Universidad Politécnica Salesiana

Nombre: Fernando Sanchez

Materia: Simulación

Tema: Simulación de Eventos.

Prueba 1

Fecha: 11/05/2021

Objetivo:

- Consolidar los conocimientos adquiridos en clase para desarrollar simulaciones.

Enunciado:

- Diseñe y desarrolle un modelo y/o script que permita simular el siguiente caso real:
  - Se tiene los datos del ecuador (https://github.com/andrab/ecuacovid/tree/master/datos_crudos).

  En base a ello obtener los siguientes modelos:
    - Generar graficas para entender y procesar los datos:
      - Generar graficas y reportes del total de personas vacunadas.

In [1]:
```
#importar las librerias necesarias
import matplotlib.pyplot as plt
import matplotlib.lines as mlines
import numpy as np
import pandas as pd
```

In [3]:

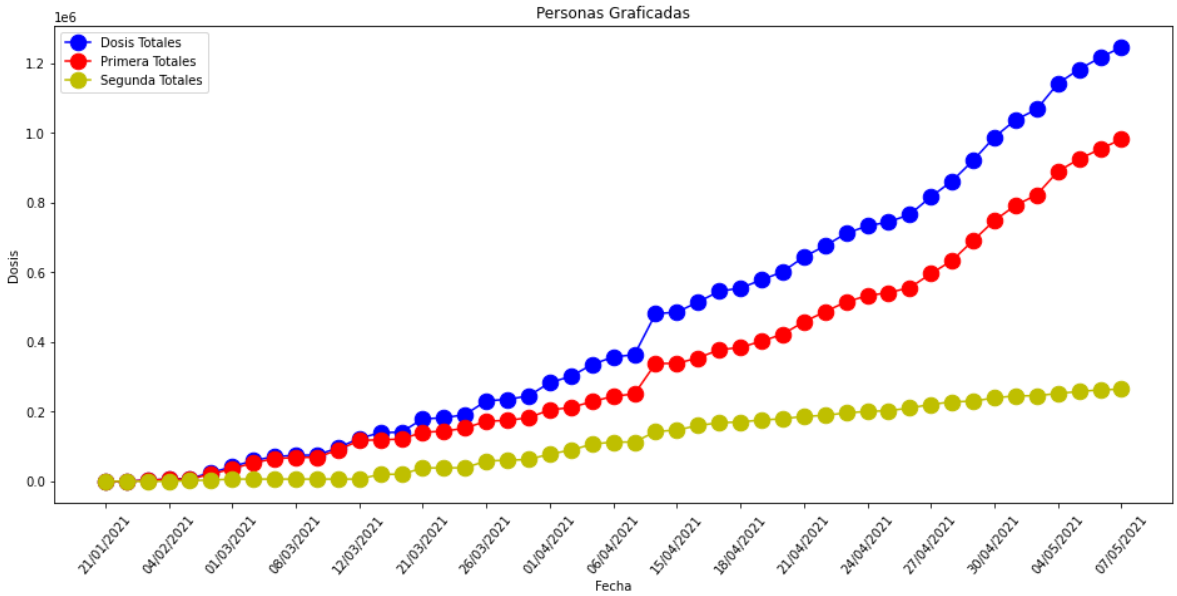
```
vacunas = pd.read_csv('ecuacovid-master/datos_crudos/vacunas/vacu
#imprimir los primeros 5 datos del archivo
vacunas.head(5)
```

Out[3]:

	fecha	dosis_total	primera_dosis	segunda_dosis
0	21/01/2021	0	0	0
1	22/01/2021	108	108	0
2	27/01/2021	2982	2982	0
3	04/02/2021	6228	6228	0
4	17/02/2021	8190	6228	1962

In [4]:

```
plt.figure(figsize=(16,7))

plt.title('Vacunas')

plt.plot(vacunas.fecha, vacunas.dosis_total, 'b.-', markersize=25
plt.plot(vacunas.fecha, vacunas.primera_dosis, 'r.-', markersize=
plt.plot(vacunas.fecha, vacunas.segunda_dosis, 'y.-', markersize=

plt.xticks(vacunas.fecha[::3].tolist())

plt.xlabel('Fecha')
plt.xticks(rotation=50)
plt.ylabel('Dosis')
plt.legend()
plt.show()
```

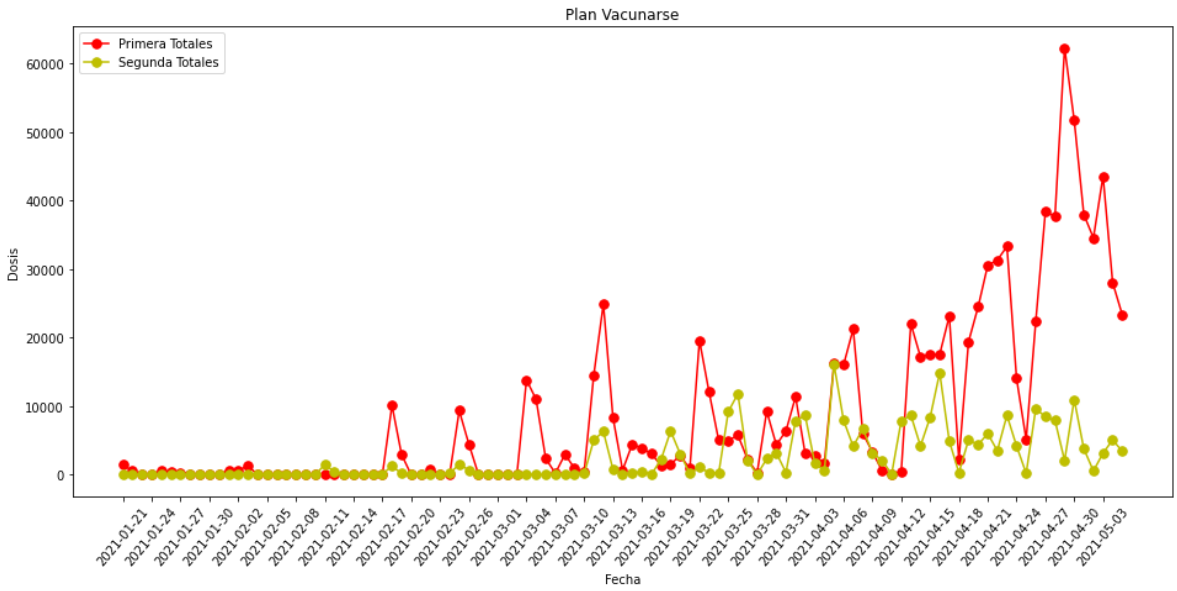


In [33]:
```
vacunas_planvacunarse = pd.read_csv('ecuacovid-master/datos_crudo
#imprimir los primeros 5 datos del archivo
vacunas_planvacunarse.head(5)
```

Out[33]:

	fecha	primera_dosis	segunda_dosis
0	2021-01-21	1500	0
1	2021-01-22	538	1
2	2021-01-23	31	0
3	2021-01-24	0	0
4	2021-01-25	622	0

In [38]:
```
plt.figure(figsize=(16,7))

plt.title('Plan Vacunarse')


plt.plot(vacunas_planvacunarse.fecha, vacunas_planvacunarse.prime
plt.plot(vacunas_planvacunarse.fecha, vacunas_planvacunarse.segun

plt.xticks(vacunas_planvacunarse.fecha[::3].tolist())

plt.xlabel('Fecha')
plt.xticks(rotation=50)
plt.ylabel('Dosis')
plt.legend()
plt.show()
```



- Generar grafico de pie por fabricante de la vacuna.

In [5]:

```
#Trabajar con datos en formato .csv
fabricantes = pd.read_csv('ecuacovid-master/datos_crudos/vacunas/
#imprimir los primeros 5 datos del archivo
fabricantes.head(18)
```

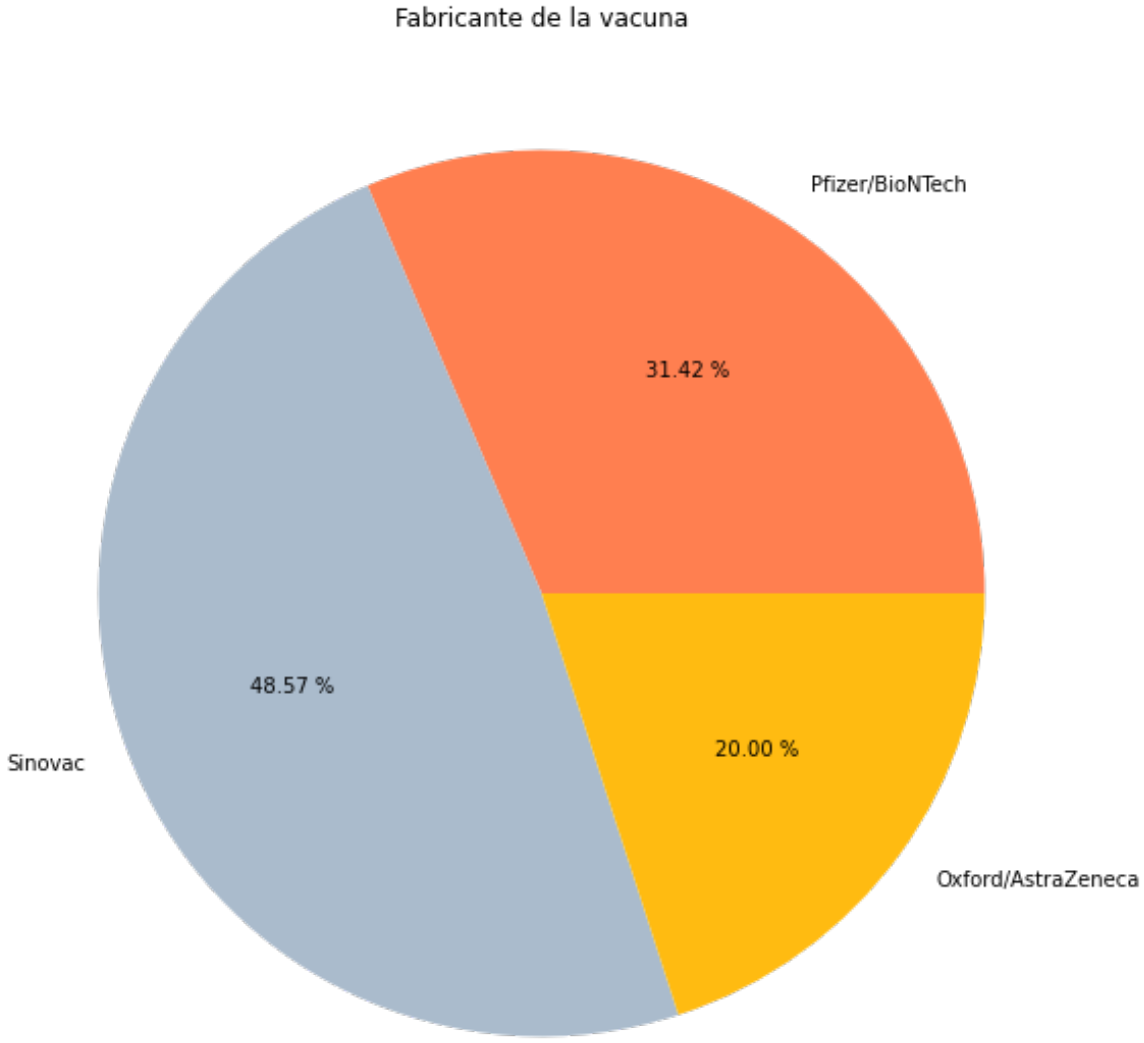
Out[5]:

