": (
            "DAG para importar datos proporcionados a servidor de base de datos"
        ),
        "depends_on_past": False,
        "start_date": dates.days_ago(1),
        "retries": 1,
        "retry_delay": timedelta(minutes=1),
        "provide_context": True,
    }
    new_dag = DAG(
        dag_id,
        default_args=default_args,
        schedule_interval=timedelta(minutes=5),
    )
```

```

30 def importarCsv(**kwargs):
31     logger.info('=====Ejecutando Tarea de Importacion =====')
32     logger.info('===== Archivo =====')
33     logger.info(kwargs['archivo'])
34     logger.info('===== Tabla =====')
35     logger.info(kwargs['tabla'])
36     host = Variable.get("127.0.0.1")
37     db_name = Variable.get("db")
38     username = Variable.get("user")
39     password = Variable.get("P02020GALILEO")
40     connection = create_engine('mysql://(username):(password)@(url)/(db_name)?charset=utf8'
41                               .format(username=username, password=password,
42                                       url=host, db_name=db_name), echo=False)
43     conn = connection.connect()
44     rs = conn.execute('SELECT * FROM time_series_covid19_recovered_global')
45     i = 0
46     for row in rs:
47         i = i + 1
48         data = ( { "id": 1},
49                 { "id": 2},
50             )
51
52     sql = "INSERT INTO " + logger.info(kwargs['tabla']) + " (id,pais, fecha, count) VALUES (%s,%s,%s, %s)"
53     df = pd.read_csv ('./data/'+kwargs['archivo'])
54     for index, row in df.iterrows():
55         for col in df.columns:
56             if col != 'Province/State' and col != 'Country/Region' and col != 'Lat' and col != 'Long':
57                 date_object = datetime.strptime(col, '%m/%d/%y')
58                 val = (0,row['Country/Region'],date_object ,row[col])
59                 conn.execute(sql, val)
60     conn.close()
61
62 with new_dag:
63     task1 = PythonOperator(task_id='Importar1',
64                             python_callable=importarCsv,
65                             op_kwargs=
66                             {
67                                 'archivo': 'time_series_covid19_confirmed_global.csv',
68                                 'tabla': 'time_series_covid19_confirmed_global'
69                             },
70                             provide_context=True)
71
72     task2 = PythonOperator(task_id='Importar2',
73                             python_callable=importarCsv,
74                             op_kwargs=
75                             {
76                                 'archivo': 'time_series_covid19_deaths_global.csv',
77                                 'tabla': 'time_series_covid19_deaths_global'
78                             },
79                             provide_context=True)
80
81     task3 = PythonOperator(task_id='Importar3',
82                             python_callable=importarCsv,
83                             op_kwargs=
84                             {
85                                 'archivo': 'time_series_covid19_recovered_global.csv',
86                                 'tabla': 'time_series_covid19_recovered_global'
87                             },
88                             provide_context=True)
89
90     task2.set_upstream(task1)
91     task3.set_upstream(task2)
92     return new_dag
93 dag_id = "import-dag"
94 globals()[dag_id] = create_dag(dag_id)

```

Asi mismo podemos ver dentro de la interfaz web la forma de nuestro dag



Este dag será encargado de leer los datos de los archivos csv e ingresarlos a nuestro servidor de base de datos.

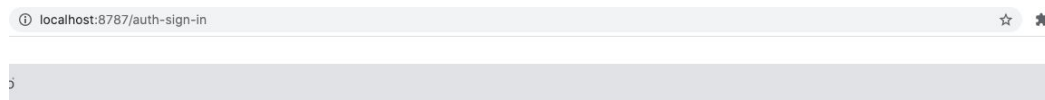
casos	286	Afghanistan	2020-11-01	34326
geoloc	287	Afghanistan	2020-11-02	34342
muertes	288	Afghanistan	2020-11-03	34355
recuperados	289	Afghanistan	2020-11-04	34362
	290	Afghanistan	2020-11-05	34440
	291	Afghanistan	2020-11-06	34440
	292	Afghanistan	2020-11-07	34446
	293	Afghanistan	2020-11-08	34458
	294	Afghanistan	2020-11-09	34721
	295	Afghanistan	2020-11-10	34954
	296	Afghanistan	2020-11-11	34967
	297	Afghanistan	2020-11-12	35024
	298	Afghanistan	2020-11-13	35036
	299	Afghanistan	2020-11-14	35067
	300	Afghanistan	2020-11-15	35092
	301	Afghanistan	2020-11-16	35137
	302	Afghanistan	2020-11-17	35160
	303	Afghanistan	2020-11-18	35295
	304	Afghanistan	2020-11-19	35350
	305	Afghanistan	2020-11-20	35370
	306	Afghanistan	2020-11-21	35422
	307	Afghanistan	2020-11-22	35934
	308	Afghanistan	2020-11-23	35976
	309	Albania	2020-01-22	0
	310	Albania	2020-01-23	0
	311	Albania	2020-01-24	0
	312	Albania	2020-01-25	0
	313	Albania	2020-01-26	0
	314	Albania	2020-01-27	0

TABLE INFORMATION

- created: 12/10/20
- engine: InnoDB
- rows: 30,500

R Studio Server

Una vez el servidor de estudio esté levantado podemos conectarnos a RStudio server por medio del puerto 8787, al ingresar a la página nos pedirá que ingresemos usuario y contraseña la cual dependerá de la imagen del contenedor que utilizamos.



