



MANUAL TECNICO

“DISEÑO Y DESARROLLO DE UNA PLATAFORMA
EDUCATIVA BASADA EN VISIÓN ARTIFICIAL Y
APLICACIONES MÓVILES PARA EL APOYO EN LA
ENSEÑANZA, RESCATE Y PROTECCIÓN DE LA
MEDICINA ANCESTRAL DE PUEBLOS ANDINOS.”

Manual técnico orientado al desarrollo y funcionamiento de la plataforma educativa basada en visión artificial y aplicaciones móviles para el apoyo en la enseñanza, rescate y protección de la medicina ancestral de pueblos andinos.

Franklin Villavicencio – Christian Flores
fvillavicenciob@est.ups.edu.ec – cfloreso1@est.ups.edu.ec

Tabla de contenido

1. Introducción	4
1.1. <i>Propósito</i>	4
1.2. <i>Alcance</i>	4
2. Manual de Ejecución	4
2.1. <i>Requerimientos</i>	4
2.1.1. <i>Hardware</i>	4
2.1.2. <i>Software</i>	4
2.2. <i>Diseño de la Plataforma.</i>	6
2.3. <i>Especificación detallada de los Componentes Principales</i>	7
2.3.1. <i>Especificación de la Base de Datos</i>	7
2.3.2. <i>Especificación del sistema de gestión</i>	11
2.3.3. <i>Especificación de la aplicación móvil.</i>	20
2.3.3.1.1. <i>Crear Aplicación.</i>	21
2.4. <i>Configuraciones del sistema operativo / Instalación de la Base de Datos / Creación de entornos de desarrollo / instalaciones de paquetes.</i>	23
2.4.1. <i>Creación de BD / Configuración de servidores</i>	25
2.4.2. <i>Configuración e instalación de paquetes para aplicación móvil.</i>	35
2.4.2.1. <i>Librerías.</i>	36
2.4.2.2. <i>Configuración de Tensor Flow en la aplicación móvil.</i>	36
2.4.2.3. <i>Consumir WS en la aplicación móvil.</i>	36
2.4.3. <i>Puesta en marcha del sistema.</i>	37
2.5. <i>Cuentas asociadas al sistema</i>	37
3. Consideraciones / Recomendaciones	38
3.1. <i>Consideraciones / Recomendaciones</i>	38
Referencias	39

Tabla de Figuras

Figura 1. Diagrama de estructura de la plataforma YURAKU (nombre la plataforma).....	7
Figura 2. Tablas para el manejo de usuarios de la plataforma.	8
Figura 3. Diagrama entidad-relación de la plataforma.	10
Figura 4. Tablas necesarias para el funcionamiento de la plataforma.	11
Figura 5. Tablas para el registro y/o sesión de usuarios con una red social en el sistema.	11
Figura 6. Estructura del sistema generado en Django.	12
Figura 7. Archivo model.py (modelo).....	13
Figura 8. Archivo form.py (formulario).	14
Figura 9. Archivo views.py (vista).....	15
Figura 10. Archivo admin.py (admin).	15
Figura 11. Archivo tests.py (tests).	16
Figura 12. Archivo urls.py (urls).....	16
Figura 13. Archivo wsgi.py (wsgi).	17
Figura 14. Archivo settings.py (settings).	17
Figura 15. Archivo manage.py (manage).....	18
Figura 16. Archivo init.py (init).	18
Figura 17. Carpeta static.	19
Figura 18. Carpeta media.	19
Figura 19. Carpeta template.	20
Figura 20. Descargar node.js.	20
Figura 21. Descargar Visual Studio Code.....	21
Figura 22. Creación de aplicación.....	21
Figura 23. Selección de la Actividad.	22
Figura 24. Selección de la Actividad.	22
Figura 25. Aplicación abierta en Android Studio.....	23
Figura 26. Configuración del host y las aplicaciones en el archivo settings.py.....	29
Figura 27. Configuración del template y base de datos en el archivo settings.py.....	30
Figura 28. Configuración para el registro con Gmail y Github en el archivo settings.py.	31
Figura 29. Administración para registro de usuarios desde la cuenta de Gmail.....	32
Figura 30. Configuración del URL en Gmail.....	33
Figura 31. Administración para registro de usuarios desde la cuenta de GitHub.....	34
Figura 32. Configuración del URL en GitHub.....	35
Figura 33. Librerías necesarias para la aplicación móvil.	36
Figura 34. Configuración de Tensor Flow en la aplicación móvil.	36
Figura 35. Consumir Web Service en aplicación móvil.	37
Figura 36. Arrancado del Sistema en una máquina Ubuntu.	37

1. Introducción

Este presente manual técnico ha sido desarrollado con la finalidad de presentar la plataforma desde un punto de vista técnico, familiarizando al personal encargado en las actividades de mantenimiento, revisión, solución de problemas, instalación y configuración de este.

La parte Web es la encargada de la administración, el mismo que contiene un módulo de registro de usuarios y un módulo de registro de plantas, los dos cuentan con métodos de insertar, eliminar, actualizar y listar; tiene también un módulo de reconocimiento de plantas mediante una imagen, incorpora juegos con relación a la información de las plantas y un buscador de plantas el cual nos permite acudir mucho más rápido a la información deseada.

La parte Móvil mediante Web Service, la aplicación consume los datos guardados en la base de datos previamente gestionados por la parte de administración, por otra parte, cuenta con un buscador el cual nos permite acudir mucho más rápido a la información deseada, además tiene incorporado un módulo de visión artificial utilizado para el reconocimiento de plantas en tiempo real.

Este documento facilita las tareas de instalación y servicio de esta plataforma, dándole a los responsables del sistema las herramientas necesarias que le permitan cumplir de manera eficiente las tareas de configuración de la plataforma.

1.1. *Propósito*

El propósito general de este documento es dar a conocer a la o las personas que lean la forma en que se desarrolló esta plataforma, recalcando cada una de las partes fundamentales que contiene y su funcionamiento.

1.2. *Alcance*

Esta plataforma se lo va a implementar en el servidor de la Catedra Unesco par que ahí se pueda utilizar para cualquier institución.

2. Manual de Ejecución

2.1. *Requerimientos*

2.1.1. *Hardware*

Estos requerimientos son básicos que se tiene en un computador personal.

- Procesador Intel Core i7 Generación.
- Memoria RAM: Mínimo 4 GB
- Disco Duro: Mínimo 80 GB
- Dispositivo Móvil con sistema operativo Android superior a 8.1 Oreo
- Servicio de Internet

2.1.2. *Software*

Estos requerimientos son esenciales para el funcionamiento correcto de la plataforma y son:

Tabla 1. Listado de Software utilizado con sus versiones correspondientes.

Herramientas	Versión	Lugar de Descarga
Windows	10	https://www.microsoft.com/es-es/software-download/windows10
Python	3.6.0	https://www.python.org/downloads/
Django	2.1.0	https://www.djangoproject.com/download/
PostgreSQL	10.4	https://www.postgresql.org/download/
Pip	19.02	https://pypi.org/project/pip/
Virtualenv	16.4.0	https://pypi.org/project/virtualenv/
social-auth-app-django	3.1.0	https://pypi.org/project/social-auth-app-django/
django-select-multiple-field	0.4.2	https://pypi.org/project/django-select-multiple-field/
django-smart-selects	1.5.4	https://pypi.org/project/django-smart-selects/
psycopg2	2.7.7	https://pypi.org/project/psycopg2/
psycopg2-binary	2.7.7	https://pypi.org/project/psycopg2-binary/
Pillow	6.1.0	https://pypi.org/project/Pillow/
Djangorestframework	3.10.12	https://www.django-rest-framework.org/
django-chartjs	1.5.0	https://pypi.org/project/django-chartjs/
django-ratelimit	2.0.0	https://django-ratelimit.readthedocs.io/en/stable/
Reportlab	3.5.23	https://www.reportlab.com/
Tensor Flow	1.12.0	https://www.tensorflow.org/install/pip

Sistema Operativo:

- Windows: Se eligió este sistema operativo ya que es un sistema operativo más conocido, amigable, se integra perfectamente con Python y lo fundamental, se puede utilizar el software de Android Studio con mayor facilidad, sin tanto inconveniente y hay gran cantidad de información de este software para este sistema (Windows 10).

Lenguaje de Programación:

- Python: Python es poderoso y rápido; puede funcionar en cualquier sistema operativo; es amigable, fácil de aprender y es abierto [2].

Framework:

- Django: es un marco web de Python de alto nivel que fomenta el desarrollo rápido y el diseño limpio y pragmático. Creado por desarrolladores experimentados, se encarga de gran

parte de las complicaciones del desarrollo web, por lo que puede concentrarse en escribir su aplicación sin necesidad de reinventar la rueda. Es gratis y de código abierto [1].

Base de datos:

- PostgreSQL: es un potente sistema de base de datos relacional de objetos de código abierto con más de 30 años de desarrollo activo que le ha ganado una sólida reputación de confiabilidad, solidez de características y rendimiento [3].

Paquetes Adicionales:

- Pip: pip es el instalador de paquetes para Python. Puede usar pip para instalar paquetes desde el Índice de Paquetes de Python y otros índices [4].
- social-auth-app-django: Python Social Auth es un mecanismo de autenticación / registro social fácil de configurar con soporte para varios marcos y proveedores de autenticación [5].
- django-select-multiple-field: Seleccione múltiples opciones en un solo campo de modelo de Django. Utilícela siempre que desee almacenar varias opciones en un campo sin utilizar una relación de muchos a muchos [5].
- django-smart-selects: Este paquete permite filtrar o agrupar rápidamente modelos "encadenados" agregando una clave externa personalizada o un campo de muchos a muchos a sus modelos. Esto utilizará una consulta AJAX para cargar solo los objetos encadenados aplicables [5].
- psycopg2 y psycopg2-binary: Psycopg es el adaptador de base de datos PostgreSQL más popular para el lenguaje de programación Python. Sus características principales son la implementación completa de la especificación Python DB API 2.0 y la seguridad de subprocesos (varios subprocesos pueden compartir la misma conexión) [5].
- Pillow: Pillow es el amistoso tenedor PIL por Alex Clark y colaboradores. PIL es la biblioteca de imágenes de Python por Fredrik Lundh y colaboradores [5].
- Django Chartjs le permite administrar gráficos en su aplicación Django. Esto es compatible con las bibliotecas Chart.js y Highcharts JS. Usando un conjunto de vistas basadas en clases predefinidas, puede comenzar después de escribir solo su consulta SQL [6].
- Django Ratelimit proporciona un decorador para limitar las vistas de velocidad. La limitación puede basarse en la dirección IP o en un campo en la solicitud, ya sea una variable GET o POST [7].
- ReportLab tiene más de 20 años de experiencia creando servicios web PDF utilizando una variedad de tecnologías que incluyen JSON y XML. A lo largo del año, hemos desarrollado una forma sencilla de brindar a las empresas un servicio de generación de documentos: crea un paquete de datos en formato JSON y lo publica en una URL web que lo convierte en PDF [8].

