

CURSO DE Python 2023

LUIS SEBASTIAN MORONI - COMISIÓN 47775

- Entrega individual

EL PROYECTO FUE DESARROLLADO DE FORMA INDIVIDUAL
POR LUIS SEBASTIAN MORONI.

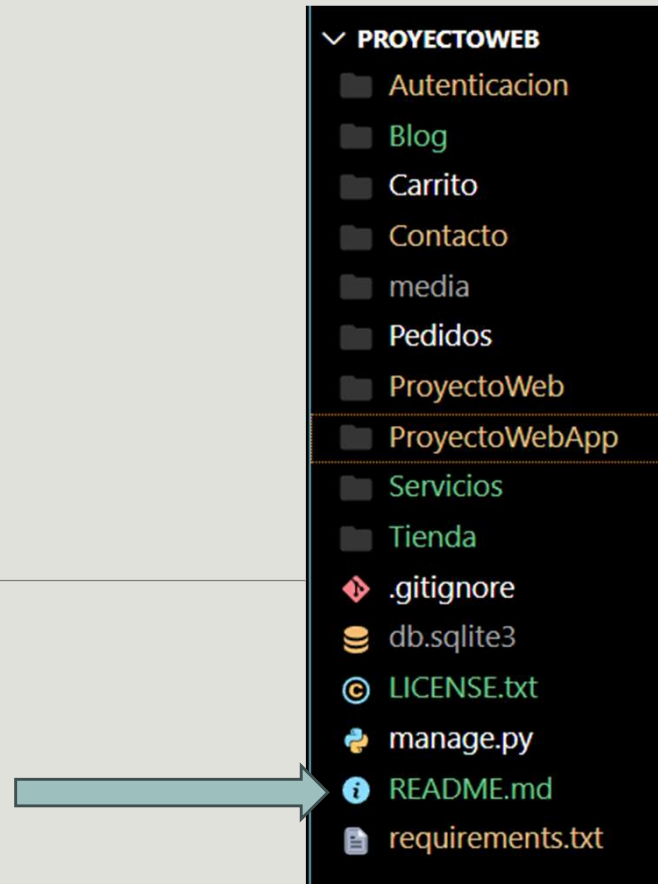
- subir a GitHub

SE ADJUNTO LINK AL REPOSITORIO DE GITHUB:

[HTTPS://GITHUB.COM/SEBA-MORONI/MORONI-SEBASTIAN_PROYECTO_FINAL_CODER_2023.GIT](https://github.com/SEBA-MORONI/MORONI-SEBASTIAN_PROYECTO_FINAL_CODER_2023.GIT)

- readme como la entrega 3

EL ARCHIVO README.MD ES
PARTE DEL PROYECTO



- video de máximo 10 min que muestre la página y sus funcionalidades (con o sin audio)

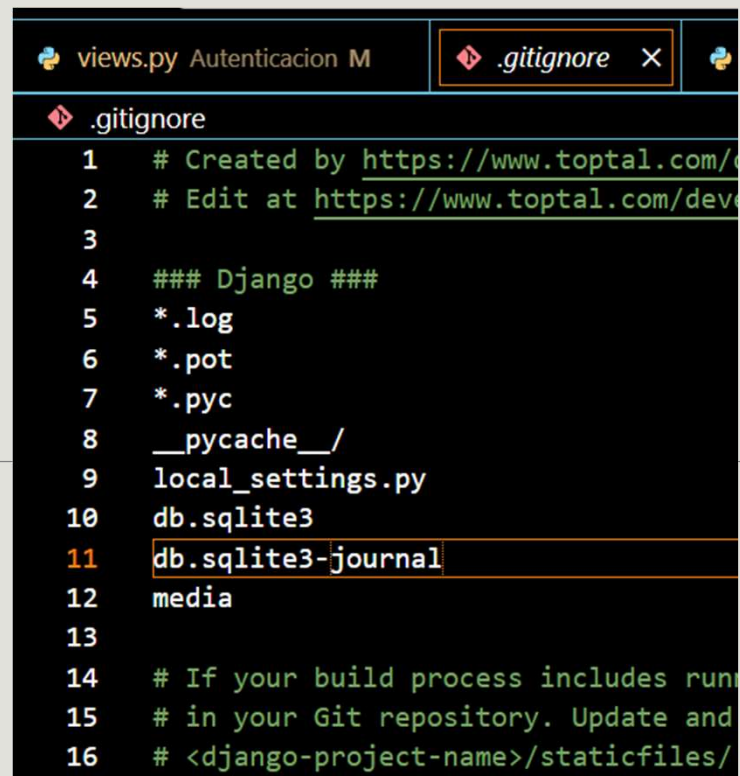
SE ADJUNTA LINK AL CANAL DE YOUTUBE CON LA GRABACIÓN DEL PROYECTO:

PARTE 1: [HTTPS://YOUTU.BE/CQ3SMQIQBGS](https://youtu.be/CQ3SMQIQBGS)

PARTE 2: [HTTPS://YOUTU.BE/CPOCDV7EFIY](https://youtu.be/CPOCDV7EFIY)

- No agregar la Base de datos (el archivo db.sqlite3) en la entrega. Debería estar en el .gitignore

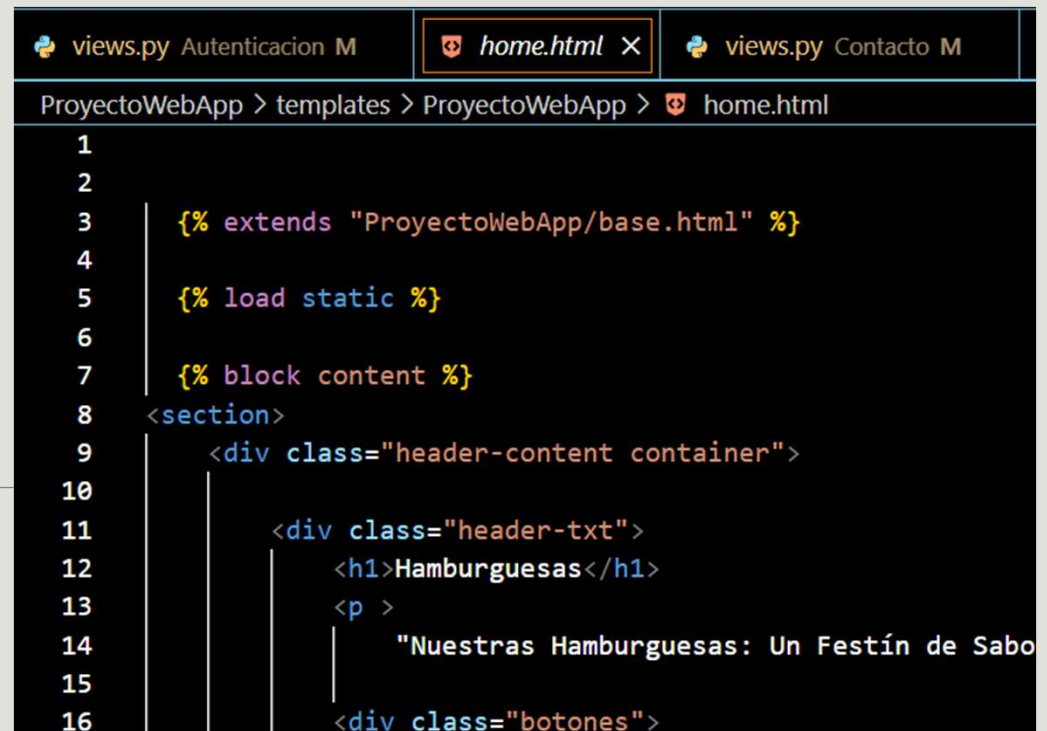
LA BASE ESTA IGNORADA EN
GITIGNORE



```
views.py Autenticacion M .gitignore X
.gitignore
1 # Created by https://www.toptal.com/
2 # Edit at https://www.toptal.com/dev
3
4 ### Django ###
5 *.log
6 *.pot
7 *.pyc
8 __pycache__/
9 local_settings.py
10 db.sqlite3
11 db.sqlite3-journal
12 media
13
14 # If your build process includes run
15 # in your Git repository. Update and
16 # <django-project-name>/staticfiles/
```

- Uso de herencia de templates

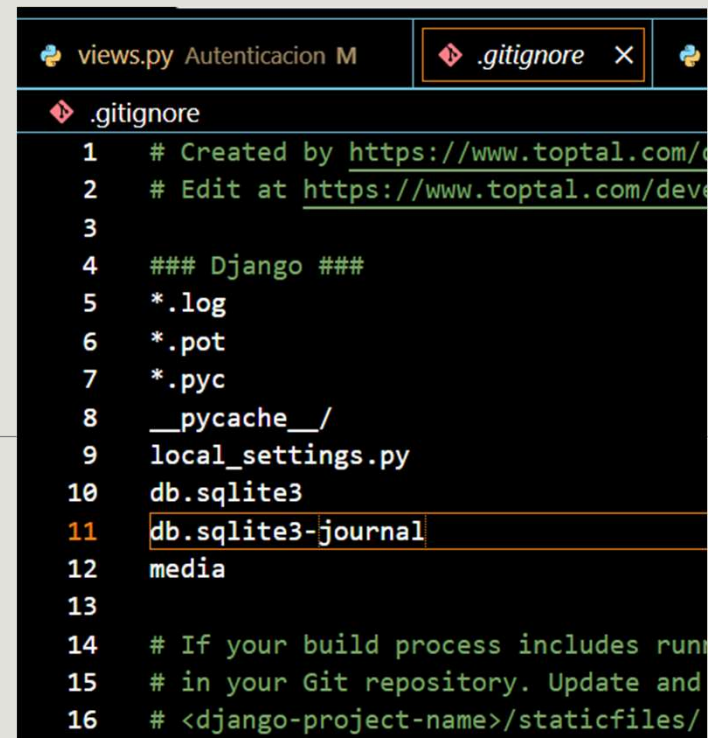
ACÁ HAY UN EJEMPLO DE
HERENCIA DE TEMPLATES



```
1
2
3     {% extends "ProyectoWebApp/base.html" %}
4
5     {% load static %}
6
7     {% block content %}
8 <section>
9     <div class="header-content container">
10
11         <div class="header-txt">
12             <h1>Hamburguesas</h1>
13             <p >
14                 "Nuestras Hamburguesas: Un Festín de Sabo
15
16         <div class="botones">
```

