I. Diagrama de Componentes

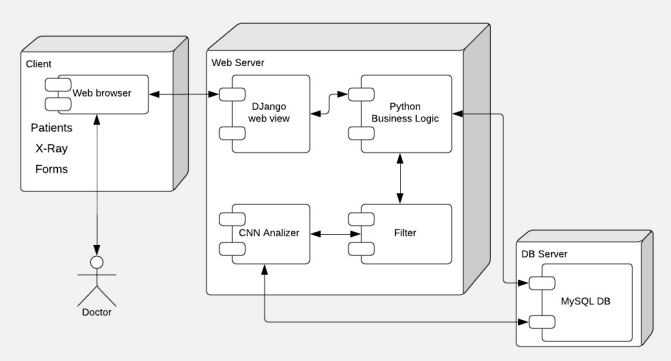

Fig. 1. Diagrama de Componentes.

A. *Componentes*

- **Doctor:** es el usuario principal del sistema y encargado de ingresar los datos de los pacientes, imágenes de los rayos-X o buscar a el paciente en caso de que ya tenga historial.
- **Web browser:** al ser una aplicación web, el usuario debe utilizar un web browser y dirigirse a la dirección web correcta.
- **DJango web view:** componente creado con la herramienta DJango, es la parte que va a interactuar con el usuario, el front end de la aplicación.
- **Python Business Logic:** es el back end de la aplicación, debe tener comunicación con la base de datos.
- **Filter:** componente que se encarga de aplicar los filtros necesarios a las imágenes, para su posterior análisis.
- **CNN Analizer:** una vez que a la imágen se le han aplicado los filtros necesarios, este componente procede a analizar y calcular la edad óosea del paciente.
- **MySQL DB:** componente de guardar toda la información obtenida, tanto del análisis de la imágen, como la de los datos personales del paciente.

II. Trozos de Código

A. *Cargar una imagen*

```
import matplotlib.pyplot as plt
import numpy as np
from pylab import imread, imshow

img = imread('D:\Esteban\Desktop\
Ezatara\EZ\EZ-Images\grim.jpg')
.astype(np.float32)

image = img / 255
imshow(image)
```

```
Python 3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 10:22:32) [MSC v.1900 64 bit
(AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 6.2.1 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/PCEsteban/.spyder-py3/temp.py', wdir='C:/Users/
PCEsteban/.spyder-py3')
```

Fig. 2. Código para cargar una imágen

- Utilización de las librerías numpy, matplotlib y pylab para cargar una imagen. Se lee la imágen con el path respectivo y luego se muestra con la función ishow que pertenece a matplotlib.

B. *Cargar archivo CSV*

```
import numpy as np

data = np.genfromtxt('D:/Esteban/Desktop/
Ezatara/EZ-U/Aseguramiento/POC/train.csv',
delimiter = ',', skip_header=1, dtype=None)

for row in data:
        print(row)
```

```
IPython console
Console 1/A
(15579, 96, True)
(15580, 162, False)
(15581, 132, False)
(15582, 150, False)
(15583, 132, True)
(15584, 54, True)
(15585, 162, True)
(15586, 192, True)
(15587, 50, False)
(15588, 113, False)
(15589, 84, True)
(15591, 162, True)
(15593, 180, True)
(15594, 108, True)
(15595, 78, True)
(15596, 72, True)
(15597, 192, True)
(15598, 120, False)
(15599, 36, True)
(15600, 144, False)
(15602, 82, False)
(15603, 106, False)
(15604, 168, True)
(15605, 50, False)
(15606, 113, False)
(15608, 55, False)
(15609, 150, True)
```

Fig. 3. Código para cargar un archivo cvs

- Se utiliza la librería numpy y su función de genfromtext para leer el archivos csv, se le pasa el path, se indica el delimitador, además de indicar en el dtype como nulo, para que no exista problemas de lectura de ints, floats, booleans o strings.

C. *Aplicación de contraste a una imágen*

```
from scipy import ndimage
import matplotlib.pyplot as plt
import numpy as np
from pylab import imread, imshow, gray, mean

img = imread('D:\Esteban\Desktop\
Ezatara\EZ\EZ-Images\grim.jpg')
.astype(np.float32)

image = mean(img,2)

plt.imshow(image, cmap=plt.cm.gray, vmin=30,
vmax=200)

plt.axis('off')
plt.show()
```

Fig. 4. Código para aplicar y manejar contraste de una imágen

- En este trozo se aplican varias funciones de las librerias numpy, matplotlib y pylab, se pasa la imágen por medio de su dirección, se hace un mapeo de la imágen en gris, se dicta la intensidad para el contraste para luego mostrar el resultado de las modificaciones.

D. *Login en Django*

```
from django.views.generic.edit
import FormView
from django.http.response
import HttpResponseRedirect
from django.core.urlresolvers
import reverse_lazy
from django.contrib.auth.forms
import AuthenticationForm
from django.contrib.auth
import login


class Login(FormView):
    template_name = 'login.html'
    form_class = AuthenticationForm
    success_url =
reverse_lazy("personas:
____home")
    def dispatch(self, request,
    *args, **kwargs):
        if request.user.is_authenti
        cated():
            return HttpResponseRed
            rect(self.get_success_
            url())
        else:
            return super(Login, self).
            dispatch(request,
            *args, **kwargs)
    def form_valid(self, form):
        login(self.request, form.
        get_user())
        return super(Login, self).
        form_valid(form)
```

Fig. 5. Logind de prueba en Django

- En este trozo se aplican varias funciones de las bibliotecas de Django. Se importa una vista génerica del tipo Formview, se importa el formulario de de autenticación de Django y se crea la clase necesaria para el funcionamiento del login.

III. Problemas Encontrados con las Herramientas

- A uno de los integrantes del grupo se le presentó un problema con la instalación de TensorFlow, que es necesario para Keras. La solución fue utlizar Anaconda para poder instalar tanto TensorFlow como todas las demás herramientas.