Informe de Laboratorio 05

Tema: Django

Nota

Estudiante	Escuela	Asignatura
Andre Hilacondo Begazo Gonzalo Layme Mamani Andre Delgado Allpan Sebastian Chirinos Negron Gabriel Soto Ccoya @unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: III Código: 20231001

Laboratorio	Tema	Duración
05	Django	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	Del 05 Junio 2023	Al 12 Junio 2023

1. Tarea

- Elabore un primer informe grupal de la aplicacion que desarrollara durante este semestre.
- Utilicen todas las recomendaciones dadas en la aplicación library.

1. Los grupos pueden estar conformado por 1 a 4 integrantes.
2. Solo se presenta un informe grupal.
3. Solo se revisa un repositorio. (El unico que este en el informe grupal).
4. Todos los integrantes del grupo tienen una copia del laboratorio e informe en su repositorio privado.
5. Todos los integrantes deben pertenecer al mismo grupo de laboratorio.
6. El docente preguntara en cualquier momento a un integrante sobre el proyecto, codigo fuente, avance.

2. Equipos, materiales y temas utilizados

- Sistema Operativo (GNU/Linux de preferencia).
- VIM 9.0.
- Git 2.39.2.
- Cuenta en GitHub con el correo institucional.
- Python.
- Entorno virtual.
- Django.

3. URL de Repositorio Github

- URL del Repositorio GitHub principal.
- https://github.com/GonzaloRail/Laboratorio05.git
- URL de los repositorios individuales.

1. **ANDRE HILACONDO BEGAZO**
   - https://github.com/ahilacondo/pweb2-laboratorios
2. **GONZALO LAYME MAMANI**
   - https://github.com/GonzaloRail/myParteLab05PWEB
3. **ANDRE DELGADO ALLPAN**
   - https://github.com/andre98652/my-first-blog
4. **SEBASTIAN CHIRINOS NEGRON**
   - https://github.com/BastleyNait/Blog-en-Django
5. **GABRIEL SOTO CCOYA**
   - https://github.com/GabSoto/JOB-s

4. Resolución del Ejercicio

4.1. Explicación del proyecto

- Nos concentramos en realizar una plataforma en línea donde los empleadores pueden publicar ofertas de trabajo y los candidatos pueden buscar y postular a dichas ofertas.

- Características generales:

1. **Registro de usuarios:** Los empleadores y los candidatos se registran en plataforma utilizando un nombre de usuario y una contraseña.
2. **Autenticación de usuarios:** Los usuarios pueden iniciar sesión y cerrar sesión en la plataforma.
3. **Publicación de ofertas de trabajo:** Los empleadores pueden crear y publicar nuevas ofertas de trabajo especificando un título, una descripción, los requisitos y la ubicación.
4. **Búsqueda de ofertas de trabajo:** Los candidatos pueden buscar ofertas de trabajo por título, ubicación o categoría.
5. **Detalles de las ofertas de trabajo:** Los candidatos pueden ver los detalles de una oferta de trabajo, incluyendo la descripción, los requisitos y la información de contacto del empleador.
6. **Postulación a ofertas de trabajo:** Los candidatos pueden postularse a las ofertas de trabajo a través de la plataforma enviando su currículum y una carta de presentación.

4.2. Diagrama entidad relación

- Lo mencionado anteriormente se explica mejor en el siguiente diagrama.

