Diseño de Bases de Datos y Aplicaciones

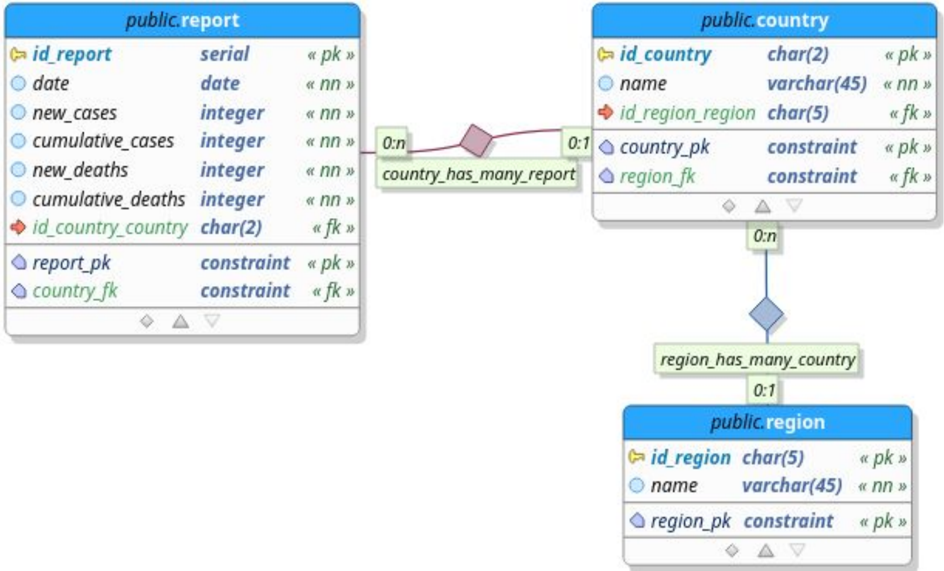
Diseño y Desarrollo de Aplicaciones

Diseño y Desarrollo de Aplicaciones

- Python https://www.python.org/
- Postgres https://www.postgresql.org/
- Psycopg2 https://pypi.org/project/psycopg2/
- Dash https://dash.plotly.com/

Diseño y Desarrollo de Aplicaciones

- Caso de Estudio Covid 19



public.report		
id_report	serial	« pk »
date	date	« nn »
new_cases	integer	« nn »
cumulative_cases	integer	« nn »
new_deaths	integer	« nn »
cumulative_deaths	integer	« nn »
id_country_country	char(2)	« fk »
report_pk	constraint	« pk »
country_fk	constraint	« fk »

public.country		
id_country	char(2)	« pk »
name	varchar(45)	« nn »
id_region_region	char(5)	« fk »
country_pk	constraint	« pk »
region_fk	constraint	« fk »

public.region		
id_region	char(5)	« pk »
name	varchar(45)	« nn »
region_pk	constraint	« pk »

Diseño y Desarrollo de Aplicaciones

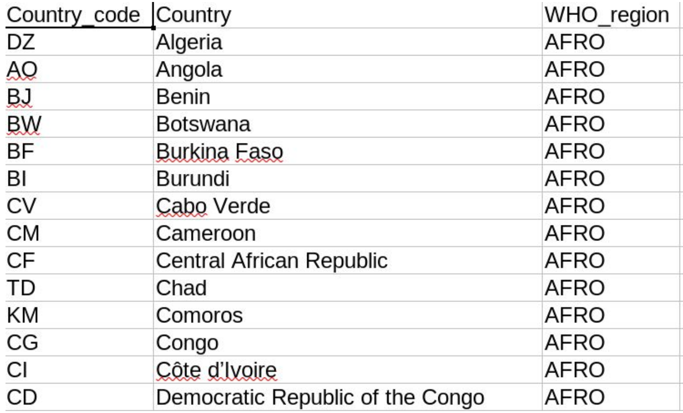
- Datos de regiones

```
INSERT INTO region VALUES ('AFRO ', 'Africa');
INSERT INTO region VALUES ('AMRO ', 'Americas');
INSERT INTO region VALUES ('EMRO ', 'Eastern Mediterranean');
INSERT INTO region VALUES ('EURO ', 'Europe');
INSERT INTO region VALUES ('SEARO', 'South-East Asia');
INSERT INTO region VALUES ('WPRO ', 'Western Pacific');
```