2.2. *Diseño de la Plataforma.*

Esta plataforma está estructurada con diferentes capas las mismas que son:

Capa de Presentación.

Formado por página web y aplicación móvil, es la capa encargada de la presentación a los usuarios de la información y módulos implementados que en el sistema.

Página web.

Está enfocado a la administración de los usuarios, información de las plantas medicinales, reconocimiento, juegos así también como reportes de aprendizaje por parte de cada uno de los usuarios que utilicen nuestra plataforma.

Aplicación móvil.

Está enfocado a la búsqueda de información de las plantas medicinales, también cuenta con el módulo de visión artificial para el reconocimiento de plantas en tiempo real.

Capa de Negocio.

Esta capa es la encargada de manejar la lógica del sistema, la forma de gestionar la información y los recursos almacenados para la realización de actividades; además, en esta capa se encuentra el módulo de reconocimiento, que permitirá en base a datos obtenidos, brindar información sobre las plantas que forman parte de la medicina ancestral.

Capa de Datos.

En esta capa se representa el almacenamiento de la información que manejará el sistema, como los diferentes perfiles de los usuarios, así como reportes de los usuarios que utilizan nuestra plataforma. Ver Figura 1.

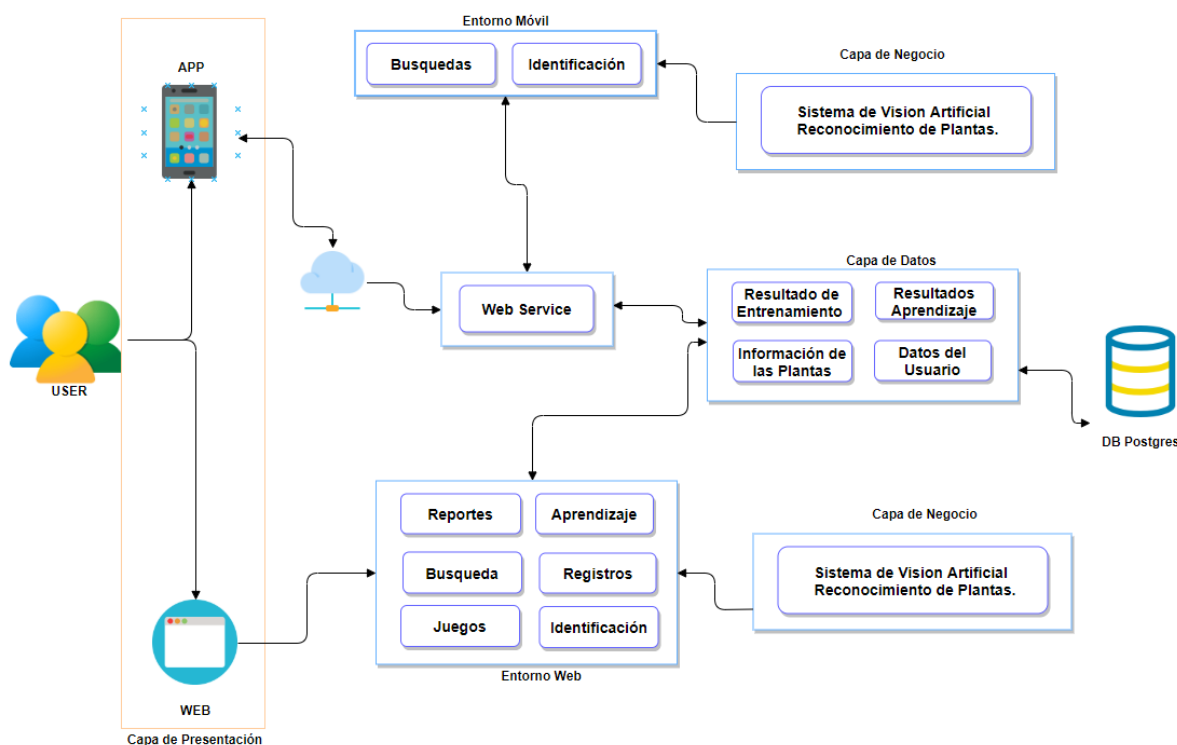


Figura 1. Diagrama de estructura de la plataforma YURAKU (nombre la plataforma).

2.3. Especificación detallada de los Componentes Principales

2.3.1. Especificación de la Base de Datos

La base de datos (DB) escogida para este proyecto fue PostgreSQL, su selección fue por su facilidad de integrarse a cualquier framework o lenguaje de programación en este caso Django (Python), es libre, rápida eficiente para cualquier sistema ya sea pequeños o grandes, por estas razones fue seleccionada para el almacenamiento de toda la información que utilice esta plataforma.

Para este proyecto se generaron las siguientes tablas detalladas a continuación:

1. Las Tablas generadas para el manejo de usuarios, estas se relacionan con: registro, datos de inicio de sesión, permisos y roles, las cuales están manejadas directamente por el módulo de seguridad de Django, las tablas para este proceso son:
 - A. **auth_group**: contiene los roles (en forma de grupos) para los usuarios de la página web.
 - B. **auth_group_permissions**: Contiene la relación de muchos a muchos entre la tabla **auth_group** y **auth_permissions**.
 - C. **auth_permissions**: contiene los permisos que se dará a cada rol para la administración de la página web.
 - D. **auth_users**: Contiene toda la información de los usuarios registrados y/o creados en la página web (nombres, apellidos, usuario (Nick), email, clave, etc.).
 - E. **auth_user_groups**: contiene la relación entre la tabla **auth_group** y **auth_users** que permite dar a los usuarios el rol.
 - F. **auth_user_user_permissions**: Contiene la relación de muchos a muchos entre la **auth_users** y **auth_permissions** es decir cuando un usuario tiene más de un rol (pertenece a más de un grupo).
 - G. **django_admin_log**: Contiene el historial de las acciones que realizara la cuenta admin propia de django.
 - H. **django_content_type**: Contiene todos los modelos que han sido creados para la administración de la página web.
 - I. **django_migrations**: Contiene un historial con las acciones que se realizan con respecto a los usuarios (creación/registro, modificación) mediante una red social y las migraciones que se realicen de la página web.
 - J. **django_session**: Contiene la información de las sesiones que se encuentran activas indicando la fecha de cierre de la sesión.

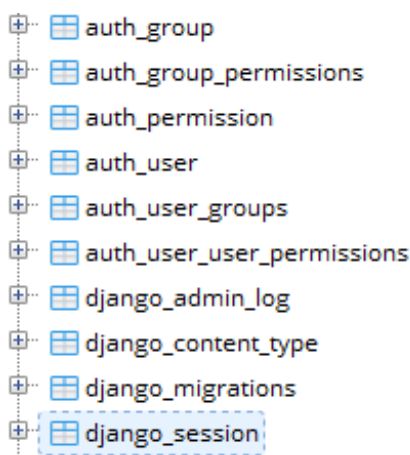


Figura 2. Tablas para el manejo de usuarios de la plataforma.

2. Las tablas generadas para el almacenamiento de la información de la plataforma, basado en el modelo entidad relación de la figura 3.
A continuación, se detallará cada una de ellas:
 - a. **gestionbusqueda_busqueda**: Esta tabla va a contener la información sobre las Búsquedas de diferentes plantas que el usuario realice en la página web. Sus campos son: id, nombre_busqueda, fecha_busqueda y Usuario_id_id, siendo este último una clave foránea que determina la relación con la tabla **auth_user**.

- b. **gestioncomentario_comentario:** Esta tabla va a contener los comentarios a diferentes plantas realizado por parte de los usuarios en la página web. Sus campos son: id, comentario, fecha_comentario, Usuario_id_id siendo este una clave foránea que determina la relación con la tabla **auth_user** y Planta_id_id es una clave foránea que determina la relación con la tabla **gestionplantas_planta**.
- c. **gestionjuego_juego:** Esta tabla va a contener la información sobre las participaciones de los usuarios en los juegos de la página web. Sus campos son: id, nombre_juego, tiempo, fecha_juego, aciertos, intentos y Usuario_id_id, siendo este último una clave foránea que determina la relación con la tabla **auth_user**.
- d. **gestionperfil_perfil:** Esta tabla va a contener la información sobre los perfiles de los usuarios registrados en la página web. Sus campos son: id, nombres_perfil, apellidos_perfil, edad_perfil, grado_perfil, fecha_nacimiento_perfil, imagen_perfil, escuela_perfil y Usuario_id_id, siendo este último una clave foránea que determina la relación con la tabla **auth_user**.
- e. **gestionplantas_planta:** Esta tabla va a contener la información sobre las plantas medicinales para la plataforma. Sus campos son: id, nombre_planta, nombre_común, nombre_científico, reino, lugar_adaptacion, lugar_proviene, uso, como ingerir, dosis, tipo, clase, orden, familia, genero, especie e imagen_planta.
- f. **gestionreconocimiento_reconocimiento:** Esta tabla va a contener la información sobre los reconocimientos de diferentes plantas que los diferentes usuarios realizaran en la página web. Sus campos son: id e imagen_reconocimiento.
- g. **gestionresultado_resultado:** Esta tabla va a contener la información de los resultados que genere el reconocimiento de plantas realizado en la página web. Sus campos son: id, fecha_busqueda, puntuación, mas_resultados, Reconocimiento_id_id es una clave foránea que determina la relación con la tabla **gestionresultado_resultado**, Planta_id_id es una clave foránea que determina la relación con la tabla **gestionplantas_planta** y Usuario_id_id, siendo este último una clave foránea que determina la relación con la tabla **auth_user**.
- h. **Tablas Intermedias (gestionbusqueda_busqueda_Planta):** Estas tablas son intermedias que permiten relacionar de muchos a muchos con la tabla **gestionplantas_planta**.