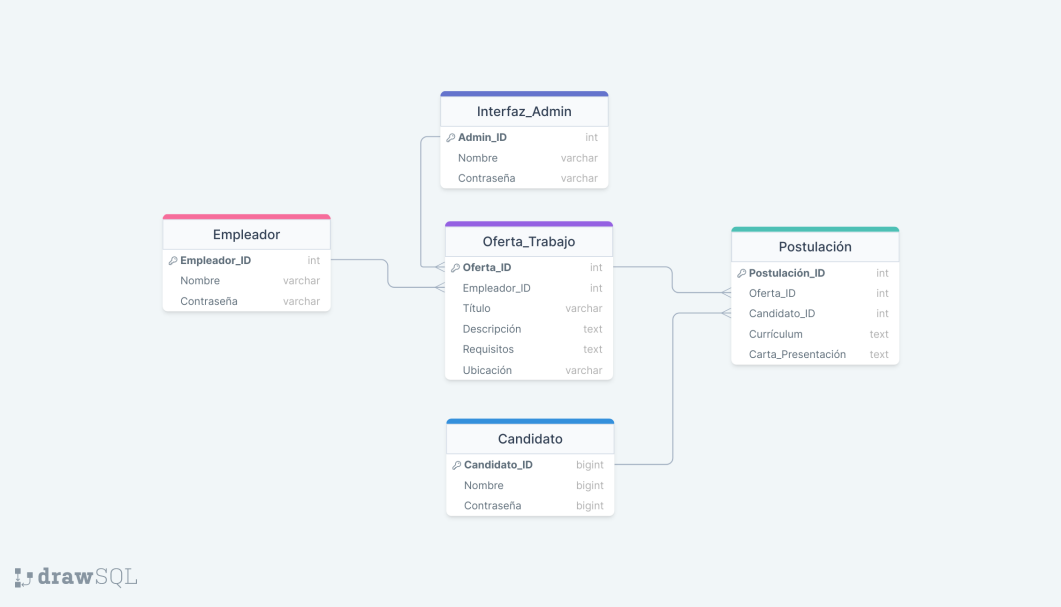

Figura 1: Diagrama entidad relación.

4.3. Desarrollo del proyecto

4.3.1. .gitignore

- Se creo el archivo **.gitignore** para no considerar los archivos **pycache, myvenv, db.sqlite3, etc.** que son innecesarios hacer seguimiento.

Listing 1: Creando .gitignore

```
$ vim lab05/.gitignore
```

Listing 2: lab05/.gitignore

```
*.pyc
*~
__pycache__
myvenv
db.sqlite3
/static
.DS_Store
```

Listing 3: Commit de .gitignore

```
$ git add .
$ git commit -m "Creando .gitignore para lab05"
$ git push -u origin main
```

4.3.2. jobs_board/jobs_board/accounts/admin.py

- El código dado registra el modelo CustomUser en el panel de administracion de Django, lo que permite administrar y manipular los registros de usuario personalizados a través de una interfaz facil de usar.

Listing 4: admin.py

```
from django.contrib import admin

# Register your models here.
from .models import CustomUser

admin.site.register(CustomUser)
```

4.3.3. jobs_board/jobs_board/accounts/apps.py

- Configura el tipo de campo automático predeterminado como 'django.db.models.BigAutoField' y establece el nombre de la aplicación como 'accounts'.

Listing 5: apps.py

```
from django.apps import AppConfig
```

```
class AccountsConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'accounts'
```

4.3.4. jobs_board/jobs_board/accounts/forms.py

- Los formularios necesarios para el inicio de sesión y el registro de usuarios en una aplicación Django.

Listing 6: forms.py

```
from django import forms
from django.contrib.auth.forms import UserCreationForm
from .models import CustomUser

class LoginForm(forms.Form):
   email = forms.EmailField(label='Email')
   password = forms.CharField(label='Contrasea', widget=forms.PasswordInput)

class RegistrationForm(UserCreationForm):
   email = forms.EmailField(label='Email')
   first_name = forms.CharField(label='Nombre')
   last_name = forms.CharField(label='Apellido')

class Meta(UserCreationForm.Meta):
   model = CustomUser
   fields = ('email', 'first_name', 'last_name', 'password1', 'password2')
```

4.3.5. jobs_board/jobs_board/accounts/migrations/0001_initial.py

- El modelo 'CustomUser' incluye campos como 'id', 'password', 'last_login', 'is_superuser', 'email', 'first_name', 'last_name', 'is_active', 'is_staff', 'date_joined', 'groups' y 'user_permissions'. Estos campos representan información relevante para la autenticación, autorización y gestión de usuarios en la aplicación.
- Además de la definición del modelo, el código establece opciones para el mismo, en este caso, se configura 'abstract' como 'False'.