	vaccine	total	arrived_at
0	Pfizer/BioNTech	8190	20/01/2021
1	Pfizer/BioNTech	16380	17/02/2021
2	Pfizer/BioNTech	17550	24/02/2021
3	Pfizer/BioNTech	31590	03/03/2021
4	Sinovac	20000	06/03/2021
5	Pfizer/BioNTech	73710	10/03/2021
6	Oxford/AstraZeneca	84000	17/03/2021
7	Pfizer/BioNTech	62010	17/03/2021
8	Pfizer/BioNTech	65520	24/03/2021
9	Pfizer/BioNTech	66690	31/03/2021
10	Pfizer/BioNTech	53820	05/04/2021
11	Sinovac	300000	07/04/2021
12	Sinovac	700000	10/04/2021
13	Pfizer/BioNTech	53820	14/04/2021
14	Pfizer/BioNTech	54990	21/04/2021
15	Oxford/AstraZeneca	336000	24/04/2021
16	Pfizer/BioNTech	54990	28/04/2021
17	Pfizer/BioNTech	100620	04/05/2021

In [6]:

```
# Generar un grafico de cual es su pie diestro
aux = 0
aux1 = 0
aux2 = 0
vaccine1 = fabricantes.loc[fabricantes.vaccine == 'Pfizer/BioNTec
for i in vaccine1:
    aux = aux+i

vaccine2 = fabricantes.loc[fabricantes.vaccine == 'Sinovac']['tot
for i in vaccine2:
    aux1 = aux1+i

vaccine3 = fabricantes.loc[fabricantes.vaccine == 'Oxford/AstraZe
for i in vaccine3:
    aux2 = aux2+i

plt.figure(figsize=(10,10))

etiquetas = ['Pfizer/BioNTech', 'Sinovac','Oxford/AstraZeneca']
colores = ['#ff7f50', '#aabbcc','#ffbb11']
```

```
plt.pie([aux,aux1,aux2], labels=etiquetas, colors=colores, autopc
plt.title('Fabricante de la vacuna')
plt.show()
```



In [7]:
```
sumaTotalVacunas = aux + aux1 +aux2
print(sumaTotalVacunas)
```

```
2099880
```

- Generar histogramas de vacunas por mes de l lega y fabricante.

In [8]:

```
fabricantes['arrived_at'] = pd.to_datetime(fabricantes['arrived_a

font = {'family': 'serif',
        'color':  'darkred',
        'weight': 'normal',
        'size': 16,
        }
j = 1000
```

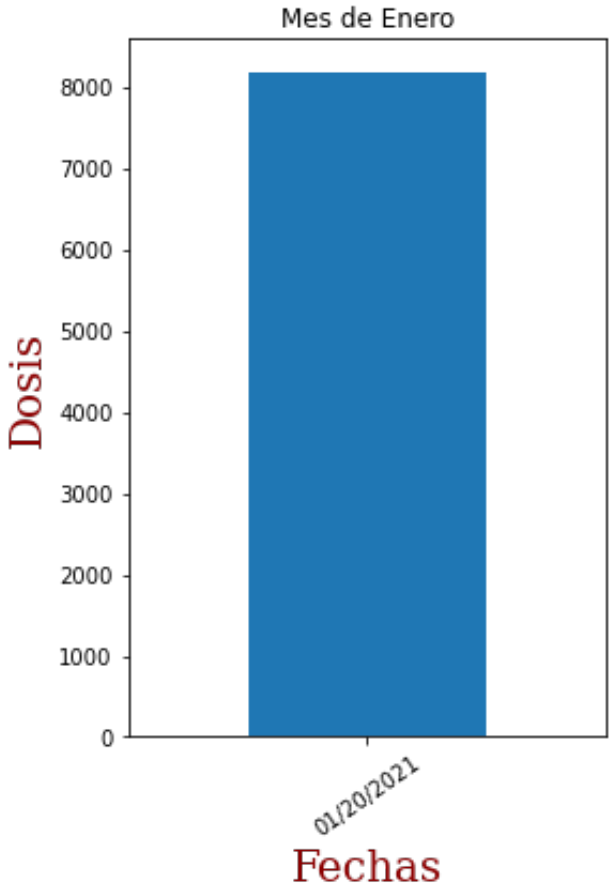
In [29]:

```
listae =[]
listae1 =[]
listae2 =[]

enero = fabricantes.loc[fabricantes.arrived_at.dt.month == 1]

for i in enero['arrived_at']:
    date_time = i
    d = date_time.strftime("%m/%d/%Y")
    listae.append(d)
print(listae)

for i in enero['total']:
    listae1.append(int(i))
print(listae1)

for i in enero['vaccine']:
    listae2.append((i))
print(listae2)
```

```
['01/20/2021']
[8190]
['Pfizer/BioNTech']
```

In [30]:

```
# Declaring the figure or the plot (y, x) or (width, height)
plt.figure(figsize = (4,6))

# Annotating the bar plot with the values (Fabricantes)
for i in range(len(listae)):
    plt.annotate(listae2[i]+' '+str(listae1[i]), (-0.25 + i, list

plt.title("Mes de Enero")
freq_series = pd.Series(listae1)
ax = freq_series.plot(kind='bar')
ax.set_xticklabels(listae, rotation = 35)
plt.xlabel('Fechas', fontsize=20, fontdict=font)
plt.ylabel('Dosis', fontsize=20, fontdict=font)
# Saving the plot as a 'png'
plt.savefig('Enero.png')
```



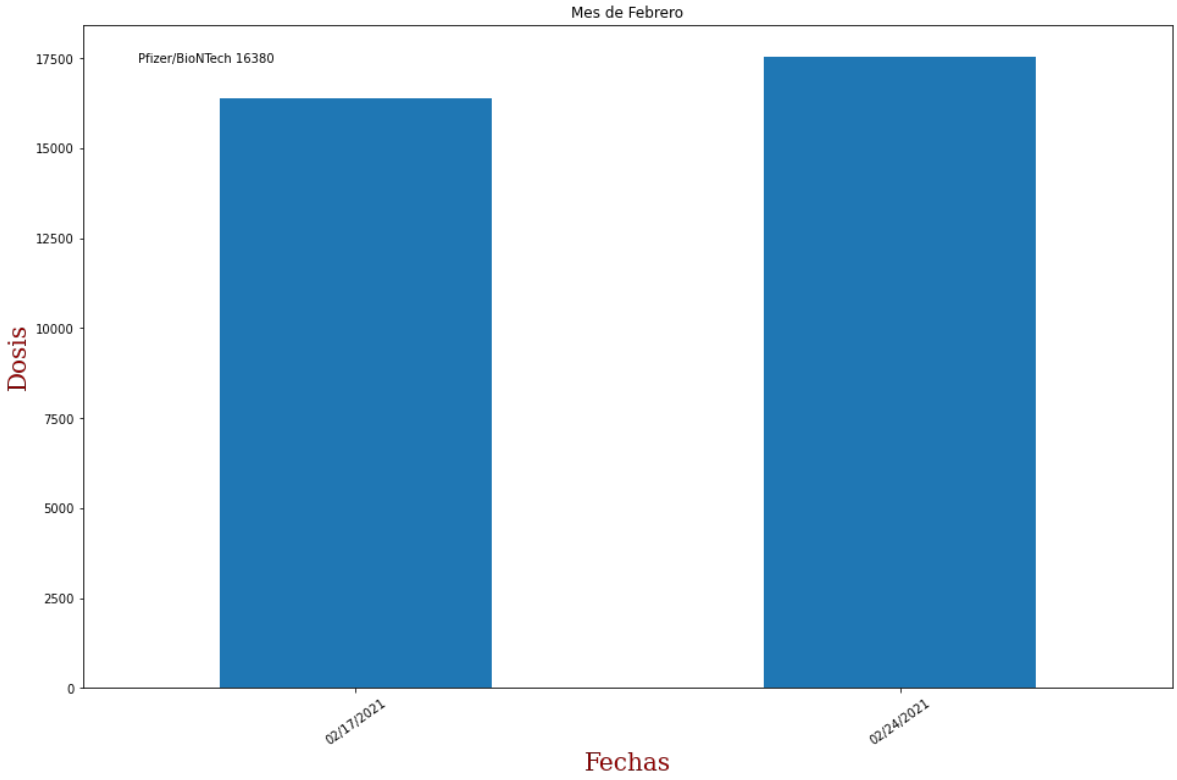
In [9]:

```
listaf =[]
listaf1 =[]
listaf2 =[]

febrero = fabricantes.loc[fabricantes.arrived_at.dt.month == 2]

for i in febrero['arrived_at']:
    date_time = i
    d = date_time.strftime("%m/%d/%Y")
    listaf.append(d)
print(listaf)

for i in febrero['total']:
    listaf1.append(int(i))
print(listaf1)

for i in febrero['vaccine']:
    listaf2.append((i))
print(listaf2)
```

```
['02/17/2021', '02/24/2021']
[16380, 17550]
['Pfizer/BioNTech', 'Pfizer/BioNTech']
```