- Exista gitignore con: pycache db.sqlite3 media

ESTA IGNORADO EN GITIGNORE

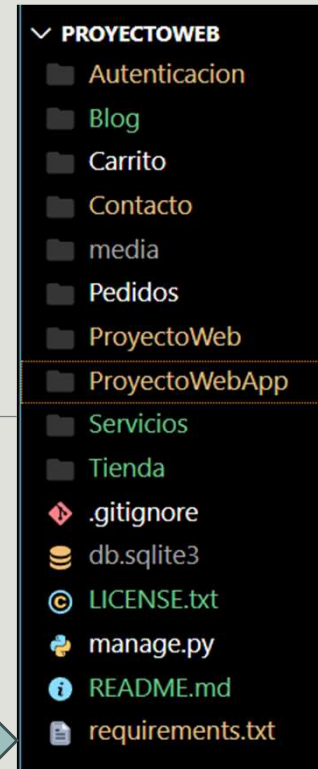


The screenshot shows a code editor window with a tab labeled '.gitignore'. The file content is as follows:

```
1 # Created by https://www.toptal.com/
2 # Edit at https://www.toptal.com/dev
3
4 ### Django ###
5 *.log
6 *.pot
7 *.pyc
8 __pycache__/
9 local_settings.py
10 db.sqlite3
11 db.sqlite3-journal
12 media
13
14 # If your build process includes run
15 # in your Git repository. Update and
16 # <django-project-name>/staticfiles/
```


- Existencia del archivo requirements.txt actualizado.

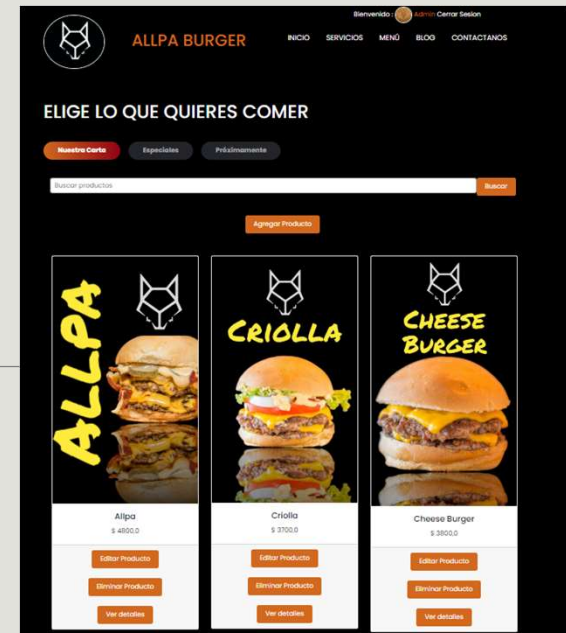
EL ARCHIVO REQUIREMENTS.TXT
ES PARTE DEL PROYECTO



- Tener en cuenta al manejar forms con imagenes hay que adaptar el template, y la vista...no solo el modelo y el formulario

EN EL MODELS.PY DE LA APP “TIENDA” TENEMOS UN EJEMPLO DE MANEJOS DE IMÁGENES, DE IGUAL MANERA TAMBIÉN PODEMOS VER LA GESTIÓN DE IMÁGENES EN EL AVATAR DEL USUARIO.


```
16
17
18 class Producto(models.Model):
19     nombre = models.CharField(max_length=100)
20     descripcion = models.TextField(null=True, blank=True)
21     categorias = models.ForeignKey(CategoriaProd, on_delete=models.CASCADE)
22     imagen = models.ImageField(upload_to='Tienda', null=True, blank=True)
23     moneda = models.CharField(max_length=3, choices=[('USD', 'Dólar estadounidense'), ('EUR', 'Euro')])
24     precio = models.FloatField()
25     disponibilidad = models.BooleanField(default=True)
26     created = models.DateField(auto_now=True)
27     updated = models.DateField(auto_now=True)
28
29 class Meta:
30     verbose_name = 'Producto'
31     verbose_name_plural = 'Productos'
32
33 def __str__(self):
34     return self.nombre
35
```



- Uso de mínimo 2 clases basadas en vista.