4.3.6. jobs_board/jobs_board/accounts/models.py

- El modelo llamado CustomUser tiene campos como correo electrónico, nombre, apellido, estado de activación y fecha de registro. También incluye un administrador personalizado que proporciona métodos para crear usuarios normales y superusuarios.
- Además, el modelo establece que el correo electrónico se utilizará como campo de inicio de sesión en lugar del nombre de usuario. Tiene relaciones de muchos a muchos con grupos y permisos para asignarlos a los usuarios.

Listing 7: models.py

```
1 from django.db import models
2 from django.contrib.auth.models import AbstractBaseUser, BaseUserManager,
      PermissionsMixin
3
```

```python
class CustomUserManager(BaseUserManager):
    def create_user(self, email, password=None, **extra_fields):
        if not email:
            raise ValueError('El email debe ser proporcionado')
        email = self.normalize_email(email)
        user = self.model(email=email, **extra_fields)
        user.set_password(password)
        user.save()
        return user

    def create_superuser(self, email, password=None, **extra_fields):
        extra_fields.setdefault('is_staff', True)
        extra_fields.setdefault('is_superuser', True)
        return self.create_user(email, password, **extra_fields)

class CustomUser(AbstractBaseUser, PermissionsMixin):
    email = models.EmailField(unique=True)
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
    is_active = models.BooleanField(default=True)
    is_staff = models.BooleanField(default=False)
    date_joined = models.DateTimeField(auto_now_add=True)

    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = ['first_name', 'last_name']

    objects = CustomUserManager()


    groups = models.ManyToManyField(
        'auth.Group',
        verbose_name='groups',
        blank=True,
        help_text='The groups this user belongs to.',
        related_name='customuser_set' # Agrega un related_name nico
    )
    user_permissions = models.ManyToManyField(
        'auth.Permission',
        verbose_name='user permissions',
        blank=True,
        help_text='Specific permissions for this user.',
        related_name='customuser_set' # Agrega un related_name nico
    )




    def __str__(self):
        return self.email
```

4.3.7. jobs_board/jobs_board/accounts/views.py

- login_view maneja el inicio de sesión de los usuarios. Verifica si el formulario de inicio de sesión es válido, autentica al usuario y lo redirige a la lista de trabajos si la autenticación es exitosa.
- register_view se encarga del registro de nuevos usuarios. Guarda los datos del formulario de

registro en la base de datos y redirige al usuario a la página de inicio de sesión.

Listing 8: views.py

```python
from django.shortcuts import render, redirect

# Create your views here.
from .forms import LoginForm, RegistrationForm
from django.contrib.auth import authenticate, login

def login_view(request):
    if request.method == 'POST':
        form = LoginForm(request.POST)
        if form.is_valid():
            email = form.cleaned_data['email']
            password = form.cleaned_data['password']
            user = authenticate(request, email=email, password=password)
            if user is not None:
                login(request, user)
                return redirect('jobs:job_list')
    else:
        form = LoginForm()
    return render(request, 'accounts/login.html', {'form': form})

def register_view(request):
    if request.method == 'POST':
        form = RegistrationForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('accounts:login')
    else:
        form = RegistrationForm()
    return render(request, 'accounts/register.html', {'form': form})
```

4.3.8. jobs_board/jobs_board/jobs_board/urls.py

- La ruta 'admin/' apunta a las vistas proporcionadas por la aplicación de administración de Django.
- La ruta 'accounts/' incluye las URL definidas en el archivo urls.py de la aplicación accounts.
- La ruta '' (ruta vacía) muestra la vista definida en TemplateView con el nombre 'home.html', que representa la página de inicio de la aplicación.
- Además, se importa staticfiles_urlpatterns para configurar correctamente los patrones de URL para servir archivos estáticos en modo de desarrollo.

Listing 9: urls.py

```python
"""
URL configuration for jobs_board project.
The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/4.2/topics/http/urls/
Examples:
Function views
    1. Add an import: from my_app import views
    2. Add a URL to urlpatterns: path('', views.home, name='home')
```

```
Class-based views
    1. Add an import: from other_app.views import Home
    2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path, include
from django.contrib.staticfiles.urls import staticfiles_urlpatterns
from django.views.generic import TemplateView

urlpatterns = [
    path('admin/', admin.site.urls),
    path('accounts/', include('accounts.urls')),
    path('', TemplateView.as_view(template_name='home.html'), name='home'),
]
urlpatterns += staticfiles_urlpatterns()
```

4.3.9. templates/.html

- Incluimos un encabezado, el formulario con los campos de correo electrónico y contraseña, y un botón para enviar los datos del formulario.