In [18]:

```
# Declaring the figure or the plot (y, x) or (width, height)
plt.figure(figsize = (16,10))

# Annotating the bar plot with the values (Fabricantes)
for i in range(len(listaf)):
    plt.annotate(listaf2[i]+' '+str(listaf1[i]), (-0.4 + i, lista

plt.title("Mes de Febrero")
freq_series = pd.Series(listaf1)
ax = freq_series.plot(kind='bar')
ax.set_xticklabels(listaf, rotation = 35)
plt.xlabel('Fechas', fontsize=20, fontdict=font)
plt.ylabel('Dosis', fontsize=20, fontdict=font)
# Saving the plot as a 'png'
plt.savefig('Febrero.png')
```



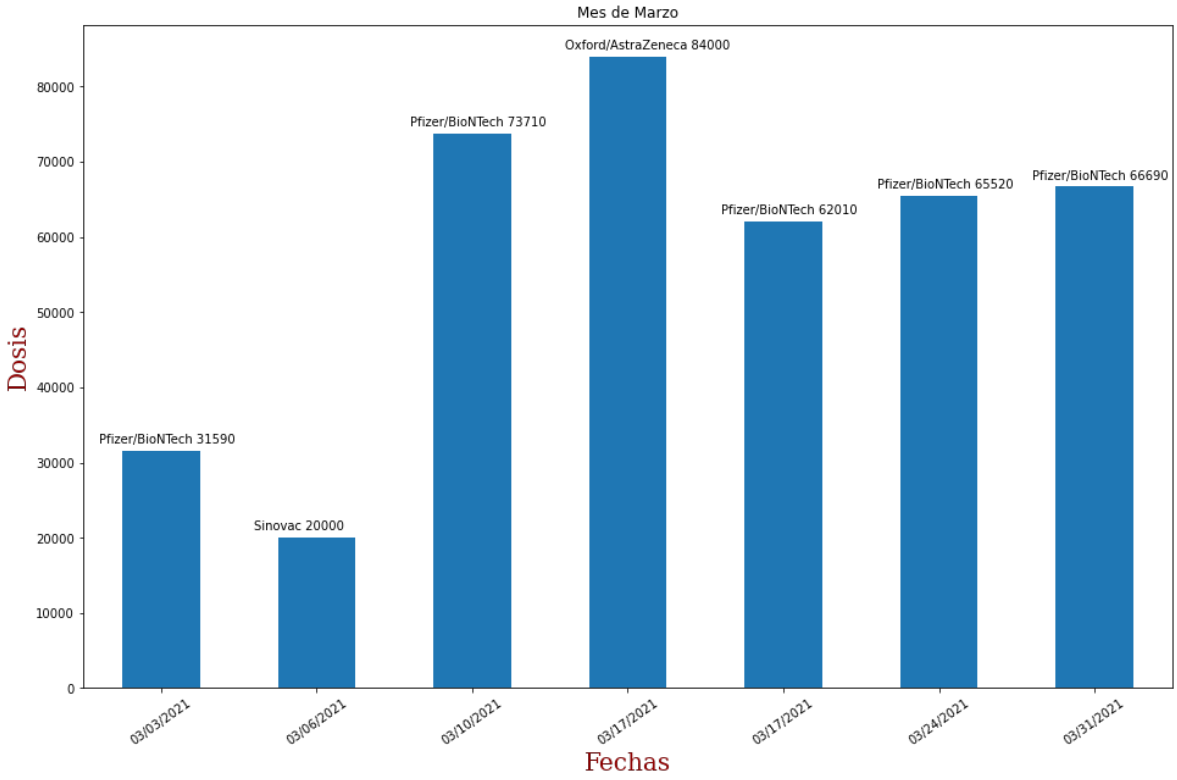
In [14]:

```
lista =[]
lista1 =[]
lista2 =[]

marzo = fabricantes.loc[fabricantes.arrived_at.dt.month == 3]

for i in marzo['arrived_at']:
    date_time = i
    d = date_time.strftime("%m/%d/%Y")
    lista.append(d)
print(lista)

for i in marzo['total']:
    lista1.append(int(i))
print(lista1)

for i in marzo['vaccine']:
    lista2.append((i))
print(lista2)
```

```
['03/03/2021', '03/06/2021', '03/10/2021', '03/17/2021', '03/17/2
021', '03/24/2021', '03/31/2021']
[31590, 20000, 73710, 84000, 62010, 65520, 66690]
['Pfizer/BioNTech', 'Sinovac', 'Pfizer/BioNTech', 'Oxford/AstraZe
neca', 'Pfizer/BioNTech', 'Pfizer/BioNTech', 'Pfizer/BioNTech']
```

In [17]:

```
# Declaring the figure or the plot (y, x) or (width, height)
plt.figure(figsize = (16,10))

# Annotating the bar plot with the values (Fabricantes)
for i in range(len(lista)):
    plt.annotate(lista2[i]+' '+str(lista1[i]), (-0.4 + i, lista1[

plt.title("Mes de Marzo")
freq_series = pd.Series(lista1)
ax = freq_series.plot(kind='bar')
ax.set_xticklabels(lista, rotation = 35)
plt.xlabel('Fechas', fontsize=20, fontdict=font)
plt.ylabel('Dosis', fontsize=20, fontdict=font)
# Saving the plot as a 'png'
plt.savefig('Marzo.png')
```



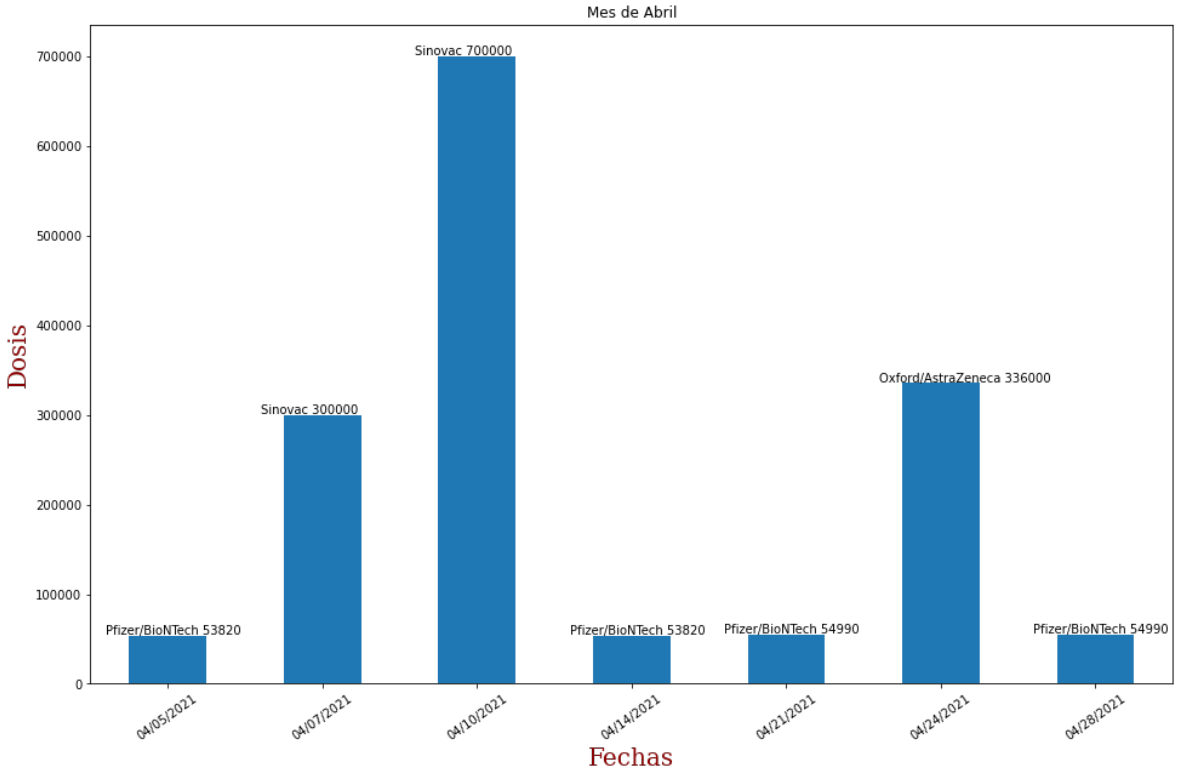
In [19]:

```
listaa =[]
listaa1 =[]
listaa2 =[]

abril = fabricantes.loc[fabricantes.arrived_at.dt.month == 4]

for i in abril['arrived_at']:
    date_time = i
    d = date_time.strftime("%m/%d/%Y")
    listaa.append(d)
print(listaa)

for i in abril['total']:
    listaa1.append(int(i))
print(listaa1)

for i in abril['vaccine']:
    listaa2.append((i))
print(listaa2)
```

```
['04/05/2021', '04/07/2021', '04/10/2021', '04/14/2021', '04/21/2
021', '04/24/2021', '04/28/2021']
[53820, 300000, 700000, 53820, 54990, 336000, 54990]
['Pfizer/BioNTech', 'Sinovac', 'Sinovac', 'Pfizer/BioNTech', 'Pfi
zer/BioNTech', 'Oxford/AstraZeneca', 'Pfizer/BioNTech']
```

In [31]:

```
# Declaring the figure or the plot (y, x) or (width, height)
plt.figure(figsize = (16,10))

# Annotating the bar plot with the values (Fabricantes)
for i in range(len(listaa)):
    plt.annotate(listaa2[i]+' '+str(listaa1[i]), (-0.4 + i, lista

plt.title("Mes de Abril")
freq_series = pd.Series(listaa1)
ax = freq_series.plot(kind='bar')
ax.set_xticklabels(listaa, rotation = 35)
plt.xlabel('Fechas', fontsize=20, fontdict=font)
plt.ylabel('Dosis', fontsize=20, fontdict=font)
# Saving the plot as a 'png'
plt.savefig('Abril.png')
```