SE ADJUNTAN EJEMPLOS DE CLASES BASADAS EN VISTAS. LA IMAGEN DE ARRIBA ESTA EN EL ARCHIVO VIEWS.PY DE LA APP “AUTENTICACIÓN” Y LA IMAGEN DE ABAJO EN EL ARCHIVO URLS.PY DE LA MISMA APLICACIÓN

```
✓ class CambiarContrasenia(PasswordChangeView):  
    template_name = 'Login/cambiar_contrasenia.html'  
    success_url = reverse_lazy('editar_perfil')
```



```
urlpatterns = [  
    path('', VRegistro.as_view(), name="Autenticacion"),  
    path('cerrar_sesion', cerrar_sesion, name="Cerrar_Sesion"),  
    path('logear', logear, name="Logear"),  
    path('logout/', LogoutView.as_view(template_name='cuentas/logout.html'), name='logout'),  
    path('perfil/editar', editar_perfil, name="Editar_Perfil"),  
    path('perfil/editar/contraseña/', CambiarContrasenia.as_view(), name='Cambiar_Contrasena')  
]
```

- Uso de mínimo un mixin en una CBV y un decorador en una view comun.

AMBAS IMÁGENES FORMAN
PARTE DEL ARCHIVO
VIEWS.PY DE LA APP
“AUTENTICACIÓN”

```
52
53 class CustomLoginMixin:
54     def login_user(self, request):
55         login(request, self.user)
56
57 def logear(request):
58     if request.method == "POST":
59         form = AuthenticationForm(request, data=request.POST)
60         if form.is_valid():
61             nombre_usuario = form.cleaned_data.get("username")
62             contrasenia = form.cleaned_data.get("password")
63             usuario = authenticate(username=nombre_usuario, password=contrasenia)
64             if usuario is not None:
65                 login(request, usuario)
66                 mixin = CustomLoginMixin()
67                 mixin.user = usuario
68                 mixin.login_user(request)
69                 return redirect('Home')
70             else:
71                 messages.error(request, "Usuario no Valido")
72         else:
73             messages.error(request, "Informacion Incorrecta")
74         form = AuthenticationForm()
75         return render(request, 'Login/login.html', {'form': form})
76
```

```

@login_required
def editar_perfil(request):
    datos_extra = DatosExtra.objects.get(user=request.user)
    formulario = EdicionPerfil(instance=request.user, initial={'biografia': datos_extra.biografia, 'avatar': datos_extra.avatar})

    if request.method == 'POST':
        formulario = EdicionPerfil(request.POST, request.FILES, instance=request.user)

        if formulario.is_valid():
            nueva_biografia = formulario.cleaned_data.get('biografia')
            nueva_avatar = formulario.cleaned_data.get('avatar')

            if nueva_biografia:
                datos_extra.biografia = nueva_biografia
            if nueva_avatar:
                datos_extra.avatar = nueva_avatar

            datos_extra.save()
            formulario.save()

            return redirect('editar_perfil')

    return render(request, 'Login/editar_perfil.html', {'formulario': formulario})
```

• Una vista de inicio

TENEMOS EN EL APP
“PROYECTOWEBAPP” UN
TEMPLATE CON UNA BARRA
DE NAVEGACIÓN QUE NOS
DIRIGE A LA VISTA INICIO Y
NOSOTROS. ACÁ VEMOS EL
INICIO

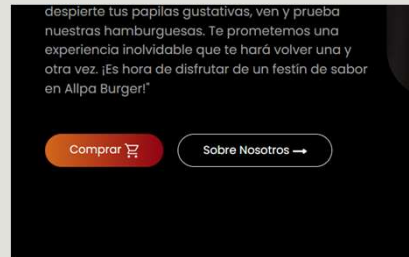
```
App > templates > ProyectoWebApp > base.html
<a href="{% url 'Logear' %}" style="color:white">Login</a>&nbsp;&nbsp;&a href="{% url 'Autenticacion' %}" style="color:wh
{% endif %}
</div>
<div class="menu container">
<a href="#">ALLPA BURGER</h1>
<input type="checkbox" id="menu">
<label for="menu"></label>

<nav class="navbar">
<ul>
<li><a href="{% url 'Home' %}">Inicio</a></li>
<li><a href="{% url 'Servicios' %}">Nosotros</a></li>
<li><a href="{% url 'Tienda' %}">La Carta</a></li>
<li><a href="{% url 'Blog' %}">Blog</a></li>
<li><a href="{% url 'Contacto' %}">Contactanos</a></li>
<!--<li><a style="color: whitesmoke;" href="#" id="abrirModal"><i class="fas fa-shopping-cart"></i></a></li>-->
</ul>
```