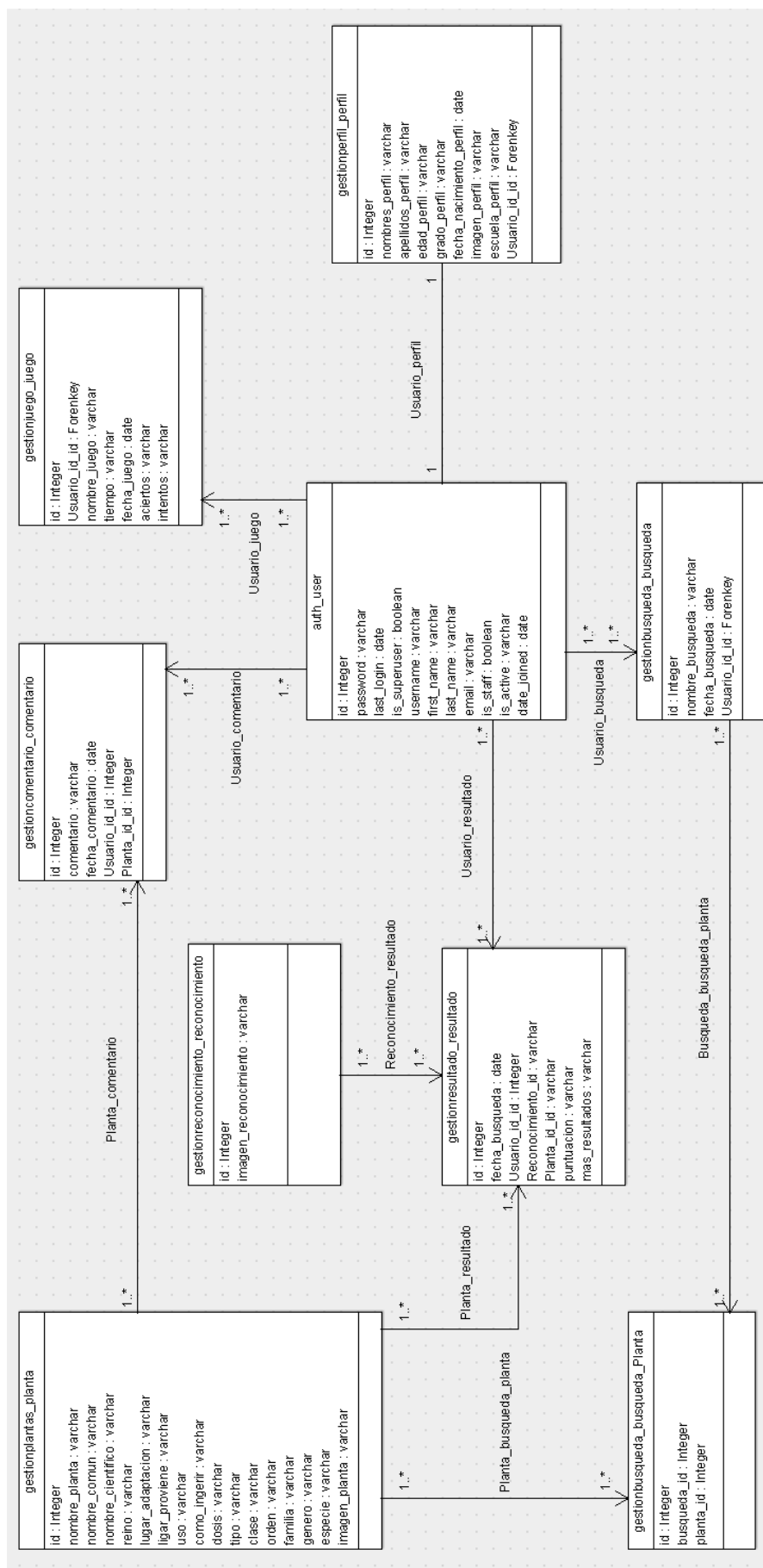


Figura 3. Diagrama entidad-relación de la plataforma.

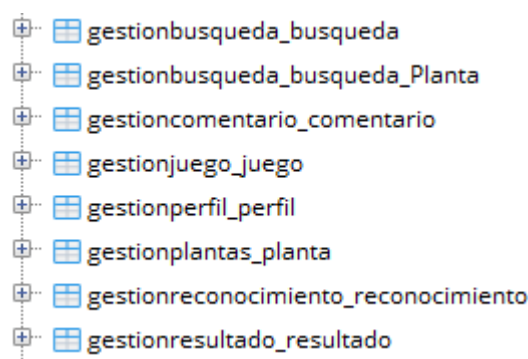


Figura 4. Tablas necesarias para el funcionamiento de la plataforma.

3. Las tablas generadas para el registro y sesión de usuarios con una red social (Gmail y GitHub).

Las tablas para este proceso son:

- **social_auth_association:** Contiene la información de la red social que se usó para registrarse en la página web.
- **social_auth_code:** Contiene el correo de la red social que se utiliza para el registro en la página web.
- **social_auth_nonce:** Contiene la información de la red social (más simple) que se usó para registrarse en la página web.
- **social_auth_partial:** Contiene el token para el registro de la cuenta con el sistema.
- **social_auth_usersocialauth:** Contiene los usuarios (Nick) que toma de la red social y se utiliza para ingresar a la página web.

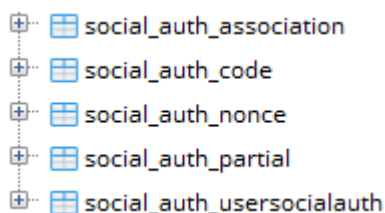


Figura 5. Tablas para el registro y/o sesión de usuarios con una red social en el sistema.

2.3.2. Especificación del sistema de gestión

El sistema de gestión es el encargado de generar toda la estructura de la plataforma, permitiendo que los usuarios finales puedan acceder a la página web y realizar las diferentes actividades que presta dicha página web (búsqueda, reconocimientos, información y juegos en base las plantas medicinales andinas del Azuay).

A continuación, se indicará que representa cada elemento y que produce para el proyecto.

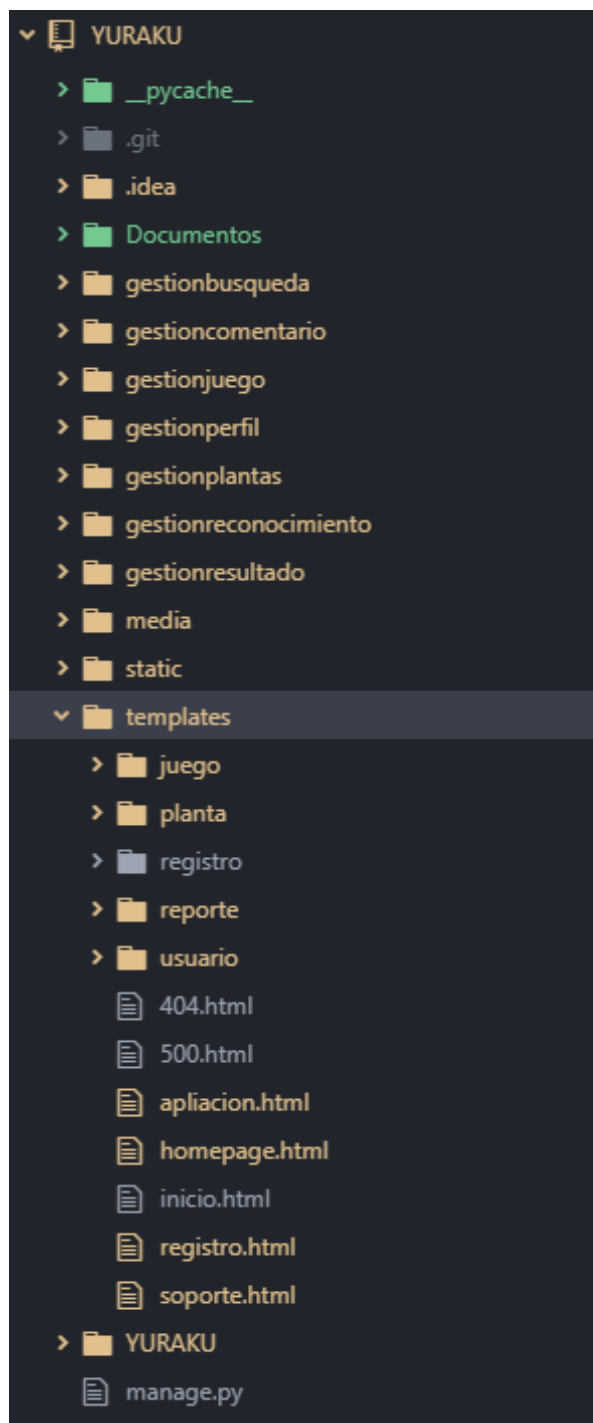


Figura 6. Estructura del sistema generado en Django.

2.3.2.1. Modelos

El modelo o modelos dentro de una aplicación de Django es un archivo escrito en el lenguaje Python llamado “**modes.py**” en el cual se representa la estructura de una tabla de una base de datos, aquí también se coloca las variables de la clase que representan las columnas que tendrá dicha tabla con los respectivos atributos y por último permite realizar las relaciones entre diferentes tablas.

```
from django.db import models

# Create your models here.
class Planta(models.Model):
    #Datos Generales
    nombre_planta = models.CharField(max_length=70)
    nombre_comun = models.CharField(max_length=70)
    nombre_cientifico = models.CharField(max_length=70, null=True, blank=True)
    #Datos Adicionales
    reino = models.CharField(max_length=50, null=True, blank=True)
    lugar_adaptacion = models.CharField(max_length=500, null=True, blank=True)
    lugar_proviene = models.CharField(max_length=500, null=True, blank=True)
    uso = models.CharField(max_length=5000, null=True, blank=True)
    como_ingerir = models.CharField(max_length=1000, null=True, blank=True)
    dosis = models.CharField(max_length=300, null=True, blank=True)
    tipo = models.CharField(max_length=50, null=True, blank=True)
    clase = models.CharField(max_length=50, null=True, blank=True)
    orden = models.CharField(max_length=50, null=True, blank=True)
    familia = models.CharField(max_length=50, null=True, blank=True)
    genero = models.CharField(max_length=50, null=True, blank=True)
    especie = models.CharField(max_length=50, null=True, blank=True)
    imagen_planta = models.ImageField(upload_to="plantas/", null=True, blank=True)

    def __str__(self):
        mensaje = (str(self.nombre_planta) + str(self.nombre_comun) + str(self.nombre_cientifico) + str(self.reino))
        return mensaje
```

Figura 7. Archivo model.py (modelo).