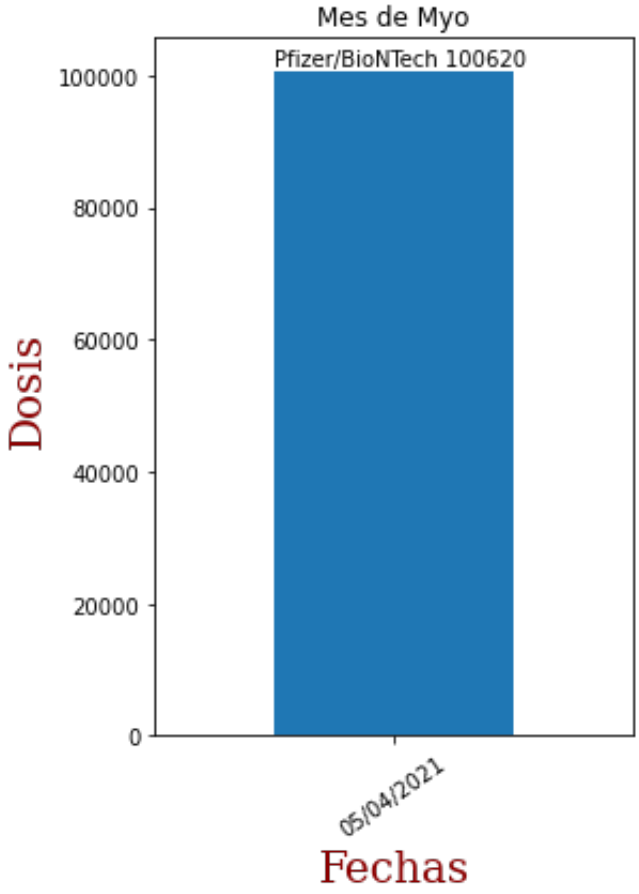
In [21]:

```
listam =[]
listam1 =[]
listam2 =[]

mayo = fabricantes.loc[fabricantes.arrived_at.dt.month == 5]

for i in mayo['arrived_at']:
    date_time = i
    d = date_time.strftime("%m/%d/%Y")
    listam.append(d)
print(listam)

for i in mayo['total']:
    listam1.append(int(i))
print(listam1)

for i in mayo['vaccine']:
    listam2.append((i))
print(listam2)
```

```
['05/04/2021']
[100620]
['Pfizer/BioNTech']
```

In [28]:

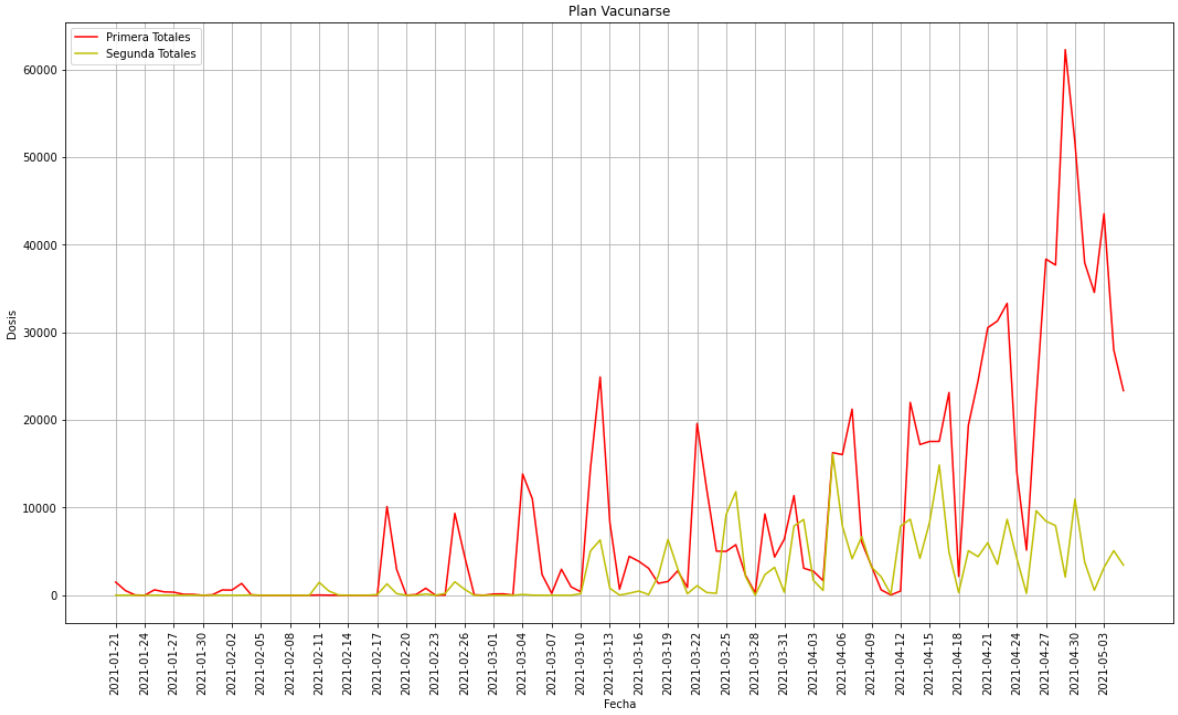
```
# Declaring the figure or the plot (y, x) or (width, height)
plt.figure(figsize = (4,6))

# Annotating the bar plot with the values (Fabricantes)
for i in range(len(listam)):
    plt.annotate(listam2[i]+' '+str(listam1[i]), (-0.25 + i, list

plt.title("Mes de Myo")
freq_series = pd.Series(listam1)
ax = freq_series.plot(kind='bar')
ax.set_xticklabels(listam, rotation = 35)
plt.xlabel('Fechas', fontsize=20, fontdict=font)
plt.ylabel('Dosis', fontsize=20, fontdict=font)
# Saving the plot as a 'png'
plt.savefig('Mayo.png')
```



- Generar un reporte parametrizado que pueda ingresar los datos de las fechas inicio y fin para obtener la información de las graficas vistas en el primer punto.

In [70]:

```
plt.figure(figsize=(18,10))

plt.title('Plan Vacunarse')
plt.plot(vacunas_planvacunarse.fecha, vacunas_planvacunarse.prime
plt.plot(vacunas_planvacunarse.fecha, vacunas_planvacunarse.segun

plt.xticks(vacunas_planvacunarse.fecha[::3].tolist())

plt.xlabel('Fecha')
plt.xticks(rotation=90)
plt.ylabel('Dosis')
plt.legend()
plt.grid(True)
plt.show()
```



- Generar un modelo matemático de predicción para regresión lineal, exponencial, polinómico y logarítmico, del procesos de vacunación en base al n umero actual de vacunados (1 y 2 dosis) y a la llegada de nuevas vacunas.

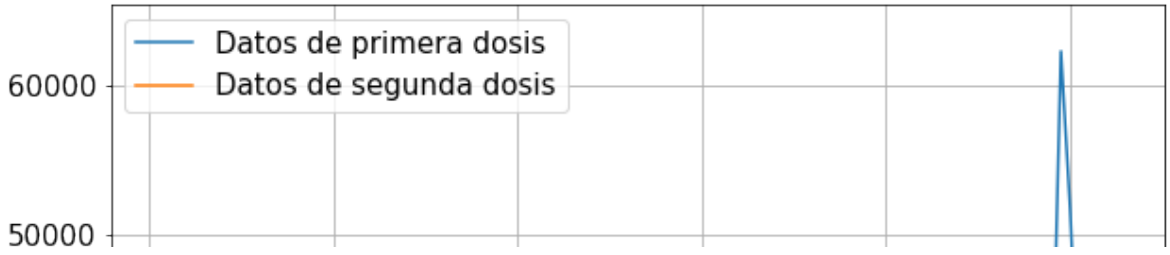
In [2]:
```
vacunas_planvacunarse = pd.read_csv('ecuacovid-master/datos_crudo
#imprimir los primeros 5 datos del archivo
vacunas_planvacunarse.head(5)
```

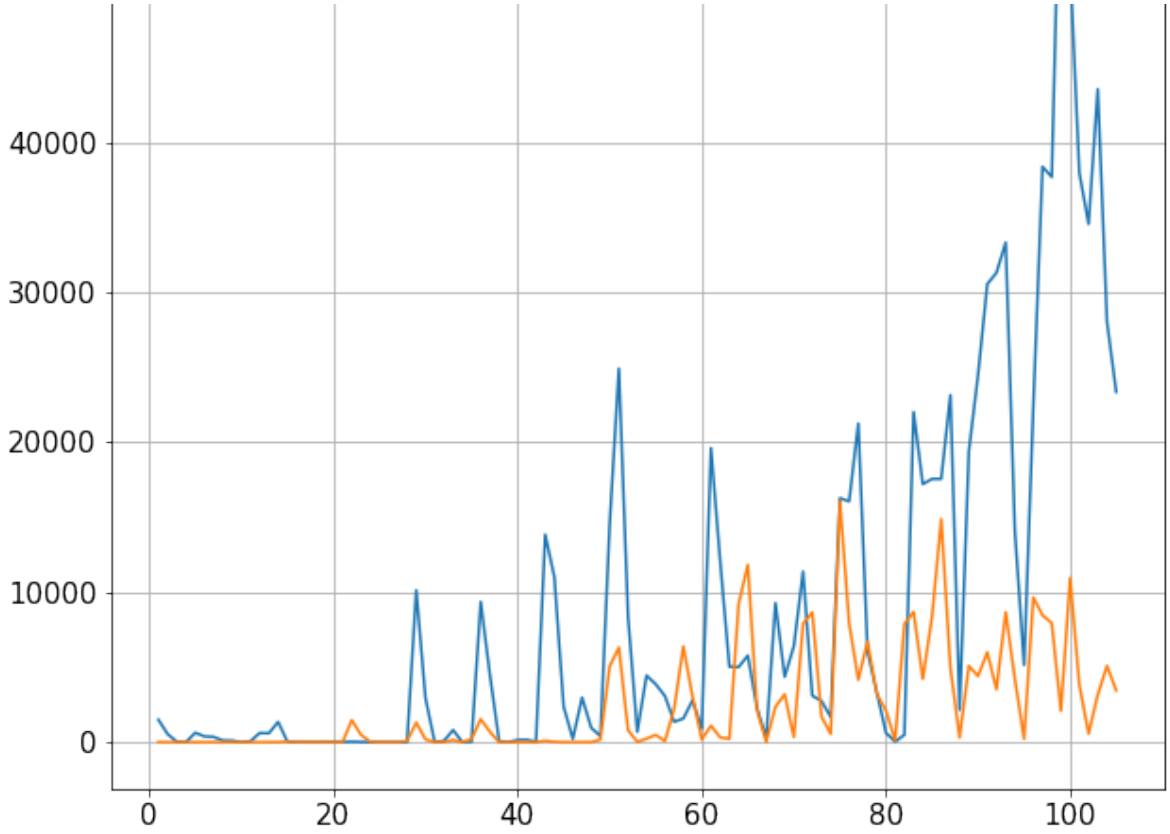
Out[2]:

	fecha	primera_dosis	segunda_dosis
0	2021-01-21	1500	0
1	2021-01-22	538	1
2	2021-01-23	31	0
3	2021-01-24	0	0
4	2021-01-25	622	0