Sign in to RStudio

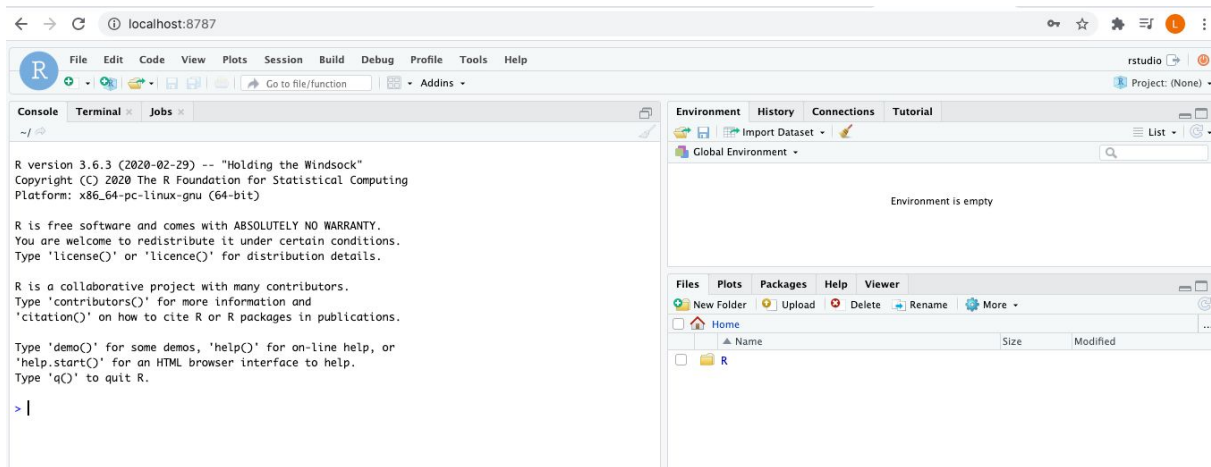
Username:

Password:

You will automatically be signed out after 60 minutes of inactivity.

☐ Stay signed in when browser closes

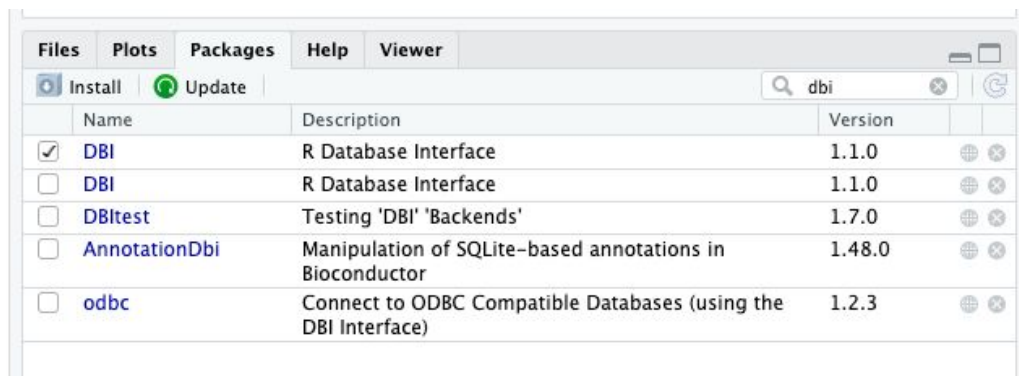
Sign In



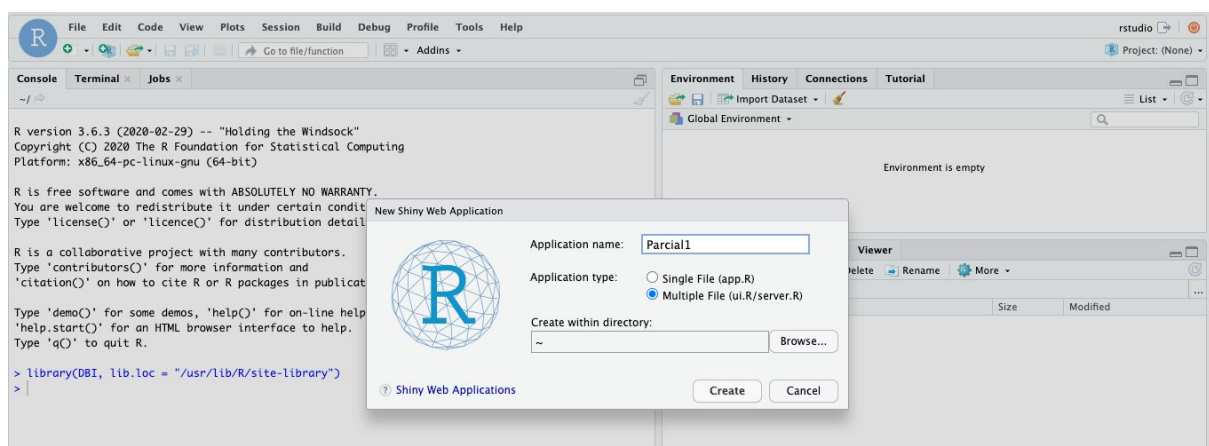
Una vez autenticados tendremos nuestro ambiente de R Studio en el cual podremos iniciar el desarrollo de nuestro dashboard.

Conexión a MYSQL en RStudio

Para conectarnos a MYSQL usaremos el paquete DBI, Mariadb, ODBC para conectarnos a nuestro servidor