2.3.2.2. Formularios

El formulario o formularios dentro de una aplicación de Django es un archivo escrito en el lenguaje Python llamado “**form.py**” en el cual se define los formularios que se llaman desde una página web. Aquí especificamos los modelos que se utilizarán, los campos de dicho modelo, las etiquetas que se les va a asignar a cada campo para mostrar en la página y los widgets que cada campo requiere como son: el tipo de jQuery que se implementa para el control, el tipo de campo, placeholder, etc.

```

from django import forms
from gestionplantas.models import Planta

class GuardarPlantaForm(forms.ModelForm):
    class Meta:
        model = Planta
        fields = ['nombre_planta',
                  'nombre_comun',
                  'nombre_cientifico',
                  'reino',
                  'lugar_adaptacion',
                  'lugar_proviene',
                  'como_ingerir',
                  'uso',
                  'dosis',
                  'tipo',
                  'clase',
                  'orden',
                  'familia',
                  'genero',
                  'especie',
                  'imagen_planta',]
        labels = {'nombre_planta': 'Nombre de la Planta:',
                  'nombre_comun': 'Nombre Común:',
                  'nombre_cientifico': 'Nombre Científico:',
                  'reino': 'Reino:',
                  'lugar_adaptacion': 'Lugar de Adaptación:',
                  'lugar_proviene': 'Lugar de donde proviene:',
                  'como_ingerir': 'Como ingerir',
                  'uso': 'Uso',
                  'dosis': 'Dosis:',
                  'tipo': 'Tipo:',
                  'clase': 'Clase:',
                  'orden': 'Orden:',
                  'familia': 'Familia:',
                  'genero': 'Genero:',
                  'especie': 'Especie:,'}
        widgets = {'nombre_planta': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Ejemplo: Floripondio'}),
                   'nombre_comun': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Ejemplo: Guanto'}),
                   'nombre_cientifico': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Ejemplo: Es para fines educativos'}),
                   'reino': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Ejemplo: Plantae'}),
                   'lugar_adaptacion': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Ejemplo: Todo el mundo'}),
                   'lugar_proviene': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Ejemplo: Todo el mundo'})

```

Figura 8. Archivo form.py (formulario).

2.3.2.3. Vistas

La vista o vistas dentro de una aplicación de Django es un archivo escrito en el lenguaje Python llamado “**views.py**” en el cual se define métodos que tienen relación directa con una página web. Desde la página web se puede enviar información mediante un “POST” hacia la views para que se almacene, consulte o modifique dicha información, también la página web puede servir información mediante un “GET” hacia la views para que se consulte y seleccione para mostrar en la página al usuario.

Los métodos de la views pueden ser para agregar, listar, modificar, eliminar datos, mostrar datos específicos de alguna consulta, listado de usuarios, modificación de claves o roles, creación y eliminación de usuarios.

Se debe poner atención en los nombres de estos métodos ya que se deberán llamar desde la página web con el mismo nombre que tiene en las views.

```

import subprocess
import threading
import time
from django.contrib.auth.decorators import login_required, permission_required
from django.contrib.auth.models import User
from django.db.models import Q
from django.http import HttpResponse
from django.shortcuts import get_object_or_404, render, redirect
from django.urls import reverse_lazy
from rest_framework import generics
from rest_framework.response import Response
from rest_framework.utils import json
from rest_framework.views import APIView
from gestionbusqueda.models import Busqueda
from gestioncomentario.form import GuardarComentarioForm
from gestioncomentario.models import Comentario
from gestionjuego.models import Juego
from gestionperfil.models import Perfil
from gestionplantas.form import GuardarPlantaForm
from gestionplantas.models import Planta
from gestionreconocimiento.form import ReconocimientoForm
from gestionreconocimiento.models import Reconocimiento
from gestionresultado.models import Resultado
from .serializers import PlantaSerializer
from ratelimit.decorators import ratelimit

# listado de plantas.
def index(request):
    plantas = Planta.objects.all().order_by('id')
    if request.user.id:
        usuario = User.objects.get(id=request.user.id)
        id = usuario.id
        perfiles = Perfil.objects.filter(Usuario_id=id)
        if len(perfiles) > 0:
            perfiles = Perfil.objects.filter(Usuario_id=id)
        else:
            perfiles = []
    else:
        perfiles = Perfil.objects.all()
    return render(request, 'planta/lista_planta.html', {'plantas': plantas, 'perfiles': perfiles})

```

Figura 9. Archivo views.py (vista).

2.3.2.4. Admin

El archivo admin dentro de una aplicación de Django es un archivo escrito en el lenguaje Python llamado “**admins.py**” en el cual se un archivo de configuración para una aplicación compilada de Django llamada Django Admin.

```

from django.contrib import admin
from gestionplantas.models import Planta

""" Registra el modelo Planta en el administrador."""
admin.site.register(Planta)

```

Figura 10. Archivo admin.py (admin).

2.3.2.5. Tests

El archivo tests dentro de una aplicación de Django es un archivo escrito en el lenguaje Python llamado “**tests.py**” en el cual es utilizado para escribir pruebas unitarias para la página web.

```
from django.test import TestCase

# Create your tests here.
```

Figura 11. Archivo tests.py (tests).

2.3.2.6. URLs

El archivo urls dentro de una aplicación de Django es un archivo escrito en el lenguaje Python llamado “**urls.py**” en el cual mapea las rutas y caminos (paths) de nuestro proyecto.

```
from django.contrib import admin
admin.autodiscover()
from django.urls import include, path, re_path
from django.contrib.auth import views as auth_views
from django.contrib.staticfiles.urls import staticfiles_urlpatterns
from django.conf.urls.static import static
import YURAKU.views
import gestionplantas.views
import gestionperfil.views
import gestioncomentario.views
import gestionjuego.views
import gestioncomentario
import gestionplantas.scripts
from django.conf import settings
from django.views.static import settings (UserSettingsHolder) Docs

urlpatterns = [
    path('admin/', admin.site.urls),

    path('admin/doc', include('django.contrib.admindocs.urls')),
    #////////////////////////web service
    #Login
    re_path(r'^weblogin/$', YURAKU.views.weblogin),
    #perfil
    re_path(r'^',include('gestionperfil.urls')),
    #plantas
    re_path(r'^',include('gestionplantas.urls')),
    #busqueda de plantas
    re_path(r'^busquedaweb/$', gestionplantas.views.buscar_plantas_web),

    # inicio y registro
    path('login', YURAKU.views.login_page, name="login"),
    path('loginv', YURAKU.views.login_view, name="loginv"),
    path('logout', YURAKU.views.logout_view, name="logout"),
    path('registro', YURAKU.views.registro, name='registro'),
    re_path(r'^oauth/', include('social_django.urls', namespace="social")),

    # carga de imagenes
    re_path(r'media/(?P<path>.*)$', serve, {'document_root': settings.MEDIA_ROOT}),
    re_path(r'media/(?P<path>.*)$', serve, {'document_root': settings.STATIC_ROOT}),

    # Juegos
    path('juegos', gestionplantas.views.juegos, name='Juego'),
```

Figura 12. Archivo urls.py (urls).

2.3.2.7. Wsgi

El archivo wsgi de una aplicación de Django es un archivo escrito en el lenguaje Python llamado “**vwsgi.py**” en el cual es una interfaz de puerta de enlace utilizada para el despliegue del proyecto.

```

"""Tesis YURAKU de Christian Flores y Franklin Villavicencio 2019"""

"""
La comunidad web Python ha creado un estándar llamado Web Server Gateway Interface,
o por sus siglas, WSGI. Este standar nos permite escribir programas los cuales puedan
comunicarse a través del protocolo HTTP, es decir, Internet.

WSGI de configuracion para proyecto YURAKU .

El WSGI invocado como una variable de nivel de módulo se lo denomina "application".

Para obtener más información sobre este archivo, vea
https://docs.djangoproject.com/en/2.1/howto/deployment/wsgi/
"""

import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'YURAKU.settings')

application = get_wsgi_application()

```

Figura 13. Archivo wsgi.py (wsgi).

2.3.2.8. Settings

El archivo settings dentro de una aplicación de Django es un archivo escrito en el lenguaje Python llamado “**settings.py**” en el cual contiene la configuración de todo el proyecto (base de datos, aplicaciones que utiliza Django, etc.).

```

import os

from django.urls import reverse_lazy

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'w4u0_l3(bnkyw9-pdj$6561zj&wod+a@#mf05&4$fb^fyd-c-!'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ['localhost', '127.0.0.1', '192.168.1.128', '192.168.137.160', '172.16.218.64', '192.168.1.11', '0.0.0.0']
#ALLOWED_HOSTS = ['franklin.pythonanywhere.com']

# Application definition

```

Figura 14. Archivo settings.py (settings).

2.3.2.9. Manage

El archivo manage dentro de una aplicación de Django es un archivo escrito en el lenguaje Python llamado “**manage.py**” en el cual es usado para ejecutar comandos de administración relacionado con el proyecto (pruebas del proyecto, creación de aplicaciones, creación de usuarios, migraciones, etc.)

```
#!/usr/bin/env python
import os
import sys

if __name__ == '__main__':
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'YURAKU.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)
```

Figura 15. Archivo manage.py (manage).

2.3.2.10. Init

El archivo init dentro de una aplicación de Django es un archivo escrito en el lenguaje Python llamado “**init.py**” en el cual indica a Python que esta carpeta es un paquete.

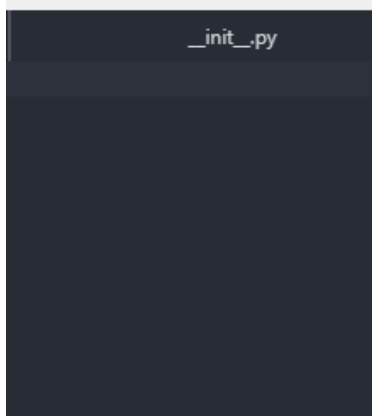


Figura 16. Archivo init.py (init).