- acceso a una vista "Acerca de mi"/"About"

TENEMOS EN AL APP "PROYECTOWEBAPP" UN TEMPLATE CON UNA BARRA DE NAVEGACIÓN QUE NOS DIRIGE A LA VISTA INICIO Y NOSOTROS. ACÁ VEMOS NOSOTROS QUE ES NOMBRE QUE DEFINIMOS PARA REEMPLAZAR A ACERCA DE MI / ABOUT. ADICIONALMENTE SE CREA UNA VISTA QUE SE ACCEDA DESDE EL INICIO DONDE SE EXPLICA MÁS SOBRE EL NEGOCIO, ES DECIR QUE APLICA COMO "ABOUT"



```
app > templates > ProjectWebApp > % base.html
<a href="{% url 'login' %}" style="color:white">Login</a>&nbsp;&nbsp;&nbsp;<a href="{% url 'autenticacion' %}" style="color:white">Autenticacion</a>
{% endif %}
</div>
<div class="menu container">
<a href="#">ALLPA BURGER</h1>
<input type="checkbox" id="menu">
<label for="menu"></label>
<nav class="navbar">
<ul>
<li><a href="{% url 'home' %}">Inicio</a></li>
<li><a href="{% url 'servicios' %}">Servicios</a></li>
<li><a href="{% url 'tienda' %}">La Carta</a></li>
<li><a href="{% url 'blog' %}">Blog</a></li>
<li><a href="{% url 'contacto' %}">Contactanos</a></li>
<li><a href="#">Carrito</a></li>
</ul>
</div>
```



- Crear un modelo principal que contenga los siguientes campos como minimo: 2 Charfield, 1 Campo de texto enriquecido (usando ckeditor), 1 campo de imagen, 1 de fecha

LA IMAGEN DE ARRIBA ES UNA CLASE DEL ARCHIVO MODELS.PY DE LA APP “TIENDA”. POR OTRO LADO, LA DE ABAJO ES DEL ARCHIVO FORMS.PY DE LA APP “CONTACTO”

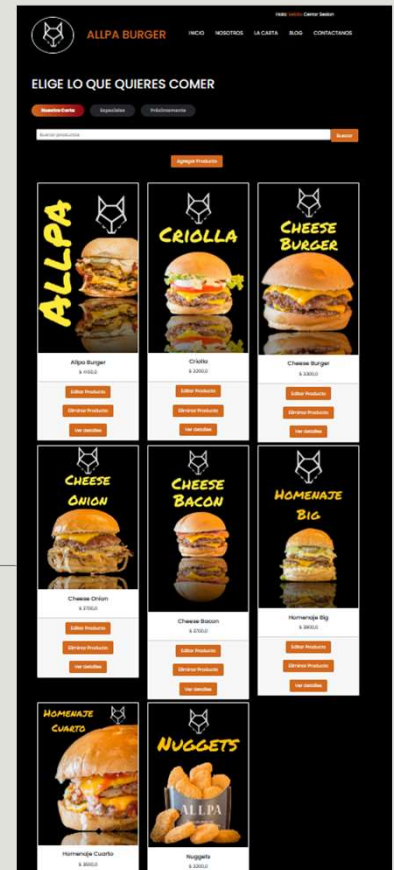
```
17
18 class Producto(models.Model):
19     nombre = models.CharField(max_length=100)
20     descripcion = models.TextField(null=True, blank=True)
21     categorias = models.ForeignKey(CategoriaProd, on_delete=models.CASCADE)
22     imagen = models.ImageField(upload_to='Tienda', null=True, blank=True)
23     moneda = models.CharField(max_length=3, choices=[('USD', 'Dólar estadounidense'), ('EUR', 'Euros')],
24     precio = models.FloatField()
25     disponibilidad = models.BooleanField(default=True)
26     created = models.DateField(auto_now=True)
27     updated = models.DateField(auto_now=True)
28
29     class Meta:
30         verbose_name = 'Producto'
31         verbose_name_plural = 'Productos'
32
33     def __str__(self):
34         return self.nombre
35
```

```
views.py Autenticacion M  base.html M  forms.py M x  views.py Contacto M  urls.py
Contacto > forms.py > ...
1  from django import forms
2  from ckeditor.widgets import CKEditorWidget
3
4  class FormularioContacto(forms.Form):
5      nombre = forms.CharField(label="Nombre", required=True, max_length=30)
6      email = forms.EmailField(label="Email", required=True)
7      contenido = forms.CharField(label="Contenido", widget=CKEditorWidget(), required=True)
8
9
10
```

- Vista de listado de los objetos del modelo principal (modelo a elección). En la cual cada objeto mostrara solo alguno de sus datos