Listing 10: login.html

```
{% extends 'base.html' %}

{% block content %}
  <h2>Iniciar sesin</h2>
  <form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Iniciar sesin</button>
  </form>
{% endblock %}
```

- Muy parecida a login, incluimos un encabezado, el formulario con los campos de correo electrónico y contraseña, y un botón para enviar los datos del formulario.

Listing 11: register.html

```
{% extends 'base.html' %}

{% block content %}
  <h2>Registrarse</h2>
  <form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Registrarse</button>
  </form>
{% endblock %}
```

- Estrcutura básica de Django.

Listing 12: base.html

```
{% load static %}
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <link rel="stylesheet" href="{% static 'css/styles.css' %}">
  <title>Jobs</title>
</head>
<body>
  {% block content %}
  {% endblock %}
</body>
</html>
```

- Se extiende de 'base.html' y muestra una página de inicio con dos botones: Ïniciar sesión̈ - egistrarse". Los botones tienen enlaces dinámicos que se determinan utilizando las etiquetas

Listing 13: home.html

```
{% extends 'base.html' %}

{% block content %}
<div class="container">
  <h1>Bienvenido a la pgina de inicio</h1>
  <p>Por favor, selecciona una opcin:</p>
  <div class="button-container">
    <a href="{% url 'accounts:login' %}" class="button">Iniciar sesin</a>
    <a href="{% url 'accounts:register' %}" class="button register">Registrarse</a>

  </div>
</div>
{% endblock %}
```

4.4. Avance del proyecto

- Ahora veremos el aspecto visual de como se empieza a ver el proyecto:

1. **home.html** es lo primero que veran los usuarios al entrar a la página, en versiones futuras habrán más aspectos llamativos.

Figura 2: home

2. **login.html** es una de las opciones que tiene el usuario al ingresar a la pagina.

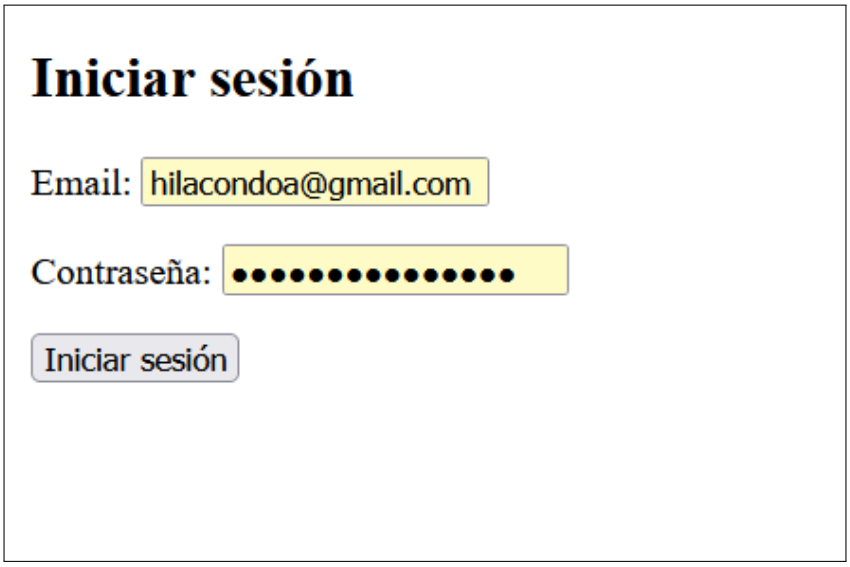

Figura 3: login

3. **register.html** es otra de las opciones que tiene el usuario al ingresar a la pagina.