Static

La carpeta llamada static contiene todos los archivos (jQuery, imágenes, archivos css) que requiere las páginas web del proyecto.

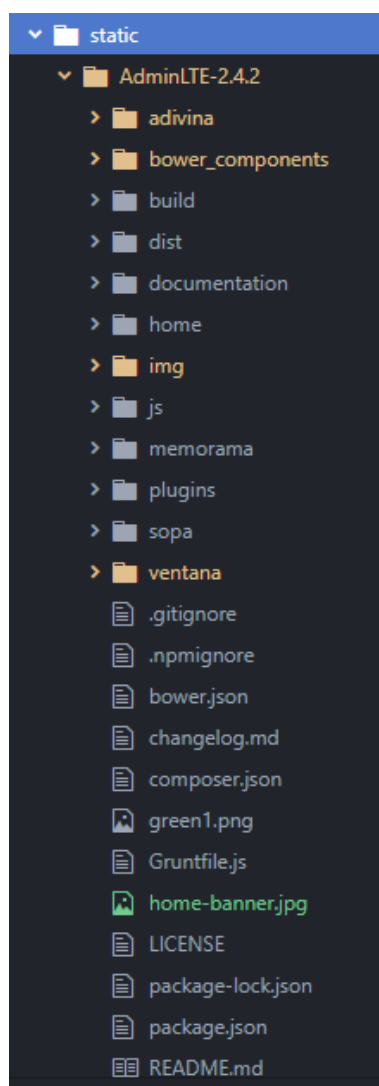


Figura 17. Carpeta static.

Media

La carpeta media contendrá las imágenes y archivos guardados desde la página web en el sistema (recordar que la base de datos almacena el nombre del archivo que está guardado en esta carpeta).

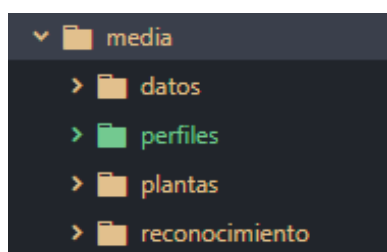


Figura 18. Carpeta media.

Templates

Los Templates o plantillas, son páginas web que el usuario final podrá ver e interactuar. Este template es necesario para cargar los archivos que tienen estilos para las páginas que se utilicen en la página web.

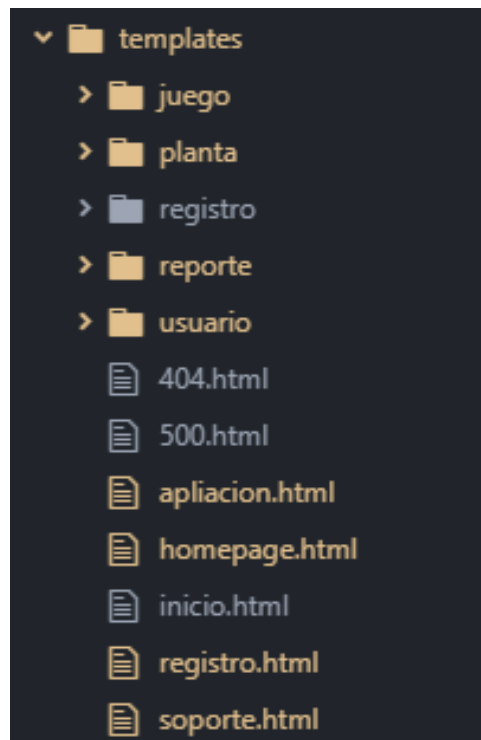



Figura 19. Carpeta template.

2.3.3. Especificación de la aplicación móvil.

La aplicación móvil fue desarrollada en Android Studio, que es un lenguaje de programación basado en Java, mismo que permite crear aplicaciones nativas para dispositivos móviles Android, para la instalación de este desarrollador de aplicaciones es necesario instalar Java en su versión 8. Java 8 se lo puede descargar desde el navegador.

Version 8 Update 221

Fecha de versión: 16 de julio de 2019

 **Actualización importante de la licencia de Oracle Java**

La licencia de Oracle Java ha cambiado para las versiones publicadas a partir del 16 de abril de 2019.

El nuevo [acuerdo de licencia de Oracle Technology Network para Oracle Java SE](#) es sustancialmente diferente a las licencias de Oracle Java anteriores. La nueva licencia permite ciertos usos, como el uso personal y de desarrollo, sin coste alguno (aunque podría haber otros usos autorizados en licencias de Oracle Java anteriores que ya no estén disponibles). Revise las condiciones con atención antes de descargar y utilizar este producto. Puede consultar las preguntas frecuentes [aquí](#).

La licencia comercial y el soporte están disponibles con una [suscripción de Java SE](#) de bajo coste.

Oracle también ofrece la última versión de OpenJDK con la [licencia pública general](#) de código abierto en [jdk.java.net](#).

Descarga gratuita de
Java

Figura 20. Descargar node.js.

Posteriormente se necesita el editor para el desarrollo de la aplicación móvil, en este caso nosotros utilizaremos, como lo habíamos indicado Android Studio el cual lo podemos descargar de la página oficial <https://developer.android.com/studio/index.html?hl=es-419>.

[DOWNLOAD ANDROID STUDIO](#)

3.5 for Windows 64-bit (710 MB)

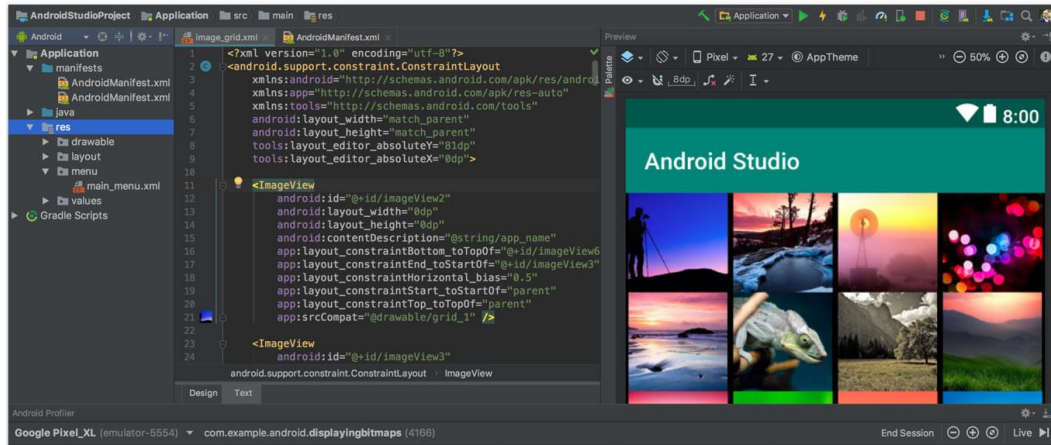
[DOWNLOAD OPTIONS](#)
[RELEASE NOTES](#)


Figura 21. Descargar Visual Studio Code.

2.3.3.1.1. Crear Aplicación.

Una vez instalado el programa tenemos que crear nuestra aplicación para lo cual se aplica los siguientes pasos:

- 1.- Damos clic en File.
- 2.- Luego a New, por ultimo New Project.

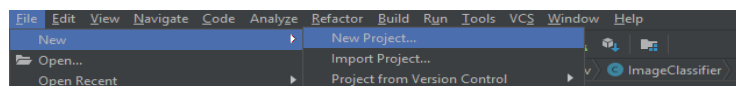


Figura 22. Creación de aplicación.

Posteriormente nos aparece varias opciones mismas que representa a diferentes tipos de diseños de la aplicación como son:

Basic Activity: Genera un Proyecto simple con Title Bar, Float Button, Menu y Back Button.

Empty Activity: Genera un proyecto que contiene solamente el Title Bar y Back Button.

Button Navigation Activity: Genera un proyecto con un Title Bar, Back Button, Menú y Navigation Bar en la parte de abajo donde podemos agregar cualquier cantidad de botones como si se tratasen de accesos directos dentro de nuestra Aplicación

Fullscren Activity: Genera un proyecto que no cuenta con Title Bar ni otros botones, como su nombre lo dice es un Activity a pantalla completa.

Google Maps Activity: Genera un proyecto con un Title Bar, Back Button y un mapa de Google.

Entre otras...

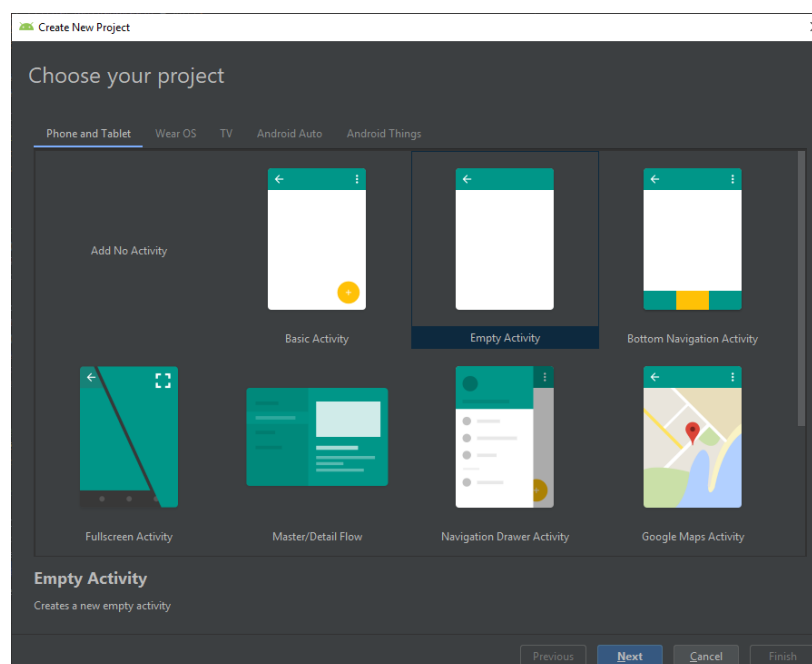


Figura 23. Selección de la Actividad.

3.- Posteriormente procedemos a colocar nuestro nombre de la aplicación, el paquete con el cual nuestra aplicación será identificada posteriormente, el lenguaje en este caso Java y por último se coloca el API mínimo que se necesita para instalar la aplicación, Ver figura 24.