Diseño y Desarrollo de Aplicaciones

- Datos de paises
  - Archivo covid19countries.csv

```
copy country(id_country, name, id_region_region)
from '<<ruta>>/covid19countries.csv'
delimiter ','
csv header;
```



Country_code	Country	WHO_region
DZ	Algeria	AFRO
AO	Angola	AFRO
BJ	Benin	AFRO
BW	Botswana	AFRO
BF	Burkina Faso	AFRO
BI	Burundi	AFRO
CV	Cabo Verde	AFRO
CM	Cameroon	AFRO
CF	Central African Republic	AFRO
TD	Chad	AFRO
KM	Comoros	AFRO
CG	Congo	AFRO
CI	Côte d'Ivoire	AFRO
CD	Democratic Republic of the Congo	AFRO

Diseño y Desarrollo de Aplicaciones

- Datos de reportes
  - Archivo covid19.csv

```
copy report(date, id_country_country, new_cases, cumulative_cases, new_deaths,
cumulative_deaths)
from '<<ruta>>/covid19.csv'
delimiter ','
csv header;
```

Diseño y Desarrollo de Aplicaciones

- Conexión a la BD

```
import psycopg2
class Connection:
    def __init__(self):
        self.connection = None
    def openConnection(self):
        try:
            self.connection = 
psycopg2.connect(host="localhost",port="5432",dbname="covid19",user="postgres"
,password="123456")
        except Exception as e:
            print (e)

    def closeConnection(self):
        self.connection.close()
```

Diseño y Desarrollo de Aplicaciones

- Consultas SQL

```
def totalCasesByCountry():
    return """select cou.id_country as country_code, cou.name as country,
max(rep.cumulative_cases) as amount
            from report rep join country cou on (rep.id_country_country =
cou.id_country)
            group by cou.id_country, cou.name
            order by amount desc"""

def totalDeathsByCountry():
    return """select cou.id_country as country_code, cou.name as country,
max(rep.cumulative_deaths) as amount
            from report rep join country cou on (rep.id_country_country =
cou.id_country)
            group by cou.id_country, cou.name
            order by amount desc"""
```

Diseño y Desarrollo de Aplicaciones

- Dash

```
import dash
import dash_core_components as dcc
import dash_html_components as html
import plotly.express as px
import pandas as pd
from Connection import Connection
import covidSQL as sql

external_stylesheets =
["https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css
"]

app = dash.Dash(__name__, external_stylesheets=external_stylesheets)
```

Diseño y Desarrollo de Aplicaciones

- Dash

```
#Cases by country
con = Connection()
con.openConnection()
query = pd.read_sql_query(sql.totalCasesByCountry(), con.connection)
con.closeConnection()
dfCases = pd.DataFrame(query, columns=["country_code", "country", "amount"])
figBarCases = px.bar(dfCases.head(20), x="country", y="amount")
figMapCases = px.choropleth(dfCases, locations="country",
                            locationmode="country names",
                            color="amount",
                            hover_name="country",
                            color_continuous_scale=["#99ccff", "#ff3333"])
```

Diseño y Desarrollo de Aplicaciones

- Dash

```
#Deaths by country
con.openConnection()
query = pd.read_sql_query(sql.totalDeathsByCountry(), con.connection)
con.closeConnection()
dfDeaths = pd.DataFrame(query, columns=["country_code", "country", "amount"])
figBarDeaths = px.bar(dfDeaths.head(20), x="country", y="amount",
color_discrete_map={"name":"red"})
figMapDeaths = px.choropleth(dfDeaths, locations="country",
                            locationmode="country names",
                            color="amount",
                            hover_name="country",
                            color_continuous_scale=["#99ccff", "#ff3333"])
```

Diseño y Desarrollo de Aplicaciones

- Dash

```
#Layout
app.layout = html.Div(children=[
    html.H1(children='Covid 19 Dashboard'),
    html.H2(children='Cases by Country'),
    dcc.Graph(
        id='barCasesByCountry',
        figure=figBarCases
    ),
    dcc.Graph(
        id='mapCasesByCountry',
        figure=figMapCases
    ),
])
```