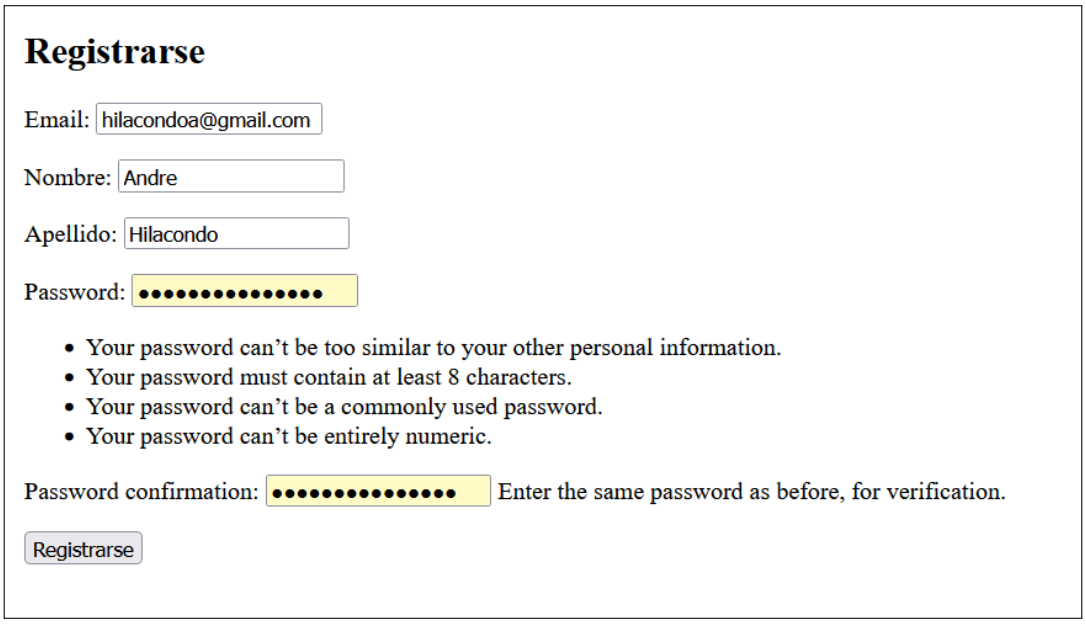

Figura 4: register

4.5. Pregunta

- Por cada integrante del equipo, resalte un aprendizaje que adquiri o al momento de estudiar Django. No se reprima de ser detallista. Coloque su nombre entre parentesis para saber que es su aporte.

  - Implementé el sistema de registro de usuarios, tambien aprendí sobre el manejo de formularios en Django y cómo validar los datos de registro y ya por ultimo tambien aprendi sobre diseño de páginas y cómo crear una interfaz de registro atractiva **(Andre Hilacondo)**.
  - Yo hice en la autenticación de usuarios, aprendí a utilizar el sistema de autenticación de Django y a diseñar la interfaz de inicio de sesión y aunque costo tambien aprendi cómo configurar las páginas de inicio de sesión y cierre de sesión correctamente **(GonzaloLayme)**.
  - Sobre de la publicación de ofertas de trabajo, esa fue mi parte, aprendí a crear un formulario para que los empleadores puedan ingresar los detalles de las ofertas y también exploré cómo guardar los datos en la base de datos y cómo manejar la carga de imágenes relacionadas con las ofertas **(Andre Delgado)**.
  - Me enfoque en la búsqueda de ofertas de trabajo para ello tuve que investigar otras platafromas parecidas y sacar lo mejor, descubrí cómo realizar consultas en la base de datos para filtrar y mostrar las ofertas de acuerdo a los criterios de búsqueda. Además, aprendí a clasificar las ofertas por categoría y a mejorar la eficiencia de las consultas **(Sebastian Chirinos)**.
  - Me encargaron los detalles de las ofertas de trabajo, implementé la vista para mostrar la información detallada de cada oferta y a diseñar una interfaz atractiva para visualizar los detalles, también trabajé en la funcionalidad para mostrar de forma segura la información de contacto del empleador y CSS**(Gabriel Soto)**.