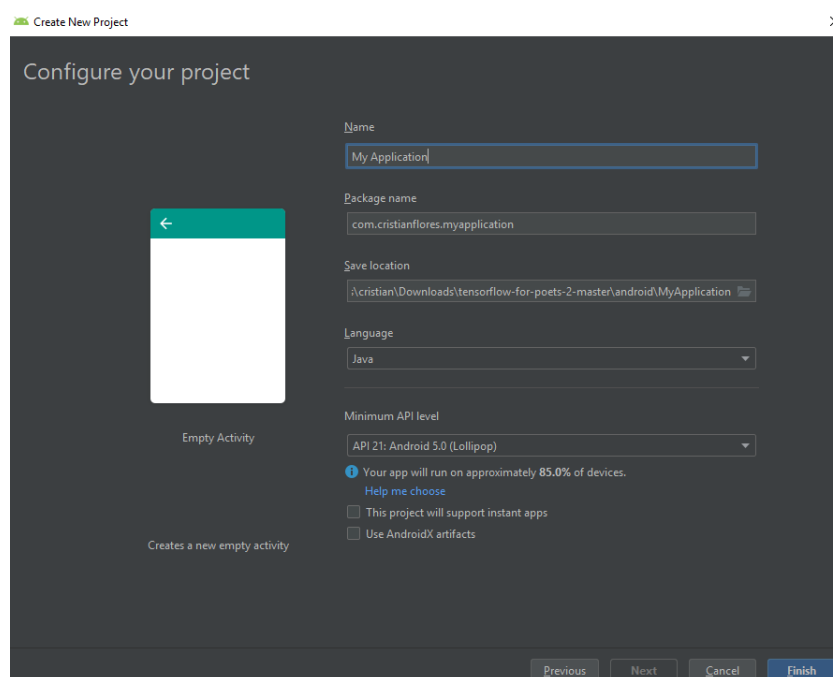


Figura 24. Selección de la Actividad.

4.- Luego de crear el proyecto simplemente con editor en nuestro caso Android Studio procedemos a abrir el proyecto creado, mismo que cuenta con archivos listos para ser modificados, Ver figura 25.

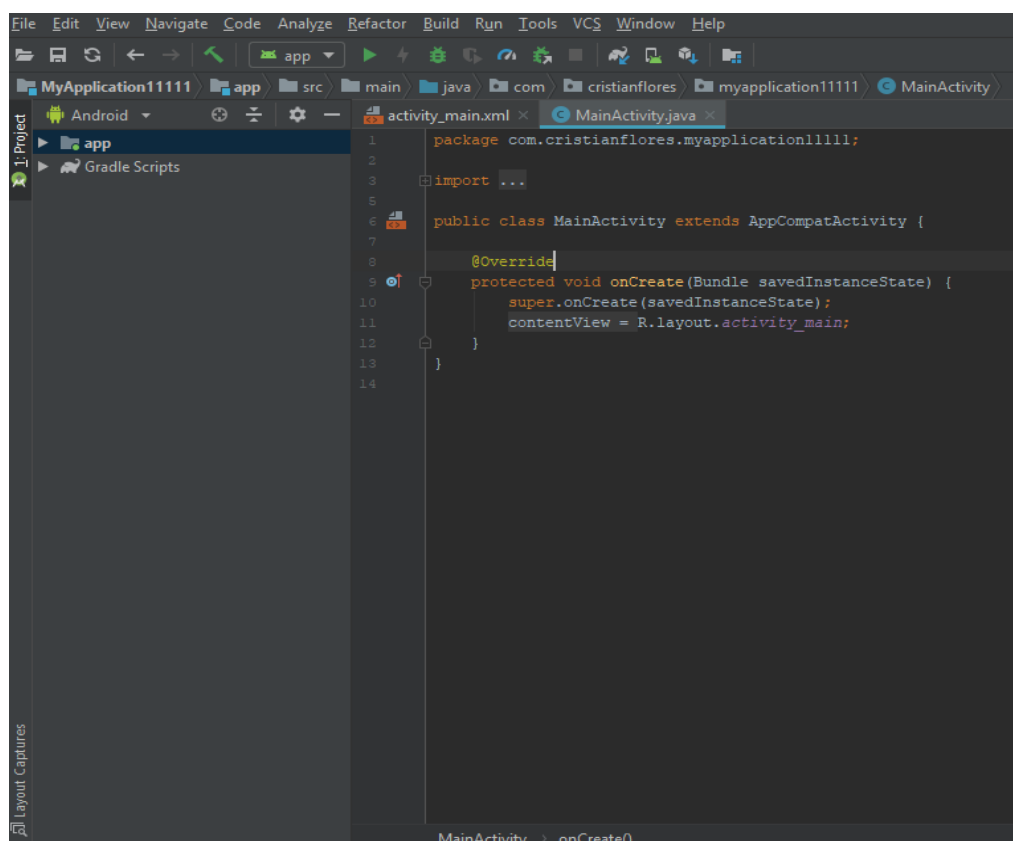


Figura 25. Aplicación abierta en Android Studio.

2.4. Configuraciones del sistema operativo / Instalación de la Base de Datos / Creación de entornos de desarrollo / instalaciones de paquetes.

Para empezar con la configuración del sistema se debe tener creado un usuario en el servidor para mejor manejo tanto de paquetes y archivos sin necesidad de provocar conflictos con otros servicios que se tenga en el mismo servidor.

Creación del Usuario en el Sistema Operativo:

sudo su accedemos como super usuario.

adduser YURAKU agregar el nuevo usuario con ese nombre y cuando pida la clave colocar ***Yuraku2019***.

Instalación de la Base de Datos PostgreSQL:

sudo apt-get install postgresql postgresql-contrib instalar la base de datos PostgreSQL.

Configuración del entorno de desarrollo:

Primer paso es la instalación de python3 y su respectivo gestor de paquetes:

sudo apt-get update siempre se debe actualizar los paquetes para no tener futuros inconvenientes de incompatibilidad.

sudo apt-get install apache2 instalar esta librería para que la máquina pueda tener el servicio de página web.

sudo apt-get install python3.6 instalar esta versión de Python ya que algunas librerías implementadas en este sistema son solo compatibles con esta versión de Python.

sudo apt-get -y install python3-pip También se debe instalar el pip3 para Python hay que con ese podemos descargar las librerías necesarias, Ahora para instalar basta con colocar la palabra pip3 y el nombre del paquete.

Luego crear el entorno virtual para el sistema, en el cual instalaremos todas los paquetes necesarios para su correcto funcionamiento.

sudo pip3 install Virtualenv se debe instalar para crear el entorno de trabajo para que django trabaje sin afectar a ningún otro servicio.

cd /home/usryuraku directorio del usuario.

mkdir py35 crear una carpeta para el entorno.

cd py35 acceder al directorio

Virtualenv -p py35 env comando para crear el entorno.

source py35/bin/activate comando para activar el entorno.

Luego de que ya este creado y activado el entorno virtual se procede a la instalación de los paquetes que son necesarios para el sistema pueda funcionar con normalidad.

pip install Django==2.1.5 Se instala django versión 2.1.5 (la más actual hasta este momento.)

pip install social-auth-app-django instalar esta librería que permitirá registrar usuarios con la información de su red social en el sistema.

pip install django-select-multiple-field instalar este paquete que permite seleccionar muchos elementos en un campo de selección múltiple.

pip install django-smart-selects instalar este paquete para que permite realizar la búsqueda de los elementos en el campo de selección multiple o único.

pip install psycopg2 instalar esta librería porque permite la conexión entre Django y la base de datos PostgreSQL.

pip install psycopg2-binary instalar esta librería porque permite que funcione la librería anterior (***psycopg2***).

pip install Pillow instalar esta librería porque permite el manejo de imágenes en el sistema.

pip install django-rest-framework instalar este paquete que permite seleccionar muchos elementos en un campo de selección multiple.

pip install django-chartjs instalar este paquete para que permite realizar la búsqueda de los elementos en el campo de selección multiple o único.

pip install django-ratelimit instalar esta librería porque permite la conexión entre Django y la base de datos PostgreSQL.

pip install reportlab instalar esta librería porque permite

pip install –upgrade tensorflow instalar esta librería porque permite

Ya con todas las librerías necesarias copiamos la estructura del proyecto dentro de la ruta que deseamos.

cd ruta_del_proyecto Ubicarse en la ruta donde está el proyecto.

cp YURAKU /home/usryuraku comando para copiar la estructura del proyecto al directorio del usuario.

Por último, desactivamos el entorno:

deactivate: comando para desactiva el entorno virtual.

2.4.1. **Creación de BD / Configuración de servidores**

Creación de la base de datos:

Para que el sistema administre la información de los proyectos, se debe generar la base de datos con su respectivo usuario, para lo cual se debe ejecutar los siguientes comandos.

Sudo service postgresql start arrancar el servicio de la base de datos postgresql.

sudo su acceder como super usuario.

sudo su – postgres ingresar con el super usuario de postgresql llamado postgres.

psql acceder a la consola de postgresql.

CREATE USER yuraku PASSWORD 'yuraku2019'; crear el usuario yuraku con su respectiva clase.

ALTER ROLE yuraku WITH SUPERUSER; dar privilegios de super usuario.

CREATE DATABASE yuraku WITH OWNER yuraku; crear la base de datos asociando el nuevo usuario como su creador.

Configuración del Servidor apache:

Para configurar la aplicación debe colocarse en un servidor web apache para lo cual se debe ejecutar los siguientes comandos:

sudo apt-get install apache2 libapache2-mod-wsgi-py3 comando para instalar el servidor apache2 y la librería mod_wsgi para la conexión con django.

/etc/init.d/apache2 start inicia el servidor de Apache.

Cd /etc/apache2/site-availabe ubicarse en esta ruta para configurar los archivos para que le apache sirva el proyecto de Django.

sudo gedit 000-default.conf nombre del archivo a editar y colocamos lo siguiente.