Lineal

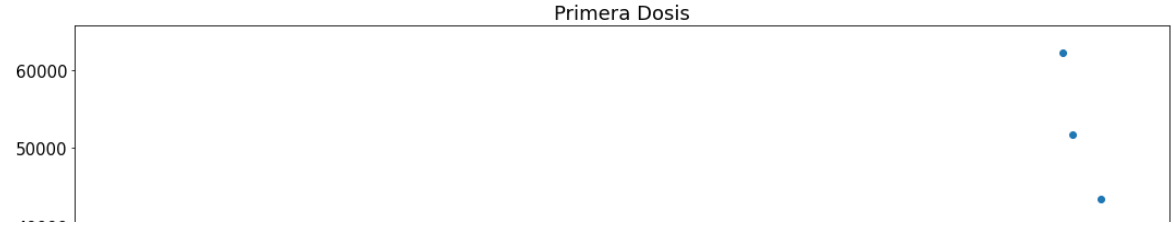
In [29]:
```
from sklearn.linear_model import LinearRegression
from sklearn import linear_model

#x = range(1,len(vacunas_planvacunarse)+1)
#y = vacunas_planvacunarse

#vacunas_planvacunarse['fecha'] = vacunas_planvacunarse['fecha'].
#vacunas_planvacunarse['fecha'] = vacunas_planvacunarse['fecha'].

start_date = "2021/01/20"

plt.rcParams['figure.figsize'] = [10,10]
plt.rc('font', size=15)
#confirmados = vacunas_planvacunarse.iloc[:105].loc[start_date:]
#vaccine2 = fabricantes.loc[fabricantes.vaccine == 'Sinovac']['to
primera_dosis = vacunas_planvacunarse.loc[:]['primera_dosis']
segunda_dosis = vacunas_planvacunarse.loc[:]['segunda_dosis']
#print(confirmados)

x = range(1,len(primera_dosis)+1)
y = primera_dosis
z = segunda_dosis

plt.plot(x, y, label = "Datos de primera dosis")
plt.plot(x, z, label = "Datos de segunda dosis")
plt.legend()
plt.grid(True)
plt.show()
```

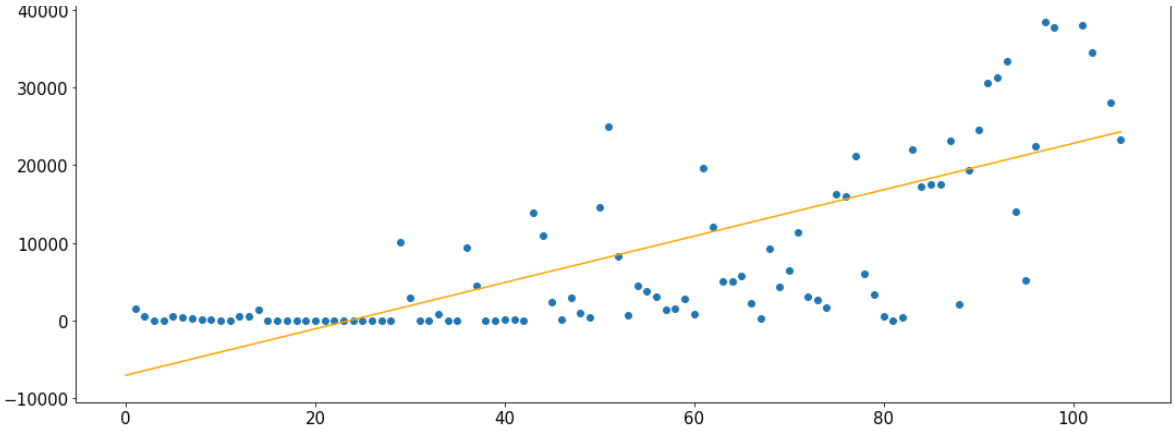




In [185]:

```
plt.figure(figsize=(18,10))

plt.title('Primera Dosis')

regr = linear_model.LinearRegression()
regr.fit(np.array(x).reshape(-1, 1) ,y)



y_prediccion = regr.predict([[107]])


plt.scatter(x,y)

x_real = np.array(range(0, 106))
ypred = regr.predict(x_real.reshape(-1, 1))



plt.plot(x_real, ypred, color='orange')
plt.show()

print("prediccion lineal para el 1 dia despues de la primera dosi
print("prediccion lineal para los 2 dias despues de la segunda do
print(ypred)

```



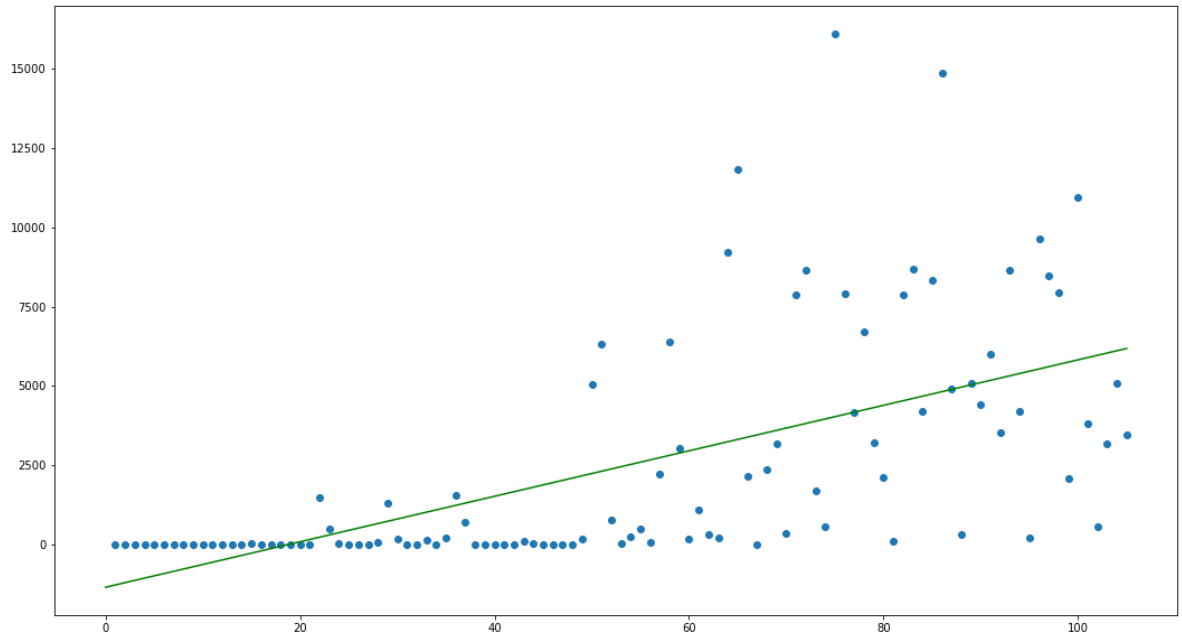



```
prediccion lineal para el 1 dia despues de la primera dosis: 2400
2.981304858666
prediccion lineal para los 2 dias despues de la segunda dosis: 24
301.1432165319
[-7005.85750916 -6707.69559748 -6409.53368581 -6111.37177414
 -5813.20986246 -5515.04795079 -5216.88603912 -4918.72412744
 -4620.56221577 -4322.4003041  -4024.23839243 -3726.07648075
 -3427.91456908 -3129.75265741 -2831.59074573 -2533.42883406
 -2235.26692239 -1937.10501071 -1638.94309904 -1340.78118737
 -1042.61927569  -744.45736402  -446.29545235  -148.13354067
   150.028371     448.19028267   746.35219435  1044.51410602
  1342.67601769  1640.83792937  1938.99984104  2237.16175271
  2535.32366439  2833.48557606  3131.64748773  3429.80939941
  3727.97131108  4026.13322275  4324.29513443  4622.4570461
  4920.61895777  5218.78086945  5516.94278112  5815.10469279
  6113.26660446  6411.42851614  6709.59042781  7007.75233948
  7305.91425116  7604.07616283  7902.2380745   8200.39998618
  8498.56189785  8796.72380952  9094.8857212   9393.04763287
  9691.20954454  9989.37145622 10287.53336789 10585.69527956
 10883.85719124 11182.01910291 11480.18101458 11778.34292626
 12076.50483793 12374.6667496  12672.82866128 12970.99057295
 13269.15248462 13567.3143963  13865.47630797 14163.63821964
 14461.80013132 14759.96204299 15058.12395466 15356.28586633
 15654.44777801 15952.60968968 16250.77160135 16548.93351303
 16847.0954247  17145.25733637 17443.41924805 17741.58115972
 18039.74307139 18337.90498307 18636.06689474 18934.22880641
 19232.39071809 19530.55262976 19828.71454143 20126.87645311
 20425.03836478 20723.20027645 21021.36218813 21319.5240998
 21617.68601147 21915.84792315 22214.00983482 22512.17174649
 22810.33365817 23108.49556984 23406.65748151 23704.81939319
 24002.98130486 24301.14321653]
```

In [168]:

```
plt.figure(figsize=(18,10))
regr.fit(np.array(x).reshape(-1, 1) ,z)

z_prediccion = regr.predict([[107]])

plt.scatter(x,z)

x_real = np.array(range(0, 106))
zpred = regr.predict(x_real.reshape(-1, 1))

plt.plot(x_real, zpred, color='green')
plt.show()
```

```
14 print("prediccion lineal para el 1 dia despues de la primera dosi
15 print("prediccion lineal para los 2 dias despues de la segunda do
16 print(zpred)
```