4.6. Commits

Listing 14: Commits del laboratorio

```
commit f621ef2f91afe924f3c828433764cd915f8d0a4b (HEAD -> ramaAndre, origin/ramaAndre)
Author: Andre Delgado <adelgadoal@unsa.edu.pe>
Date:   Mon Jun 19 22:01:40 2023 -0500

    super coommit
commit f621ef2f91afe924f3c828433764cd915f8d0a4b (HEAD -> ramaAndre, origin/ramaAndre)
Author: Andre Delgado <adelgadoal@unsa.edu.pe>
Date:   Mon Jun 19 22:01:40 2023 -0500

    super coommit

commit 66c783c5f6db75232ff466d7616f55731386aa99
Author: Andre Delgado <adelgadoal@unsa.edu.pe>
Date:   Mon Jun 19 22:00:15 2023 -0500

    commit3

commit 923e5c7d650e51cb4b0d5272a440fb36f902d5e6
Author: Andre Delgado <adelgadoal@unsa.edu.pe>
Date:   Mon Jun 19 21:57:05 2023 -0500

    commit

commit 57546aadec1ebcbc57e5ad50681a3b25a57ceb46 (origin/main, origin/HEAD, main)
Merge: c4126b2 af34c6d
Author: ahilacondo <ahilacondo@unsa.edu.pe>
Date:   Mon Jun 19 15:26:40 2023 -0500

    Cambios extras

commit af34c6d7e9f4d480d17d75847dc372b5ec475b92 (ramaHilacondo)
Author: ahilacondo <ahilacondo@unsa.edu.pe>
Date:   Mon Jun 19 15:23:07 2023 -0500

    Cambios extras

commit c4126b2d3b477897f9d1c8c8861acdbd20c36421
Author: ahilacondo <ahilacondo@unsa.edu.pe>
Date:   Mon Jun 19 15:20:43 2023 -0500

    Cambios extras

commit dd73bcc188c5b8db512110f98fb0dbab4552519a
Merge: 1c631e6 63d6000
:...skipping...
commit f621ef2f91afe924f3c828433764cd915f8d0a4b (HEAD -> ramaAndre, origin/ramaAndre)
Author: Andre Delgado <adelgadoal@unsa.edu.pe>
Date:   Mon Jun 19 22:01:40 2023 -0500

    super coommit

commit 66c783c5f6db75232ff466d7616f55731386aa99
```

```
Author: Andre Delgado <adelgadoal@unsa.edu.pe>
Date:   Mon Jun 19 22:00:15 2023 -0500

    commit3

commit 923e5c7d650e51cb4b0d5272a440fb36f902d5e6
Author: Andre Delgado <adelgadoal@unsa.edu.pe>
Date:   Mon Jun 19 21:57:05 2023 -0500

    commit

commit 57546aadec1ebcbc57e5ad50681a3b25a57ceb46 (origin/main, origin/HEAD, main)
Merge: c4126b2 af34c6d
Author: ahilacondo <ahilacondo@unsa.edu.pe>
Date:   Mon Jun 19 15:26:40 2023 -0500

    Cambios extras

commit af34c6d7e9f4d480d17d75847dc372b5ec475b92 (ramaHilacondo)
Author: ahilacondo <ahilacondo@unsa.edu.pe>
Date:   Mon Jun 19 15:23:07 2023 -0500

    Cambios extras

commit c4126b2d3b477897f9d1c8c8861acdbd20c36421
Author: ahilacondo <ahilacondo@unsa.edu.pe>
Date:   Mon Jun 19 15:20:43 2023 -0500

    Cambios extras

commit dd73bcc188c5b8db512110f98fb0dbab4552519a
Merge: 1c631e6 63d6000
Author: ahilacondo <ahilacondo@unsa.edu.pe>
Date:   Mon Jun 19 15:18:52 2023 -0500

    Merge branch 'ramaHilacondo'

commit 63d600027925968de44debafac0984f6cde5a53a (origin/ramaHilacondo)
Author: ahilacondo <ahilacondo@unsa.edu.pe>
Date:   Mon Jun 19 15:15:16 2023 -0500

    Diagrama entidad relacin, y cdigo de conducta

commit 1c631e6557c6ba99cd4cc83a10ba8c0ec5461992
Author: ahilacondo <ahilacondo@unsa.edu.pe>
Date:   Mon Jun 19 13:41:22 2023 -0500

    Cambios en el README.md

commit 41527080c7f0aed6b30395df1744d79962270ef0
Author: ahilacondo <ahilacondo@unsa.edu.pe>
Date:   Mon Jun 19 13:36:13 2023 -0500

    Iniciamos con el proyecto Job_board

commit f915221d0dd389e157814149276959e01e4bf957 (origin/andre-delgado)
```

```
Author: Gonzalo01703 <glaymem@unsa.edu.pe>
Date:   Sun Jun 18 22:35:54 2023 -0500

    .

commit f3cbcf6828cf03f6ed37b21a1e31255c4f6636c4
Author: Gonzalo01703 <glaymem@unsa.edu.pe>
Date:   Sun Jun 18 22:35:01 2023 -0500

    Se creo la plantilla base

commit 968ca7dcd53b518f101f3ca35213e0b292fb318f
Author: Gonzalo01703 <glaymem@unsa.edu.pe>
Date:   Sun Jun 18 22:31:31 2023 -0500

    Se estructura de como se trabajara

commit 35dc86f8f09be84f9c9fcf78049512daa5fbeb53
Author: GonzaloRail <132712920+GonzaloRail@users.noreply.github.com>
Date:   Fri Jun 16 14:53:11 2023 -0500

    Initial commit
```