<VirtualHost *:80>#puerto para escuchar las peticiones en apache

<Directory /var/www/yuraku/ YURAKU> #ruta en la que esta alojada la carpeta del proyecto

<Files wsgi.py>#Nombre del archivo wsgi.py con la configuración del proyecto

Require all granted #otorgar todos los privilegios a apache

</Files>

</Directory>

WSGIDaemonProcess myproject python-home=/home/usryuraku/YURAKU/py35 python-path=/home/usryuraku/YURAKU/YURAKU

#python-home indica la ruta del entorno virtual, python path la ruta del proyecto y de los paquetes utilizados

WSGIProcessGroup myproject #nombre asignado al proyecto

WSGIScriptAlias / /var/www/yuraku/YURAKUwsgi.py #formapara llamar al proyecto desde la barra del navegador.

</VirtualHost>

Configuración en el Framework Django:

Para configurar Django, solo tenemos que editar el archivo settings.py ubicado en el directorio con el mismo nombre del proyecto, aquí se modificara los siguientes parámetros:

DEBUG = True indica el modo de producción (Si está en false está en producción). Aquí se debe cambiar por “**False**” para ponerle en producción.

ALLOWED_HOSTS = ['localhost', '127.0.0.1', '192.168.0.194', 'franklin.pythonanywhere.com'] especifica las direcciones IP o los dominós permitidos para que el servidor pueda desplegarse. Aquí se edita si se tiene otra IP o si ya tiene un domino para el sistema.

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': [os.path.join(BASE_DIR, 'templates')],

'APP_DIRS': True,

'OPTIONS': {

'context_processors': [

'django.template.context_processors.debug',

'django.template.context_processors.request',

'django.contrib.auth.context_processors.auth',

'django.contrib.messages.context_processors.messages',

```

    },
  },
},
]

```

Dirección para los templates del Sistema, aquí se debe modificar si se cambia de sistema operativo. Si se quiere colocar una dirección como tal se modifica **BASE_DIR** por la dirección que este alojado los Templates.

```

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'yuraku',
        'USER': 'yuraku',
        'PASSWORD': 'yuraku2019',
        'HOST': '127.0.0.1',
        'PORT': '5432',
    }
}

```

Aquí se modifica si se tiene una base de datos, el usuario, clave o host distinto a la explicada en este manual.

LANGUAGE_CODE = 'es-sp' Permite modificar el idioma del Proyecto (Mensajes del proyecto y no el lenguaje en las páginas web).

TIME_ZONE = 'America/Guayaquil' Permite tomar la hora para el proyecto.

STATIC_URL = '/static/' Nombre del directorio de los archivos estáticos.

```

STATICFILES_DIRS = (
    os.path.join(BASE_DIR, 'static'),
)

```

Indica la ruta que tiene los archivos estáticos.

TEMPLATE_DIRS = (os.path.join(BASE_DIR, 'templates'),) Indica la ruta de los templates.

MEDIA_ROOT = os.path.join(BASE_DIR, 'media') ruta de los archivos multimedia.

MEDIA_URL = '/media/' nombre de la carpeta de los archivos multimedia.

LOGIN_REDIRECT_URL = 'homepage' comportamiento del sistema luego del inicio de sesión.

EMAIL_USE_TLS = True configuración del TLS para el envío de correos.

EMAIL_HOST = 'smtp.gmail.com' especificación del host para el envío del correo.

EMAIL_PORT = 25

#EMAIL_PORT = 587 Puerto por el que se enviara el correo electrónico. Aquí se debe comentar el puerto 25 y habilitar el 587 para la puesta en producción.

EMAIL_HOST_USER = 'yuraku2019@gmail.com' Dirección desde la que se enviara los mensajes.

EMAIL_HOST_PASSWORD= 'Yuraku2019' Clave de la dirección de correo antes mostrada.

#EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'

EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend' El correo electrónico será enviado a través de un servidor SMTP. Aquí se debe habilitar el primero y comentar el según para la puesta en marcha del proyecto.

#SOCIAL_AUTH_GOOGLE_OAUTH2_KEY = '573942488723-6jgjqneq56k25u41ic9nusok89tq7lfa.apps.googleusercontent.com'

#SOCIAL_AUTH_GOOGLE_OAUTH2_SECRET = 'Hm3SCjlnSLsvCh67XXKm22G6'

Key y clave de la cuenta de Google para poder registrarse al sistema con esa red social.

#SOCIAL_AUTH_GITHUB_KEY = 'bff4d54eafc9650b37ee'

#SOCIAL_AUTH_GITHUB_SECRET = 'a0ca5db081e8988381bd61578286144a0d3d135b'

Key y clave de la cuenta de GitHub para poder registrarse al sistema con esa red social.

Quedando de esta forma:

```
import os

from django.urls import reverse_lazy

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'w4u0_l3(bnkyw9-pdj$6561zj&wod+a@#mf05&4$fb^fyd-c-!'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ['localhost', '192.168.1.11', '0.0.0.0']
#ALLOWED_HOSTS = ['franklin.pythonanywhere.com']

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django.contrib.admindocs',
    'social_django',
    'rest_framework',
    'gestionplantas',
    'gestionperfil',
    'gestionbusqueda',
    'chartjs',
    'gestionreconocimiento',
    'gestionresultado',
    'gestioncomentario',
    'gestionjuego',
]
```

Figura 26. Configuración del host y las aplicaciones en el archivo settings.py

```
'social_django.middleware.SocialAuthExceptionMiddleware',
]

ROOT_URLCONF = 'YURAKU.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    ],
]

WSGI_APPLICATION = 'YURAKU.wsgi.application'

# Database
# https://docs.djangoproject.com/en/2.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'YURAKU',
        'USER': 'YURAKU',
        'PASSWORD': 'Yuraku',
        'HOST': '127.0.0.1',
        'PORT': '5432',
        # 'NAME': 'yuraku',
        # 'USER': 'yuraku',
        # 'PASSWORD': 'Yuraku',
        # 'HOST': 'franklin-1276.postgres.pythonanywhere-services.com',
        # 'PORT': '11276',
    }
}
```

Figura 27. Configuración del template y base de datos en el archivo settings.py.

```

LANGUAGE_CODE = 'es-sp'

TIME_ZONE = 'America/Guayaquil'

USE_I18N = True

USE_L10N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.1/howto/static-files/

STATIC_URL = '/static/'
STATICFILES_DIRS = (
    os.path.join(BASE_DIR, 'static'),
)

TEMPLATE_DIRS = (os.path.join(BASE_DIR, 'templates'),)

MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL = '/media/'

EMAIL_USE_TLS = True
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_PORT = 587
EMAIL_HOST_USER = 'villa.chispo@gmail.com'
#EMAIL_HOST_USER = 'yuraku2019@gmail.com'
EMAIL_HOST_PASSWORD = 'Frn14k9542@@'
EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'
#EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
DEFAULT_FROM_EMAIL = EMAIL_HOST_USER

LOGIN_REDIRECT_URL = 'homepage'
SOCIAL_AUTH_GOOGLE_OAUTH2_KEY = '604096279264-9vveool58nvipkh9vq59f3t8nr849fim.apps.googleusercontent.com'
SOCIAL_AUTH_GOOGLE_OAUTH2_SECRET = 'k6IHijbWBU-DwyXApEuSfYsq'

SOCIAL_AUTH_GITHUB_KEY = 'ee0d76d261070bcb8e10'
SOCIAL_AUTH_GITHUB_SECRET = '0f52ff30a45ce825af28bdb8200d3e5b6066c822'

```

Figura 28. Configuración para el registro con Gmail y Github en el archivo settings.py.

También se debe configurar el archivo wsgi:

```
sys.path.append('D:/Tesis/YURAKU/YURAKU')
```

```
#sys.path.append('/var/www/YURAKU/YURAKU')
```

Se debe comentar la primera línea y habilitar la segunda.

Luego se tiene que modificar en las cuentas de Gmail y GitHub para permitir el registro y/o inicio de sesión con dichas cuentas.

Para ingresar se usa las credenciales indicadas en este mismo informe en la sección 2.2.4.

Para Gmail, luego del ingreso se ingresa en el siguiente enlace <https://console.developers.google.com/apis/credentials/oauthclient/604096279264-9vveool58nvipkh9vq59f3t8nr849fim.apps.googleusercontent.com?project=seguimientoproyecto&folder&hl=es&organizationId>

Se presentará la siguiente imagen (Figura 23)

ID de cliente	573942488723-6jgjqneq56k25u41ic9nusok89tq7lfa.apps.googleusercontent.com
Secreto de cliente	Hm3SCJinSLsvCh67XXKm22G6
Fecha de creación	31 ago. 2019 00:17:26

Nombre ?

Restricciones

Ingresa los orígenes JavaScript, los URI de redireccionamiento o ambos [Más información](#)

Los orígenes y dominios de redireccionamiento deben agregarse a la lista de dominios autorizados en la [Configuración de consentimiento de OAuth](#).

Orígenes autorizados de JavaScript

Para su uso en las solicitudes de navegador. Se trata del URI de origen de la aplicación cliente. No puede contener caracteres comodín (https://*.ejemplo.com) ni una ruta (<https://ejemplo.com/subdir>). Si utilizas un puerto no estándar, deberás incluirlo en el URI de origen.



Ingresa el dominio y presiona Intro para agregarlo

URI de redireccionamiento autorizados

Para el uso con las solicitudes de un servidor web. Esta es la ruta en tu aplicación a la que se redirecciona a los usuarios después de autenticarse con Google. La ruta se agregará con el código de autorización para el acceso. Debe tener un protocolo. No puede contener fragmentos de URL o rutas relativas. No puede ser una dirección IP pública.



Ingresa el dominio y presiona Intro para agregarlo

Figura 29. Administración para registro de usuarios desde la cuenta de Gmail

Aquí se debe eliminar la url que está en “URLs de redirección autorizados” y colocar la nueva “http://dominio_del_proyecto/oauth/complete/Google-oauth2/” guardar y ya estaría configurado en Gmail.

ID de cliente	573942488723-6jgjqneq56k25u41ic9nusok89tq7lfa.apps.googleusercontent.com
Secreto de cliente	Hm3SCjinSLsvCh67XXKm22G6
Fecha de creación	31 ago. 2019 00:17:26

Nombre ?

yuraku

Restricciones

Ingresa los orígenes JavaScript, los URI de redireccionamiento o ambos [Más información](#)

Los orígenes y dominios de redireccionamiento deben agregarse a la lista de dominios autorizados en la [Configuración de consentimiento de OAuth](#).