SE COMPARTE LA CLASE PRODUCTO DEL ARCHIVO MODELS.PY DE LA APP “TIENDA” AL IGUAL QUE COMO SE VE EN LA WEB

```
17
18 class Producto(models.Model):
19     nombre = models.CharField(max_length=100)
20     descripcion = models.TextField(null=True, blank=True)
21     categorias = models.ForeignKey(CategoriaProd, on_delete=models.CASCADE)
22     imagen = models.ImageField(upload_to='Tienda', null=True, blank=True)
23     moneda = models.CharField(max_length=3, choices=[('USD', 'Dólar estadounidense')])
24     precio = models.FloatField()
25     disponibilidad = models.BooleanField(default=True)
26     created = models.DateField(auto_now=True)
27     updated = models.DateField(auto_now=True)
28
29     class Meta:
30         verbose_name = 'Producto'
31         verbose_name_plural = 'Productos'
32
33     def __str__(self):
34         return self.nombre
35
```



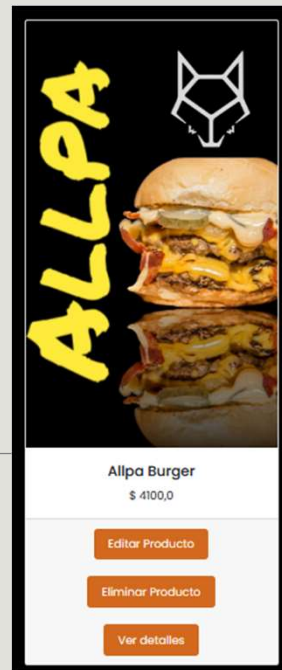
- Mensaje que de aviso en caso de no haber ningún objeto creado o al utilizar el buscador no encontrar tampoco algún objeto

EN EL PROYECTO, TENEMOS VARIOS TIPOS DE MENSAJES, PRO EJEMPLO CUANDO UN USUARIO BUSCA UN PRODUCTO Y ESTE NO EXISTE LE INDICA QUE NO SE ENCUENTRA EL PRODUCTO. POR OTRO LADO, CUANDO NO HAY PRODUCTO EN EL CARRITO, AVISA QUE NO SE INCLUYERON PRODUCTO AL CARRO Y POR ULTIMO CUANDO INGRESAMOS MAL EL USUARIO O CONTRASEÑA INDICA QUE ÑA INFORMACIÓN ES INCORRECTA.



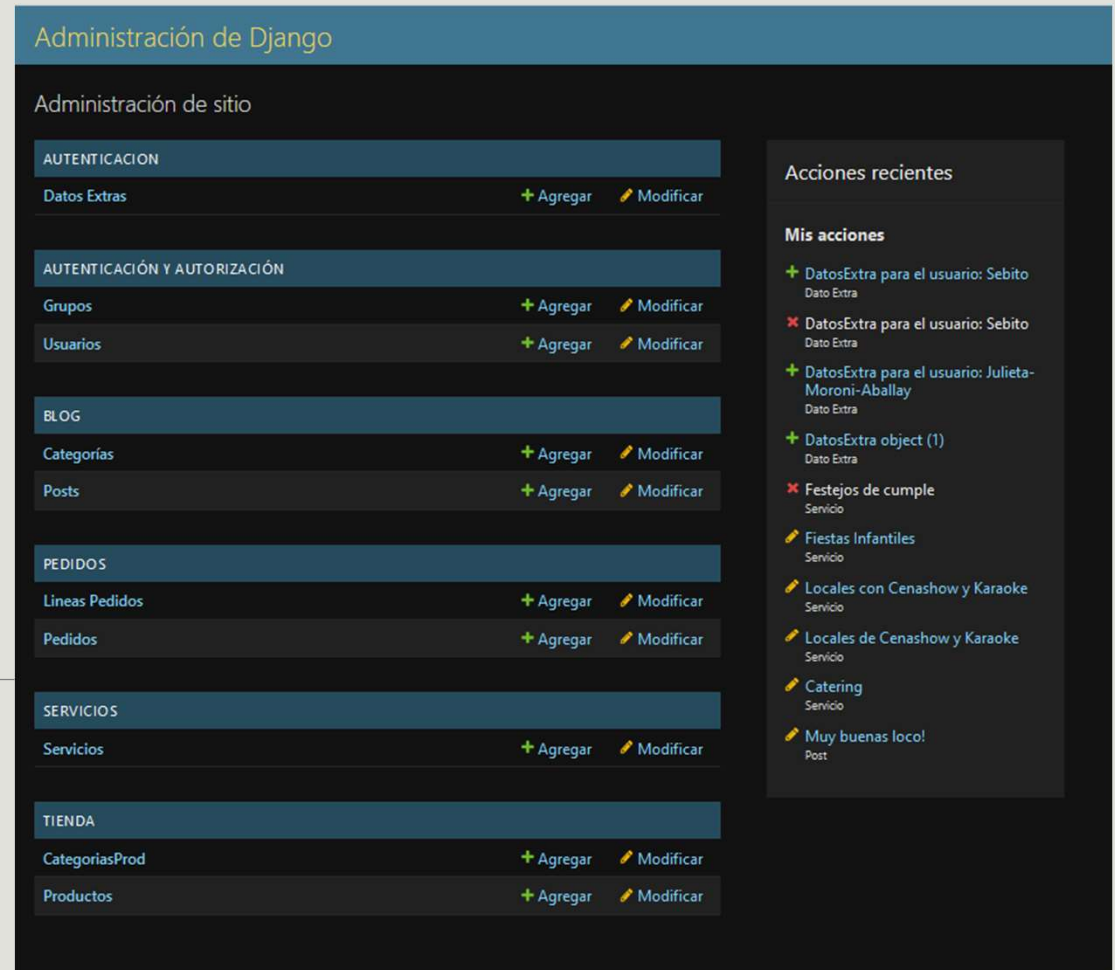
- Desde el listado:
 - i. poder acceder a una vista que muestre el detalle de el objeto seleccionado
 - ii. poder acceder a una vista de creación, una de edición y una de borrado de los objetos del listado