4.7. Etapas finales del proyecto

- Al proyecto le falta muchos detalles pero la estructura ya está definida al igual que las ideas a futuro, se investigara más sobre la interfaz atractiva y el uso de todo lo aprendido en programación web 2.
- Proyecto (actual):

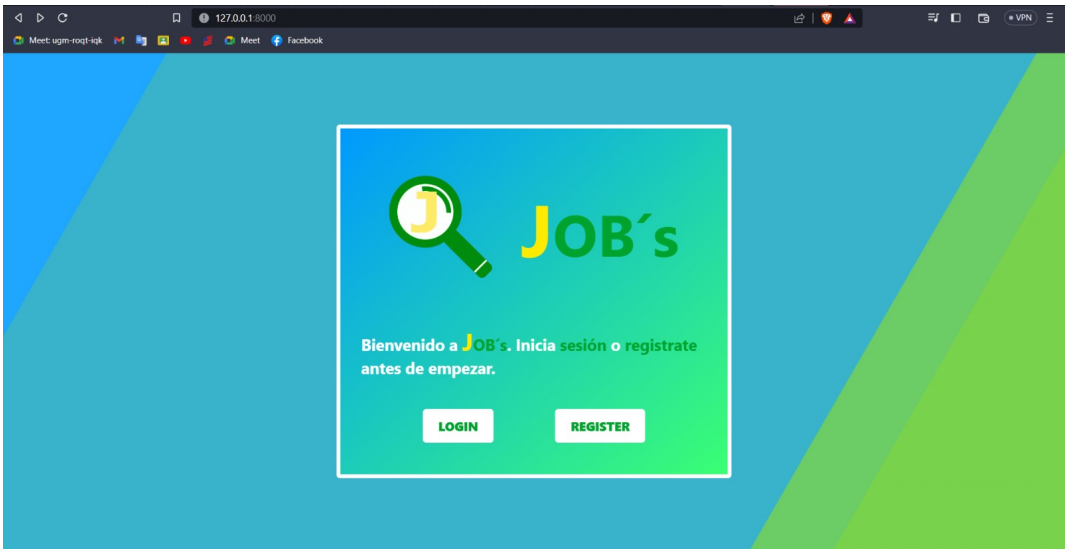

Figura 5: Proyecto "Jobs Board".