Orígenes autorizados de JavaScript

Para su uso en las solicitudes de navegador. Se trata del URI de origen de la aplicación cliente. No puede contener caracteres comodín (https://*.ejemplo.com) ni una ruta (<https://ejemplo.com/subdir>). Si utilizas un puerto no estándar, deberás incluirlo en el URI de origen.

Eliminar

<http://franklin.pythonanywhere.com><https://www.example.com>

Ingresa el dominio y presiona Intro para agregarlo

Agregar

URI de redireccionamiento autorizados

Para el uso con las solicitudes de un servidor web. Esta es la ruta en tu aplicación a la que se redirecciona a los usuarios después de autenticarse con Google. La ruta se agregará con el código de autorización para el acceso. Debe tener un protocolo. No puede contener fragmentos de URL o rutas relativas. No puede ser una dirección IP pública.

Eliminar

<http://franklin.pythonanywhere.com/oauth/complete/google-oauth2/><https://www.example.com>

Ingresa el dominio y presiona Intro para agregarlo

Agregar

Guardar

Cancelar

Figura 30. Configuración del URL en Gmail.

Para GitHub es algo parecido, accedemos con las credenciales dadas en el inciso 2.2.4, luego damos clic en el siguiente enlace <https://github.com/settings/applications/954755> se presentará una ventana como esta en la siguiente imagen (Figura 7)

YURAKU



yuraku2019 owns this application.

[Transfer ownership](#)

You can list your application in the [GitHub Marketplace](#) so that other users can discover it.

[List this application in the Marketplace](#)

0 users

Client ID

bff4d54eafc9650b37ee

Client Secret

a0ca5db081e8988381bd61578286144a0d3d135b

[Revoke all user tokens](#)[Reset client secret](#)

Application logo

[Upload new logo](#)

You can also drag and drop a picture from your computer.

Badge background color

#ffffff

The hex value of the badge background color.



Application name *

YURAKU

Something users will recognize and trust.

Homepage URL *

<http://franklin.pythonanywhere.com/>

The full URL to your application homepage.

Application description

TESIS

This is displayed to all users of your application.

Authorization callback URL *

<http://franklin.pythonanywhere.com/>Your application's callback URL. Read our [OAuth documentation](#) for more information.[Update application](#)[Delete application](#)

Figura 31. Administración para registro de usuarios desde la cuenta de GitHub.

Aquí se debe modificar dos campos “Homepage URL y AuthorizationcallBack URL”, en las dos se debe borrar la que esta y colocar “[http://domino_del proyecto](http://domino_del_proyecto)” y “[http://domino_del proyecto /oauth/complete/github](http://domino_del_proyecto/oauth/complete/github)” respectivamente.

Application logo

[Upload new logo](#)

You can also drag and drop a picture from your computer.

Badge background color

The hex value of the badge background color.



Application name *

Something users will recognize and trust.

Homepage URL *

The full URL to your application homepage.

Application description

This is displayed to all users of your application.

Authorization callback URL *

Your application's callback URL. Read our [OAuth documentation](#) for more information.

[Update application](#)[Delete application](#)

Figura 32. Configuración del URL en GitHub.

2.4.2. Configuración e instalación de paquetes para aplicación móvil.

Dentro de nuestra aplicación es necesario la importación de algunos paquetes los cuales son indispensables para el correcto funcionamiento de esta.

2.4.2.1. Librerías.

Tensorflow-lite: Se utiliza para el módulo de visión artificial que es el encargado del reconocimiento en tiempo real.

Volley: Librería que nos permite hacer peticiones HTTP para el consumo de los webs services.

Picasso: Nos ayuda para la renderización de imágenes.

Readmoretextview: Se utiliza para la mostrar más o menos información dentro de la visualización de texto.

```
implementation 'com.android.volley:volley:1.1.0'
implementation 'com.squareup.picasso:picasso:2.71828'
testImplementation 'junit:junit:4.12'
androidTestImplementation 'com.android.support.test:runner:1.0.2'
androidTestImplementation 'com.android.support.test.espresso:espresso-core'
implementation 'com.android.support:recyclerview-v7:28.0.0'
implementation 'com.android.support:cardview-v7:28.0.0'
implementation 'org.apache.httpcomponents:httpmime:4.5.6'
implementation 'uk.co.chrisjenx:calligraphy:2.2.0'
implementation 'com.github.bumptech.glide:glide:3.7.0'
implementation 'de.hdodenhof:circleimageview:3.0.0'
implementation 'com.android.support:design:28.0.0'
implementation 'com.android.support:support-annotations:28.0.0'
implementation 'com.android.support:support-v13:28.0.0'
implementation 'org.tensorflow:tensorflow-lite:1.13.1'
implementation files('libs/android-async-http-1.4.9.jar')
implementation files('libs/gson-2.6.2.jar')
implementation files('libs/jackson-all-1.8.10.jar')
implementation files('libs/json-simple-1.1.1.jar')
implementation files('libs/retrofit-2.5.0.jar')

implementation "com.borjabravo:readmoretextview:2.0.1"
```

Figura 33. Librerías necesarias para la aplicación móvil.

2.4.2.2. Configuración de Tensor Flow en la aplicación móvil.

Dentro de nuestro proyecto debemos configurar las siguientes líneas para un correcto uso de la librería, esto nos ayudara al reconocimiento en tiempo real, para esto se declara dos archivos que se encuentra dentro de la carpeta **assert** y luego colocamos la extensión de los archivos.

```
project.ext.ASSET_DIR = projectDir.toString() + '/src/main/assets'

assert file(project.ext.ASSET_DIR + "/optimized_graph_lite").exists()
assert file(project.ext.ASSET_DIR + "/retrained_labels.txt").exists()

aaptOptions {
    noCompress "tflite"
    noCompress "lite"
}
```

Figura 34. Configuración de Tensor Flow en la aplicación móvil.

2.4.2.3. Consumir WS en la aplicación móvil.

Para el consumo de WS en nuestra aplicación es necesario la importación de la librería httpcomponents mismos que contiene dos clases como son HttpClient y HttpResponse para lo que con las conexiones y peticiones hacia el servidor, también se importa la librería de JSON que es la encargada de convertir los datos de los WS en objetos para un mejor manejo y posterior renderización en la aplicación.

```
JSONArray array = new JSONArray(response);
for (int i = 0; i < array.length(); i++) {
    JSONObject planta = array.getJSONObject(i);
    plantaList.add(new Planta(
        planta.getString( name: "imagen_planta"),
        planta.getString( name: "nombre_planta"),
        planta.getString( name: "nombre_comun"),
        planta.getString( name: "uso")
    ));
}

HttpClient httpClient = new DefaultHttpClient();
HttpResponse response = httpClient.execute(new HttpGet(URL));
```

Figura 35. Consumir Web Service en aplicación móvil.

2.4.3. Puesta en marcha del sistema.

Para la puesta en marcha del sistema basta con tener las configuraciones antes mencionadas y aplicar los dos últimos comandos:

source py35/bin/activate comando para activar el entorno.

python manage.py migrate --run-syncdb permitirá la migración de las tablas que tiene el sistema.

python manage.py runserver 0.0.0.0:8080 --insecure permitirá arrancar el sistema por esa IP y por ese puerto (se recuerda que puede correr en cualquier IP que se le coloque ejemplo: **python manage.py runserver 172.16.42.122:8080**)

```
(py35) root@unesco-yuraku: /home/usryuraku/YURAKU# cd YURAKU/
(py35) root@unesco-yuraku: /home/usryuraku/YURAKU# nano settings.py
(py35) root@unesco-yuraku: /home/usryuraku/YURAKU# cd ..
(py35) root@unesco-yuraku: /home/usryuraku/YURAKU# python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).
September 12, 2019 - 18:44:14
Django version 2.1.5, using settings 'YURAKU.settings'
Starting development server at http://172.16.42.122:8080/
Quit the server with CONTROL-C.
```

Figura 36. Arrancado del Sistema en una máquina Ubuntu.

2.5. Cuentas asociadas al sistema

Para crear el proyecto se vio en la necesidad de crear cuentas en dos redes sociales como son GitHub y Gmail necesarias para el registro e inicio de sesión en el sistema.

Las credenciales para entrar a Gmail y GitHub son correo: **"yuraku2019@gmail.com"** y la clave **"Frn14K9542@@"**.

3. Consideraciones / Recomendaciones

3.1. *Consideraciones / Recomendaciones*

Para que el sistema trabaje sin ningún problema se debe tener ejecutado el entorno virtual, ya que en ese entorno estarán instalada las librerías encargadas de hacer que la red neuronal funcione y haga el reconocimiento de plantas; para un mejor manejo del sistema se debe usar el navegador Google Chrome ya que los estilos de las páginas web son mejores y son más notables que en otro navegador.

Para la aplicación móvil se recomienda revisar los archivos **lite** y **lebel**s que son de mucha importancia al momento de hacer el reconocimiento en tiempo real en nuestra aplicación. El apk estará localizado dentro de la siguiente dirección: TesisFV\app\build\outputs\apk\debug. Para el uso se debe crear la cuenta desde la pagina web ya que sin esas credenciales no podrán utilizarla.

Referencias

1. Django. (2019). *Django*. Retrieved from Django: <https://www.djangoproject.com/start/overview/>
2. Python.org. (2019). Welcome to Python.org. [online] Available at: <https://www.python.org/about/>
3. Django project.com. (2019). The Web framework for perfectionists with deadlines | Django. [online] Available at: <https://www.djangoproject.com>
4. PostgreSQL.org. (2019). PostgreSQL: The world's most advanced open source database. [online] Available at: <https://www.postgresql.org>
5. PyPI. (2019). *Pillow*. [online] Available at: <https://pypi.org/project/Pillow/>
6. HUBSCHER, R. (2019). peopledoc/django-chartjs. [online] GitHub. Available at: <https://github.com/peopledoc/django-chartjs> [Accessed 31 Aug. 2019].
7. Socol, J. (2019). jsocol/django-ratelimit. [online] GitHub. Available at: <https://github.com/jsocol/django-ratelimit> [Accessed 31 Aug. 2019].
8. Reportlab.com. (2019). ReportLab - Content to PDF Solutions. [online] Available at: <https://www.reportlab.com/> [Accessed 31 Aug. 2019].