EN LA VISTA DE LOS PRODUCTOS HAY UN BOTÓN VISIBLE SOLO PARA SUPERUSERS PARA: “EDITAR”, “ELIMINAR” Y VER EL DETALLE DE LOS PRODUCTOS. COMO ASÍ TAMBIÉN CREAR NUEVOS PRODUCTO



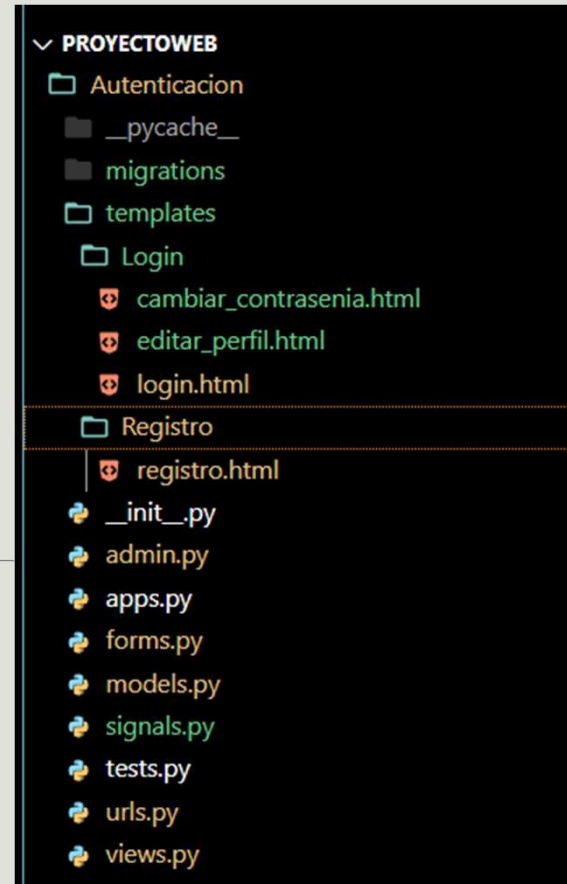
- Registrar en el apartado de admin todos los modelos creados

LOS MODELOS DE CADA APP, ESTÁN REGISTRADOS EN EL ARCHIVO ADMIN.PY DE CADA UNA DE ELLAS. SE ADJUNTA IMAGEN DE LOS MODELOS REGISTRADOS



- Tener una app para el manejo de todas las vistas relacionadas al usuario/authenticación

SE ADJUNTA IMAGEN DE LA APP “AUTENTICACIÓN”, QUE ES DONDE SE GESTIONA TODO LO RELACIONADO CON LOGIN, REGISTRO, ETC.



- Desarrollar las vistas para un login, un logout y el registro de un usuario al cual se le solicite: username, email, password

SE ADJUNTA IMAGEN DE LA VISTA LOGIN, REGISTRO Y LOGOUT.




ALLPA BURGER

Nombre de usuario:

Contraseña:

[Ingresa al sitio](#)
[Registrate](#)



ALLPA BURGER

Nombre de usuario:

Obligatorio. Longitud máxima de 150 caracteres. Solo puede estar formado por letras, números y los caracteres @/./+/_

First name:

Required. 30 characters or fewer.
Last name:

Required. 30 characters or fewer.
Email:

Required. Enter a valid email address.
Contraseña:

- Su contraseña no puede ser similar a otros componentes de su información personal.
- Su contraseña debe contener por lo menos 8 caracteres.
- Su contraseña no puede ser una contraseña usada muy comúnmente.
- Su contraseña no puede estar formada exclusivamente por números.