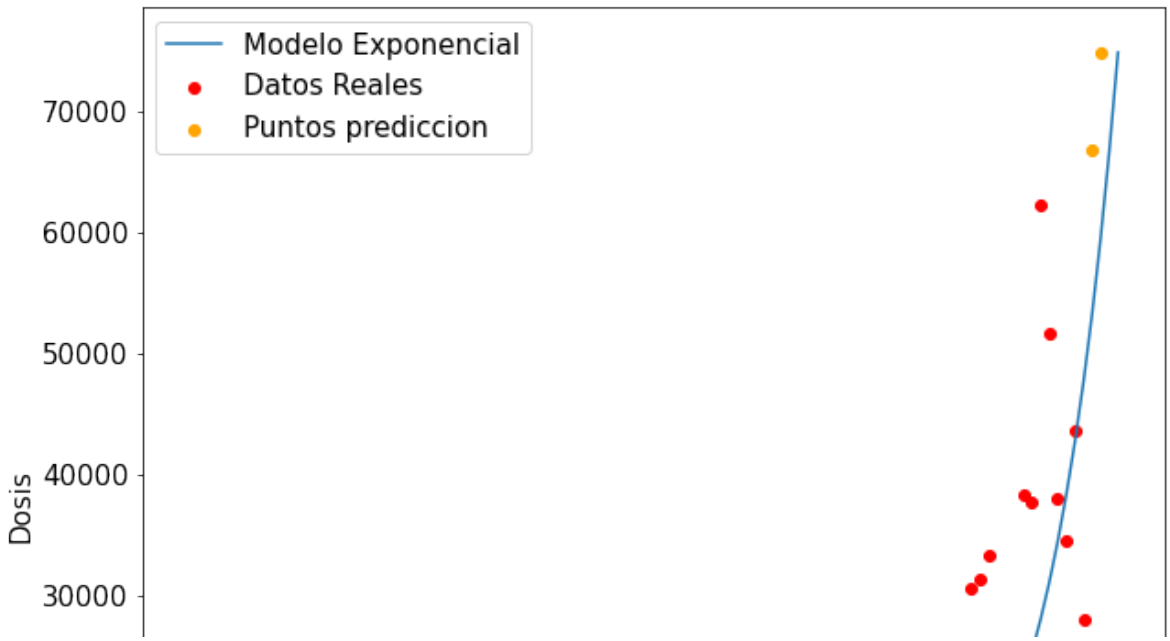

```
prediccion lineal para el 1 dia despues de la primera dosis: 6111
.855822793558
prediccion lineal para los 2 dias despues de la segunda dosis: 61
83.605750224618
[-1350.13663004 -1278.38670261 -1206.63677517 -1134.88684774
 -1063.13692031  -991.38699288  -919.63706545  -847.88713802
  -776.13721059  -704.38728316  -632.63735573  -560.88742829
  -489.13750086  -417.38757343  -345.637646    -273.88771857
  -202.13779114  -130.38786371   -58.63793628    13.11199115
    84.86191858   156.61184602   228.36177345   300.11170088
   371.86162831   443.61155574   515.36148317   587.1114106
   658.86133803   730.61126546   802.3611929    874.11112033
   945.86104776  1017.61097519  1089.36090262  1161.11083005
  1232.86075748  1304.61068491  1376.36061234  1448.11053977
  1519.86046721  1591.61039464  1663.36032207  1735.1102495
  1806.86017693  1878.61010436  1950.36003179  2022.10995922
  2093.85988665  2165.60981409  2237.35974152  2309.10966895
  2380.85959638  2452.60952381  2524.35945124  2596.10937867
  2667.8593061   2739.60923353  2811.35916096  2883.1090884
  2954.85901583  3026.60894326  3098.35887069  3170.10879812
  3241.85872555  3313.60865298  3385.35858041  3457.10850784
  3528.85843528  3600.60836271  3672.35829014  3744.10821757
  3815.858145    3887.60807243  3959.35799986  4031.10792729
  4102.85785472  4174.60778215  4246.35770959  4318.10763702
  4389.85756445  4461.60749188  4533.35741931  4605.10734674
  4676.85727417  4748.6072016   4820.35712903  4892.10705647
  4963.8569839   5035.60691133  5107.35683876  5179.10676619
  5250.85669362  5322.60662105  5394.35654848  5466.10647591
  5537.85640335  5609.60633078  5681.35625821  5753.10618564
  5824.85611307  5896.6060405   5968.35596793  6040.10589536
  6111.85582279  6183.60575022]
```

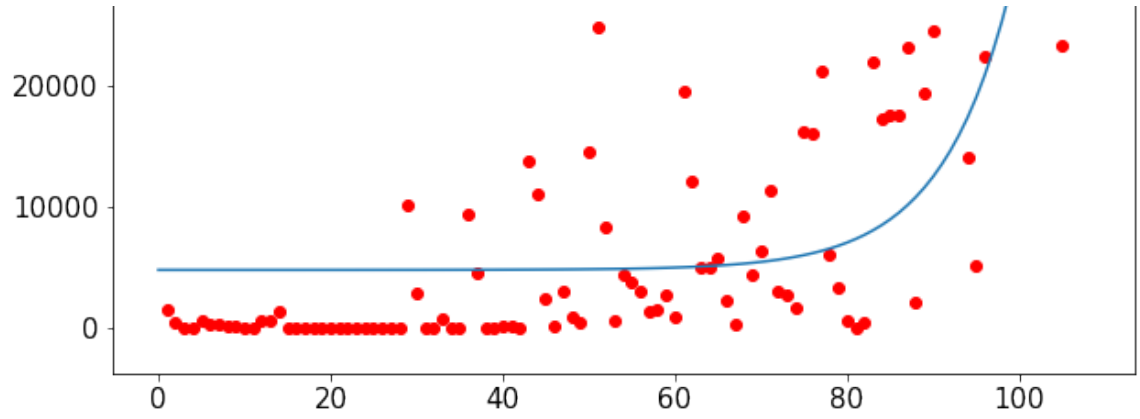
Exponencial

In [27]:

```
from scipy.optimize import curve_fit
def exponencial_model(x,a,b):
    return a+b*np.exp(x*b)

exp_fit = curve_fit(exponencial_model,x,y)
print(exp_fit)


pred_x = list(range(0,max(x)+4))
plt.rcParams['figure.figsize'] = [10,10]
plt.rc('font', size=15)

plt.scatter(x,y,label="Datos Reales",color="red")
# Predicted exponential curve
puntosreales = [exponencial_model(i,exp_fit[0][0],exp_fit[0][1])

puntosprediccion = [exponencial_model(i,exp_fit[0][0],exp_fit[0][
predi = [round(puntosprediccion[x[len(x)-1]+2]),round(puntospredi


plt.plot(pred_x, puntosprediccion, label="Modelo Exponencial" )
plt.scatter(range(max(x),max(x)+2),predi,label="Puntos prediccion
plt.legend()
plt.ylabel("Dosis")
plt.show()

print("La prediccion para el 1 dia despues de la primera dosis: "
print("La prediccion para del 2 dia despues de la primera dosis:
```

```
(array([4.80196428e+03, 1.22723498e-01]), array([[ 8.44926097e+05
, -2.88497739e-01],
       [-2.88497739e-01,  6.84604420e-07]]))
```



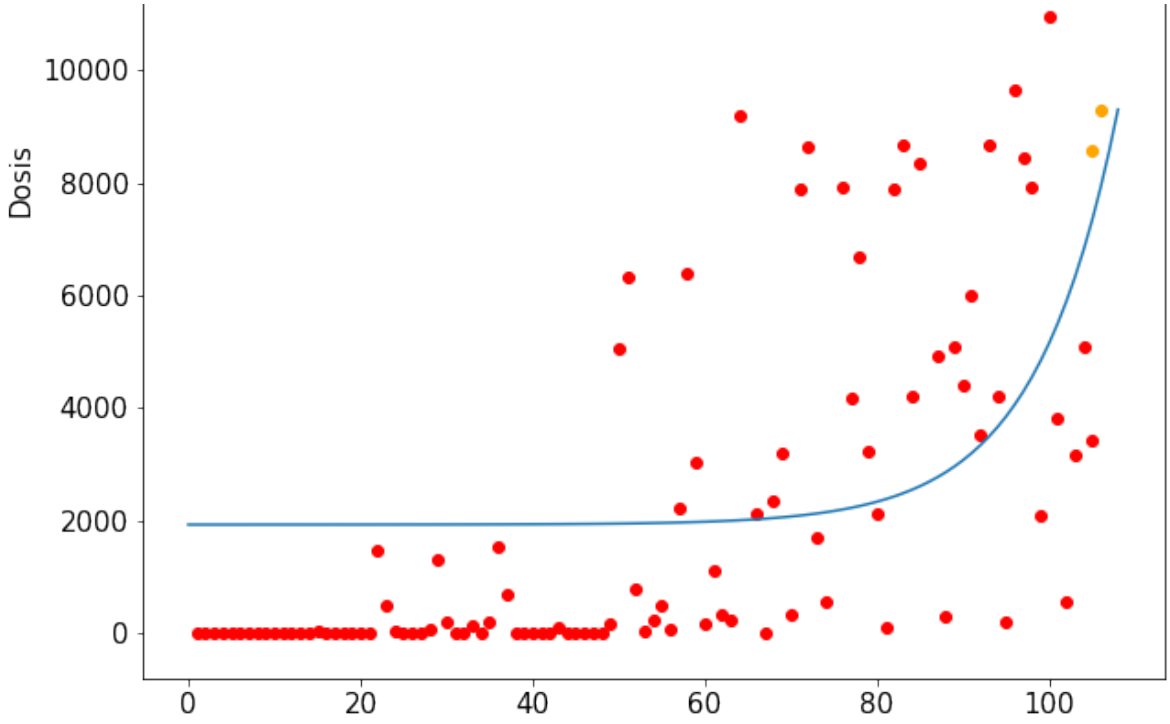



```
La prediccion para el 1 dia despues de la primera dosis:  66722
La prediccion para del 2 dia despues de la primera dosis:  74807
```

In [28]:

```
exp_fit = curve_fit(exponencial_model,x,z)
print(exp_fit)


pred_x = list(range(0,max(x)+4))
plt.rcParams['figure.figsize'] = [10,10]
plt.rc('font', size=15)

plt.scatter(x,z,label="Datos Reales",color="red")
# Predicted exponential curve
puntosreales = [exponencial_model(i,exp_fit[0][0],exp_fit[0][1])

puntosprediccion = [exponencial_model(i,exp_fit[0][0],exp_fit[0][
predi = [round(puntosprediccion[x[len(x)-1]+2]),round(puntospredi


plt.plot(pred_x, puntosprediccion, label="Modelo Exponencial " )
plt.scatter(range(max(x),max(x)+2),predi,label="Puntos prediccion
plt.legend()

plt.ylabel("Dosis")
plt.show()

print("La prediccion para el 1 dia despues de la primera dosis: "
print("La prediccion para del 2 dia despues de la primera dosis:
```

```
(array([1.92842290e+03, 1.03471785e-01]), array([[ 1.34462527e+05
, -4.08878328e-01],
       [-4.08878328e-01,  7.39347539e-06]]))
```