Luego de esto podemos hacer click en File -> New ShinyWebApp para crear nuestro dashboard



Luego de esto podemos crear nuestra conexión y cargar la información.

```

1
2 library(shiny)
3 library(RMariaDB)
4 library(dplyr)
5 con_sql <- dbConnect(RMariaDB::MariaDB(), user='user', password='PD2020GALILEO', dbname='db', host='
6 casos <- dbReadTable(conn = con_sql, name = 'casos')
7 muertes <- dbReadTable(conn = con_sql, name = 'muertes')
8 recuperados <- dbReadTable(conn = con_sql, name = 'recuperados')
9 geoloc <- dbReadTable(conn = con_sql, name = 'geoloc')
10 paises <- unique(geoloc['pais'])
11 geo <- geoloc[!duplicated(geoloc[,c('pais')]),]

```

Global Enviro

Data

mod

primary

Functions

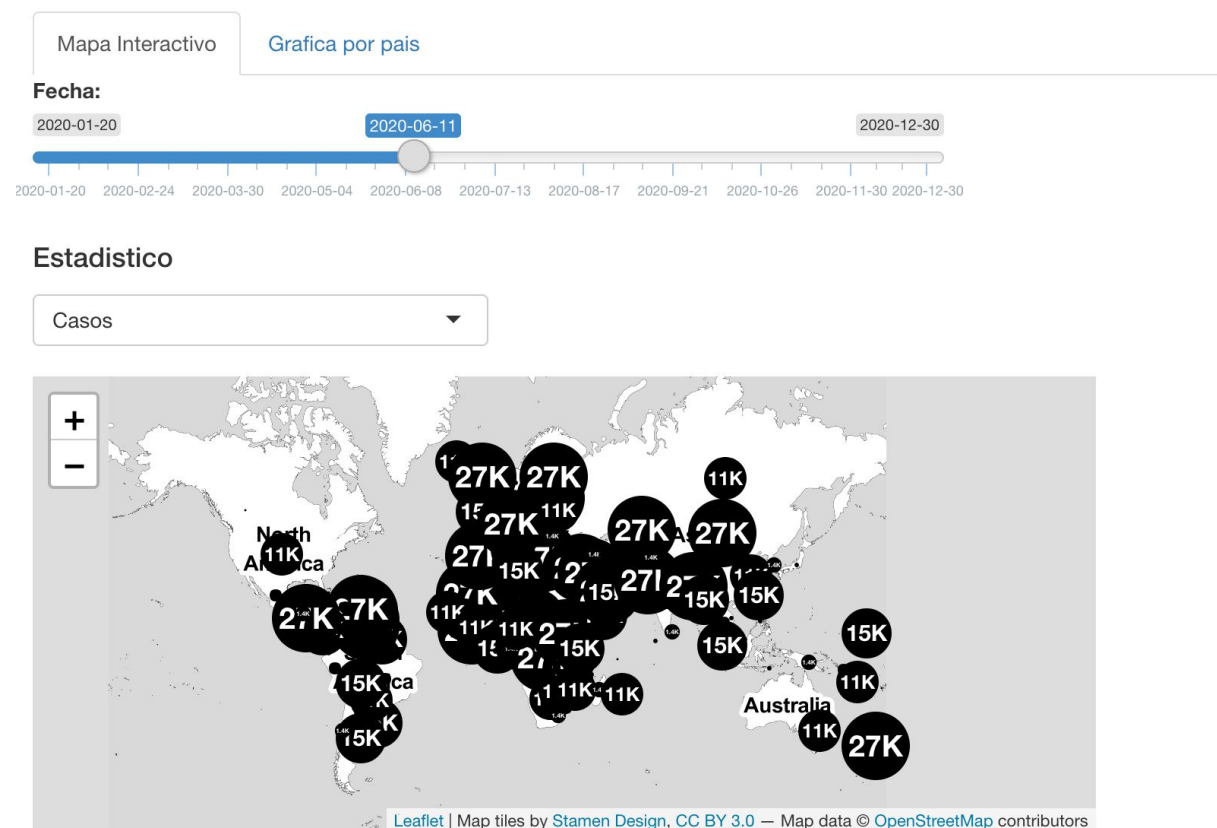
NEEstimator

Dashboard

Crearemos dos tabs en nuestro dashboard , en las cuales podremos ver el un mapa interactivo de los datos y el otro una grafica especifica de el país escogido.

Tab 1

COVID19 DASHBOARD - PROYECTO



En este podremos elegir la fecha y la fecha para la cual queremos ver la informacion en el mapa