Confirmación de contraseña:

Introduzca la misma contraseña nuevamente, para poder verificar la misma.

[Registrate](#)



ALLPA BURGER

INICIO NOSOTROS LA CARTA BLOG [CONTACTANOS](#)

Hola: [Sebito](#) [Cerrar Sesión](#)

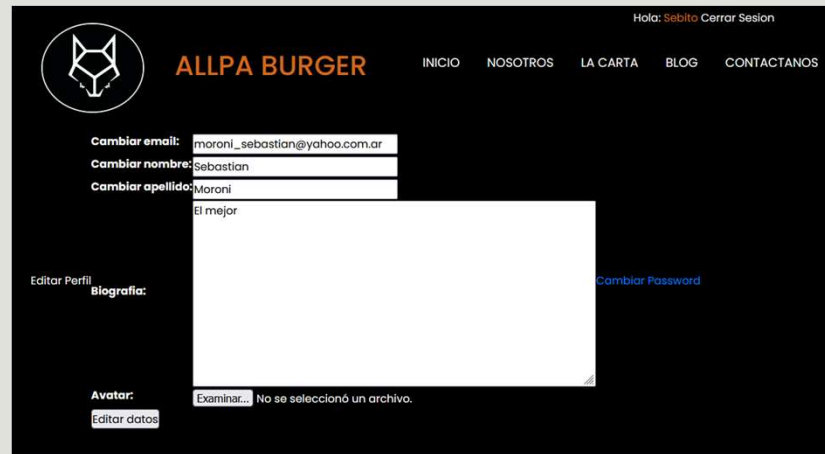
- Crear una vista de perfil donde se muestren los datos del usuario:
 - o nombre
 - o apellido
 - o email
 - o avatar
 - o biografía

TENEMOS EN LA APP UNA VISTA QUE SE ACCEDE DESDE EL NOMBRE DEL USUARIO SONDE SE VE EL DETALLE DE TODOS LOS USUARIOS CREADOS SI ESTAS LOGEADO CON SUPERUSUARIO O SI SOS USUARIO NORMAL, VES TU PROPIA INFORMACIÓN. EN AMBOS CASOS SE PUEDEN REALIZAR DISTINTAS ACCIONES DEPENDIENDO EL PERFIL DEL USUARIO.



- Desde el perfil, crear un acceso a una vista de edición de estos datos. Agregar el cambio de password.

SE ADJUNTAN IMAGEN DE
EDICIÓN DE PERFIL Y
CAMBIO DE CONTRASEÑA.



The screenshot shows the 'ALLPA BURGER' profile editing interface. At the top, there's a navigation bar with the logo, the name 'ALLPA BURGER', and links for 'INICIO', 'NOSOTROS', 'LA CARTA', 'BLOG', and 'CONTACTANOS'. The user is logged in as 'Sebito'. The main content area has a left sidebar with 'Editor Perfil' and 'Biografía:'. The main section contains form fields for 'Cambiar email:' (moroni_sebastian@yahoo.com.ar), 'Cambiar nombre:' (Sebastian), and 'Cambiar apellido:' (Moroni). Below these is a large text area for the biography with the text 'El mejor'. At the bottom left, there's an 'Avatar:' section with an 'Examinar...' button and a message 'No se seleccionó un archivo.'. On the right, there's a 'Cambiar Password' link.



The screenshot shows the 'ALLPA BURGER' password change interface. It features the same navigation bar as the previous page. The main content area has a left sidebar with 'Editor Contraseña'. The main section contains three input fields: 'Contraseña antigua:', 'Contraseña nueva:', and 'Confirmación de contraseña nueva:'. To the right of the 'Contraseña nueva:' field, there are three bullet points listing password requirements: it cannot be similar to other personal information components, it must be at least 8 characters long, it cannot be a commonly used password, and it cannot consist exclusively of numbers. At the bottom right, there is an 'Actualizar Contraseña' link. At the bottom left, there is an 'Editor datos' button.

- Aplicacion de mensajería funcional (solo para llegar al 10)

“CHATEA CON NOSOTROS” ES EL ACCESO A NUESTRA MENSAJERIA INSTANTANEA. ESTA AL HACER CLIC EN EL BOTON NOS HABRE LA MENSAJERIA EN UNA VENTANA NUEVA. AL IGUAL QUE SE PUEDE HACEDER A LAS REDES SOCIALES. POR OTRO LADO, TENEMOS UN FORMULARIO DE CONTACTO

DIRECCIÓN

Italia 1059,
San Miguel,
Buenos Aires,
Argentina.

HORARIOS

De Miércoles a Lunes:
de 11:30 a 15:30 Hs
de 19:00 a 23:55 Hs

TELÉFONO

+54911-6958-2337

EMAIL

allpaburger@gmail.com

Chatea con Nosotros

NUESTRAS REDES