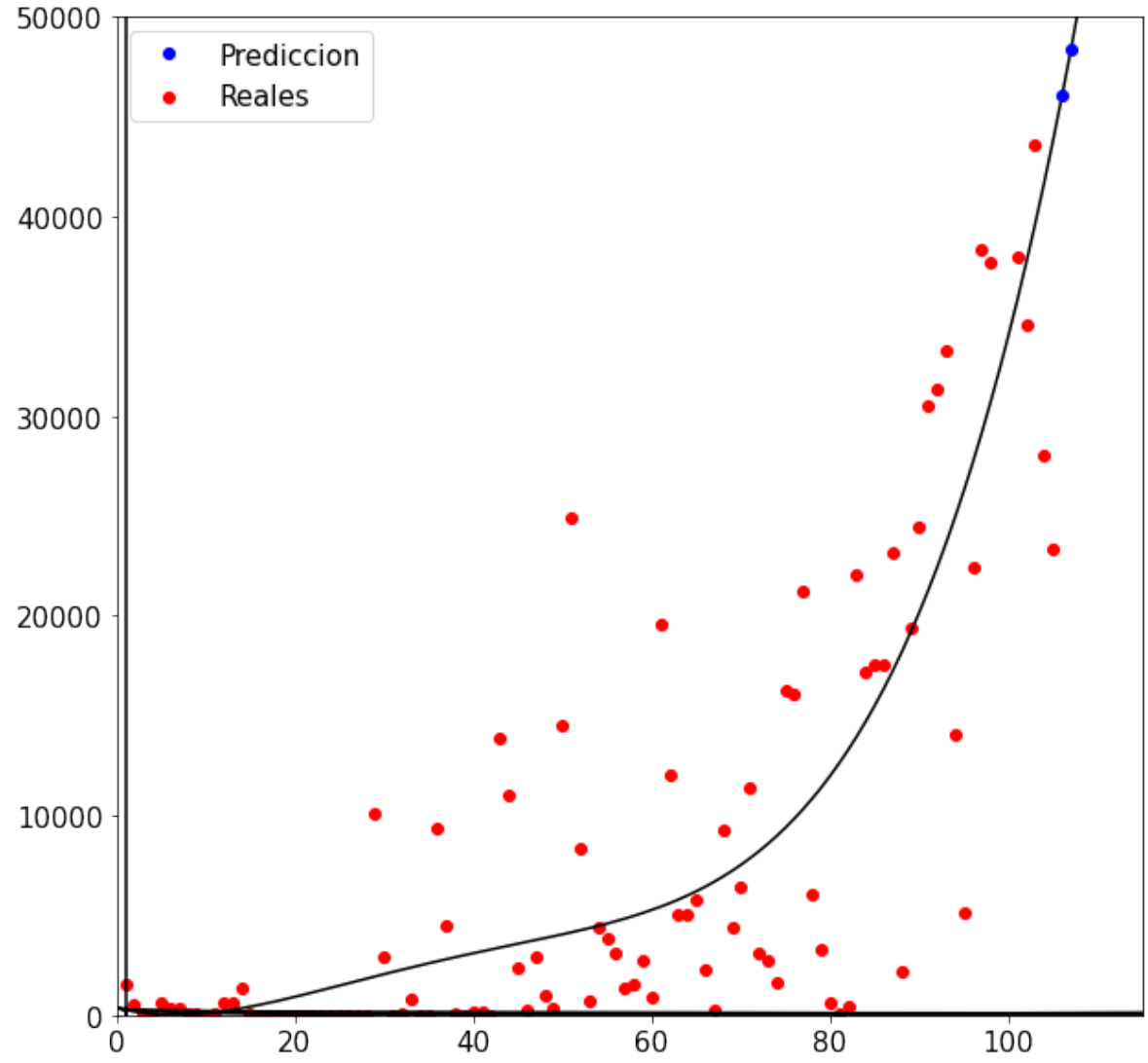

```
La prediccion para el 1 dia despues de la primera dosis:  8583
La prediccion para del 2 dia despues de la primera dosis:  9308
```

Polinomial

In [25]:

```
from sklearn.preprocessing import PolynomialFeatures
pf = PolynomialFeatures(degree = 4)    #polinomio de grado 4
X = pf.fit_transform(np.array(x).reshape(-1, 1))

plt.rcParams['figure.figsize'] = [10,10]
plt.rc('font', size=15)

regresion_lineal = LinearRegression()

regresion_lineal.fit(X, y)

pred_x = list(range(0,max(x)+107))

fil = pf.fit_transform(np.array(pred_x).reshape(-1, 1))

fpredictpol = regresion_lineal.predict(fil)
onlypredicty = [fpredictpol[max(x)+1], fpredictpol[max(x)+2] ]
xpredict = range(max(x)+1,max(x)+3)
plotpol.plot(fil, fpredictpol, color='black')
plotpol.scatter(x,y,label="Reales",color="red")
plotpol.plot(xpredict,onlypredicty, 'ob',label="Prediccion")
plotpol.ylim(0,50000)
plotpol.xlim(0,115)
plotpol.legend()
plotpol.show()


print("prediccion polinomial para 1 dias despues: "+str(onlypredi
```

```
print("prediccion polinomial para 2 dias despues: "+str( onlypred
```

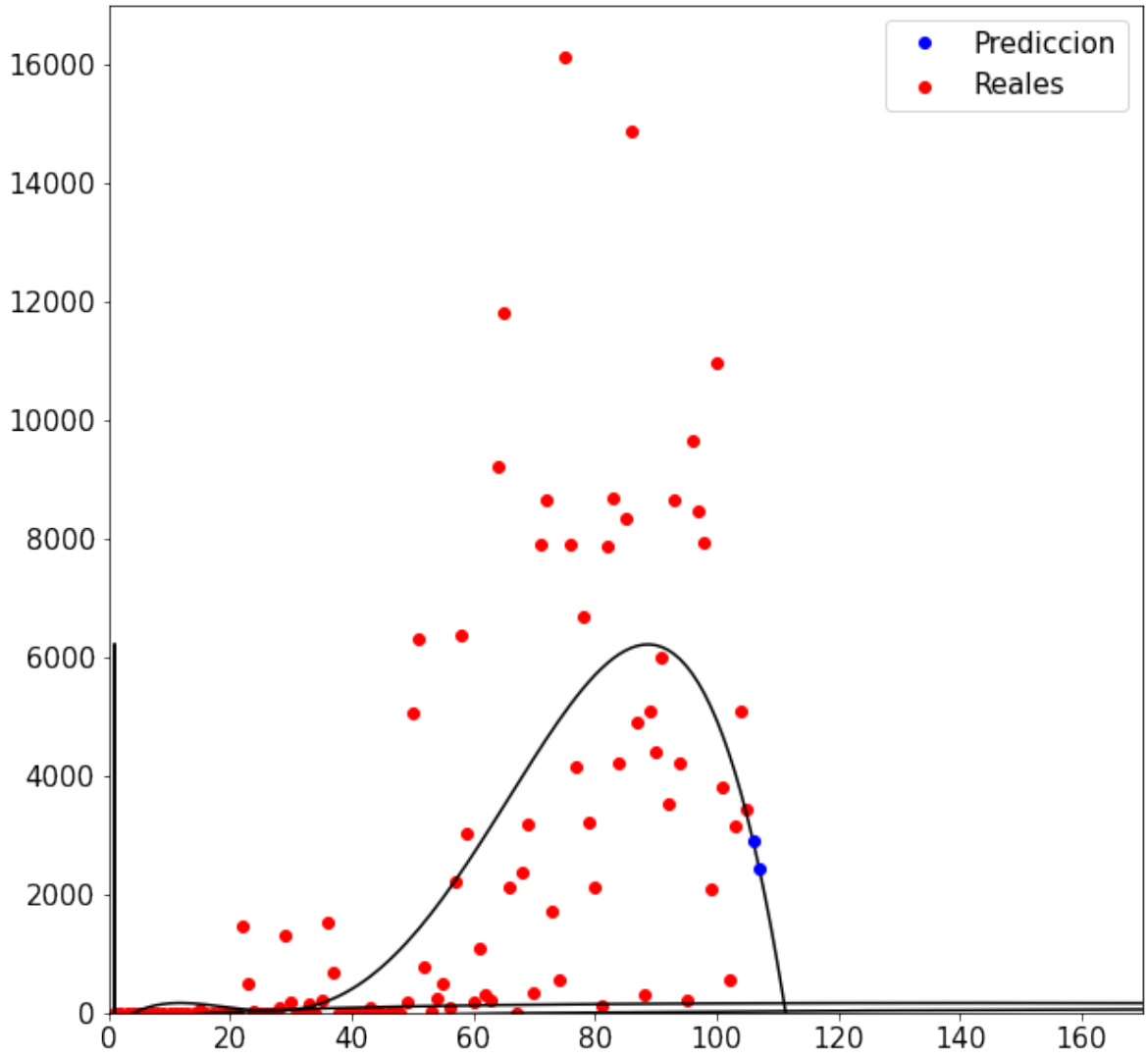



```
prediccion polinomial para 1 dias despues: 46017.55025014725
prediccion polinomial para 2 dias despues: 48333.91977340066
```

In [24]:

```
pf = PolynomialFeatures(degree = 4)    #polinomio de grado 4
X = pf.fit_transform(np.array(x).reshape(-1, 1))

plt.rcParams['figure.figsize'] = [10,10]
plt.rc('font', size=15)

regresion_lineal = LinearRegression()

regresion_lineal.fit(X, z)

pred_x = list(range(0,max(x)+107))

fil = pf.fit_transform(np.array(pred_x).reshape(-1, 1))

fpredictpol = regresion_lineal.predict(fil)
onlypredicty = [fpredictpol[max(x)+1], fpredictpol[max(x)+2] ]
xpredict = range(max(x)+1,max(x)+3)
plotpol.plot(fil, fpredictpol, color='black')
plotpol.scatter(x,z,label="Reales",color="red")
plotpol.plot(xpredict,onlypredicty, 'ob',label="Prediccion")
```

```
plotpol.ylim(0,17000)
plotpol.xlim(0,170)
plotpol.legend()
plotpol.show()


print("prediccion polinomial para 1 dias despues: "+str(onlypredi
print("prediccion polinomial para 2 dias despues: "+str( onlypred
```