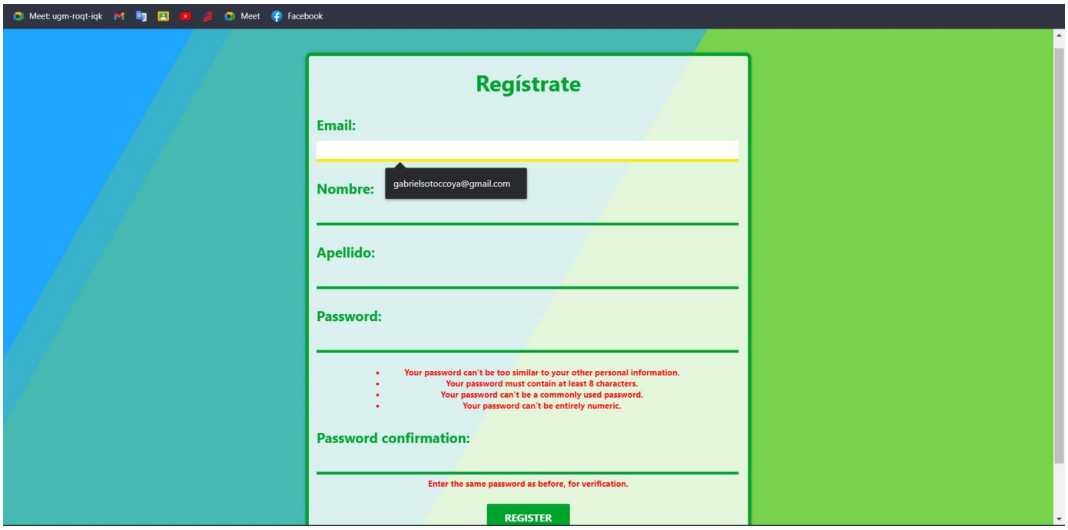

Figura 6: Register de "Jobs Board".

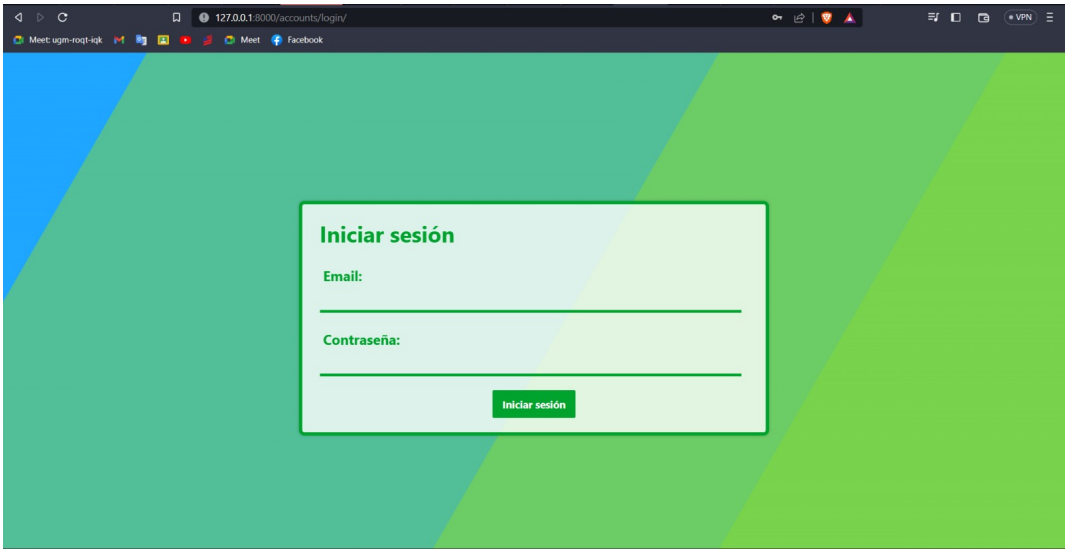

Figura 7: Login de "Jobs Board".

4.8. Estructura de laboratorio 04

- El contenido que se entrega en este laboratorio es el siguiente:

```
job_board/
|----job_board/
     |----__init__.py
     |----settings.py
     |----urls.py
     +----wsgi.py
|----jobs/
     |----__init__.py
     |----admin.py
     |----models.py
     |----forms.py
     |----urls.py
     +----views.py
|----templates/
     +----jobs/
         |----job_list.html
         +----job_detail.html
+----manage.py
```

5. Rúbricas

5.1. Entregable Informe

Tabla 1: Tipo de Informe

Informe	
Latex	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y facil de leer.

5.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumplio con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos lo items.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 2: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

Tabla 3: Rúbrica para contenido del Informe y demostración

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	4	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente estan dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	4	
Total		20		20	

6. Referencias

- `https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Tutorial_local_library_website`
- `https://github.com/mdn/django-locallibrary-tutorial`
- `https://github.com/rescobedoq/pw2/tree/main/labs/lab05`
- `https://realpython.com/`
- William S. Vincent. (2022). Django for Beginners: Build websites with Python. Django 4.0. leanpub.com. URL