COVID19 DASHBOARD - PROYECTO

Mapa Interactivo

Grafica por pais

Fecha:

2020-01-20

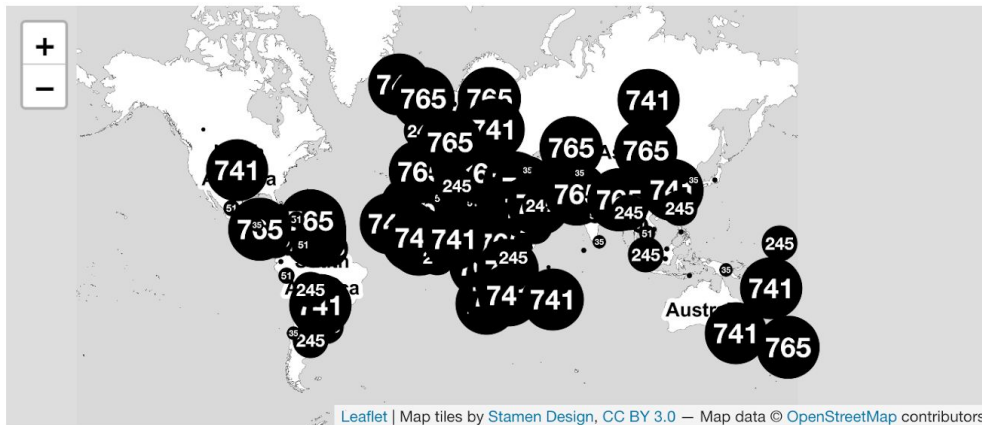
2020-06-11

2020-12-30

2020-01-20 2020-02-24 2020-03-30 2020-05-04 2020-06-08 2020-07-13 2020-08-17 2020-09-21 2020-10-26 2020-11-30 2020-12-30

Estadistico

Muertes



COVID19 DASHBOARD - PROYECTO

Mapa Interactivo

Grafica por pais

Fecha:

2020-01-20

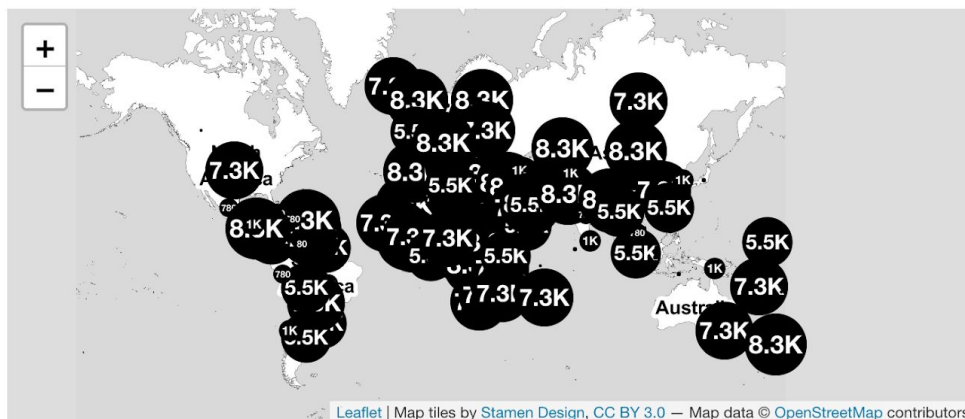
2020-06-11

2020-12-30