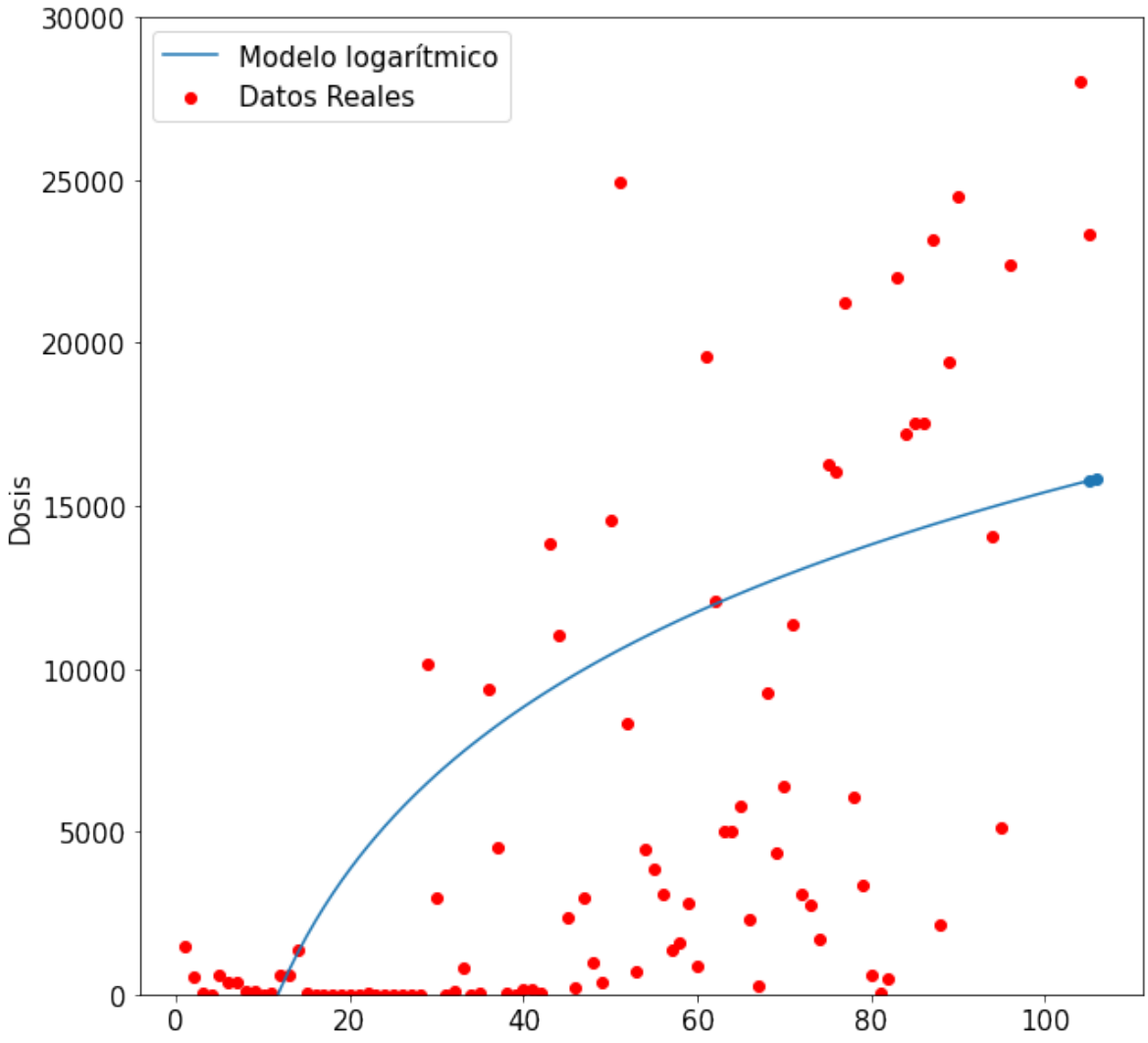

```
prediccion polinomial para 1 dias despues: 2904.646905600739
prediccion polinomial para 2 dias despues: 2447.263554780473
```

Logarítmico

In [21]:

```
from scipy.optimize import curve_fit
def logistic_model(x,a,b):
    return a+b*np.log(x)

exp_fit = curve_fit(logistic_model,x,y) #Extraemos los valores de

pred_x = list(range(min(x),max(x)+2))
plt.scatter(x,y,label="Datos Reales",color="red")
plt.rcParams['figure.figsize'] = [10,10]
```

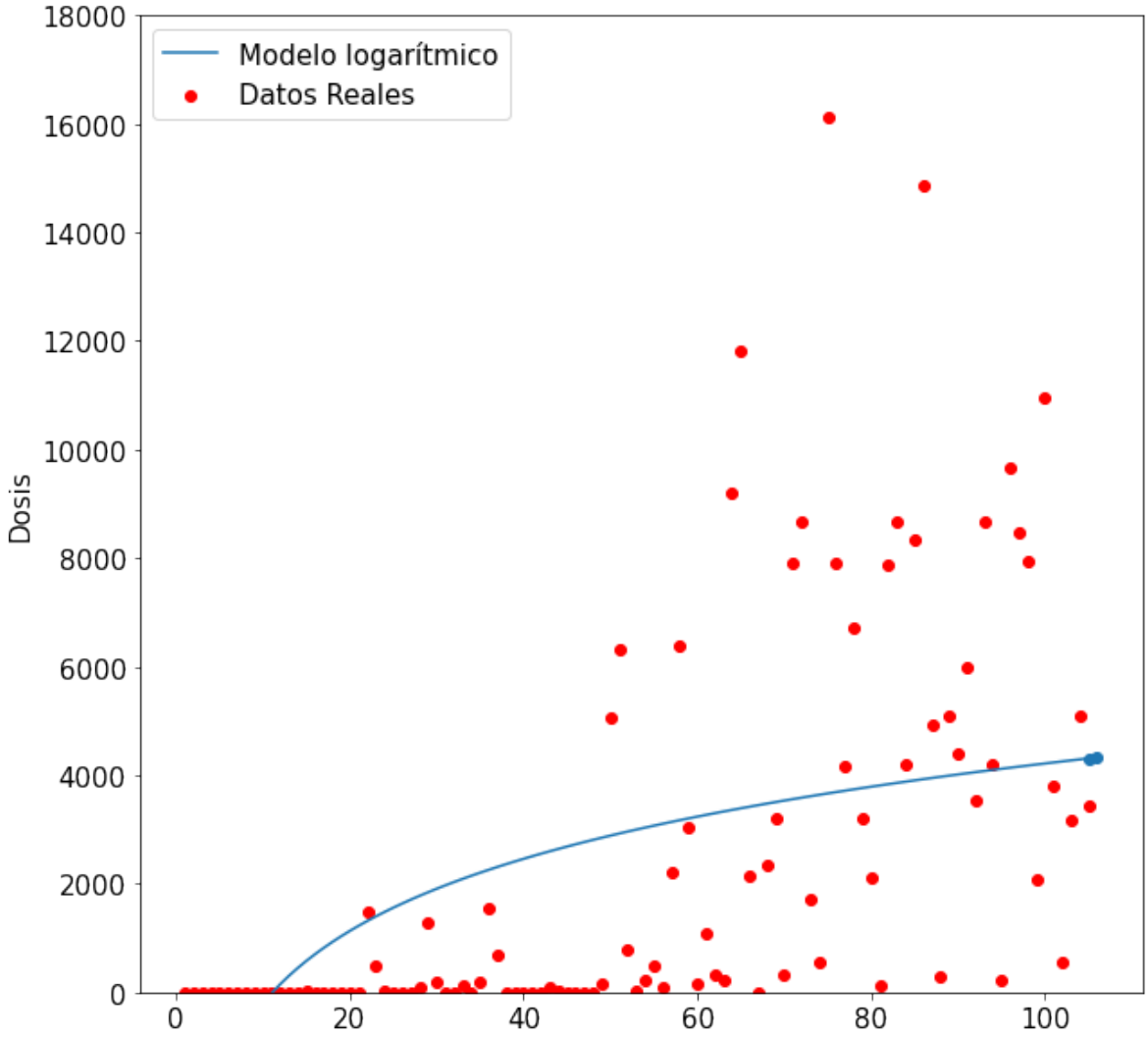
```
plt.rc('font', size=15)
predictf = [logistic_model(i,exp_fit[0][0],exp_fit[0][1]) for i i
onlypredict = [ predictf[len(predictf)-2], predictf[len(predictf)
plt.scatter(range(len(x),len(x)+2),onlypredict)
plt.plot(pred_x, predictf, label="Modelo logarítmico" )
plt.ylabel("Dosis")
plt.ylim(0,30000)
plt.legend()

plt.show()

print("Prediccion logarítmico para el 1 dia despues de la primera
print("Prediccion logarítmico para los 2 dias despues de la segun

```




```
Prediccion logarítmico para el 1 dia despues de la primera dosis:
: 15763.996631332997
Prediccion logarítmico para los 2 dias despues de la segunda dosi
s: 15832.144842489291
```

In [20]:

```
exp_fit = curve_fit(logistic_model,x,z) #Extraemos los valores de

pred_x = list(range(min(x),max(x)+2))
plt.scatter(x,z,label="Datos Reales",color="red")
plt.rcParams['figure.figsize'] = [10,10]
plt.rc('font', size=15)
predictf = [logistic_model(i,exp_fit[0][0],exp_fit[0][1]) for i i
onlypredict = [ predictf[len(predictf)-2], predictf[len(predictf)
plt.scatter(range(len(x),len(x)+2),onlypredict)
plt.plot(pred_x, predictf, label="Modelo logarítmico" )
plt.ylabel("Dosis")
plt.ylim(0,18000)
plt.legend()

plt.show()

print("Prediccion logarítmico para el 1 dia despues de la primera
print("Prediccion logarítmico para los 2 dias despues de la segun


```




```
Prediccion logarítmico para el 1 dia despues de la primera dosis:
: 4309.14697464072
Prediccion logarítmico para los 2 dias despues de la segunda dosi
s: 4327.306117838289
```

In [ ]:

```
1
```