2020-01-20 2020-02-24 2020-03-30 2020-05-04 2020-06-08 2020-07-13 2020-08-17 2020-09-21 2020-10-26 2020-11-30 2020-12-30

Estadistico

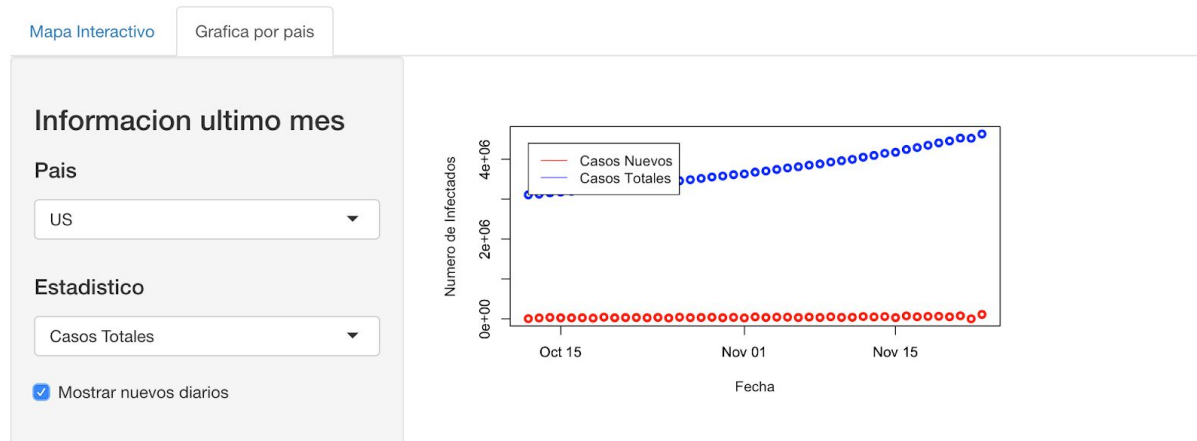
Recuperados



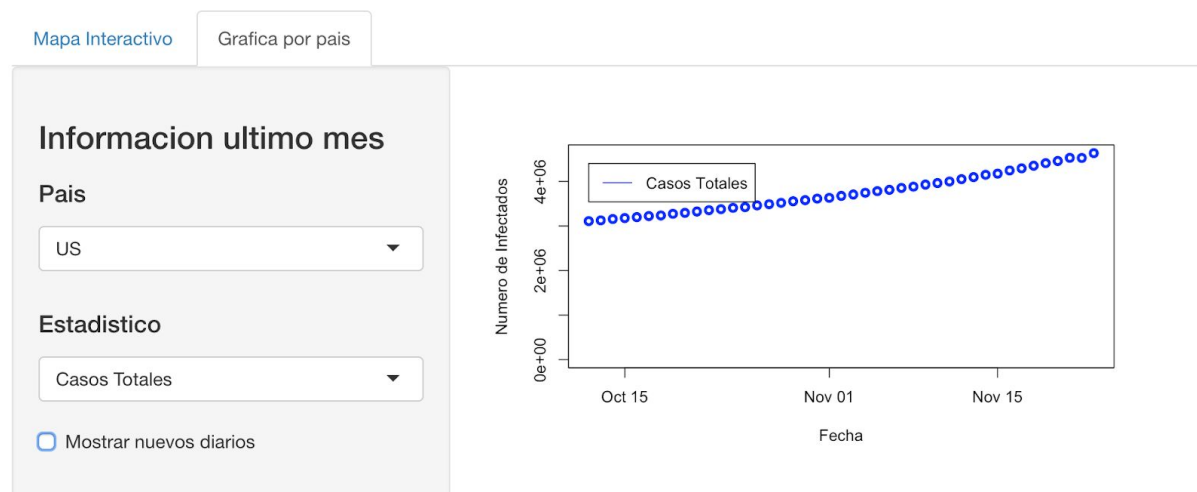
Tab 2

En este podremos ver información de un país y estadístico en específico

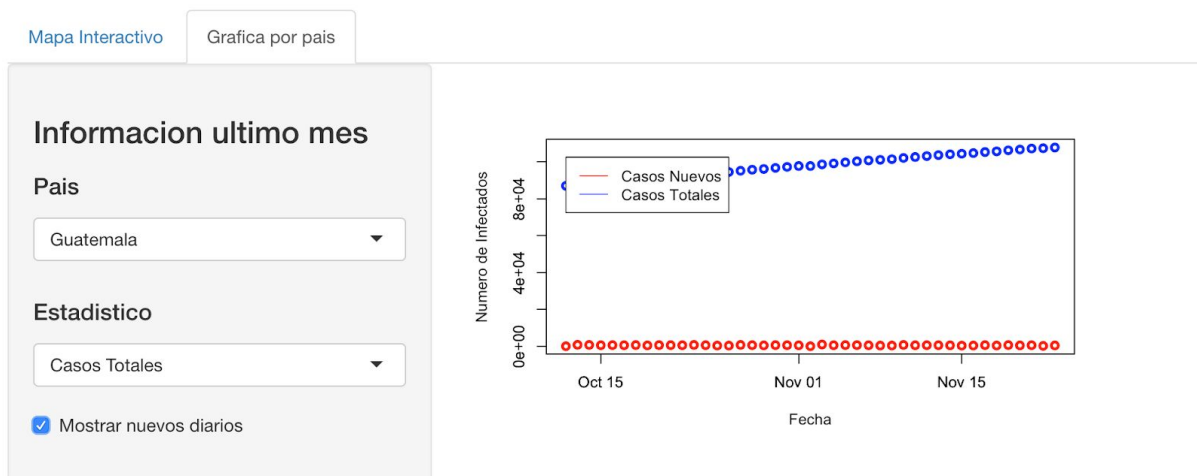
COVID19 DASHBOARD - PROYECTO



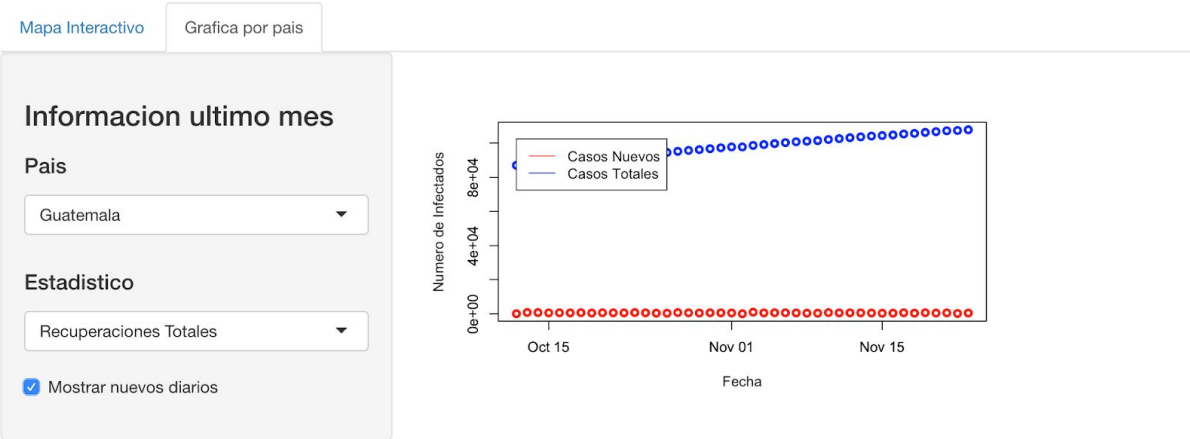
COVID19 DASHBOARD - PROYECTO



COVID19 DASHBOARD - PROYECTO



COVID19 DASHBOARD - PROYECTO



COVID19 DASHBOARD - PROYECTO

