

EVUALCIÓN FINAL

Rodolfo Alfredo Martínez Román

0902-15-539

Desarrollo web

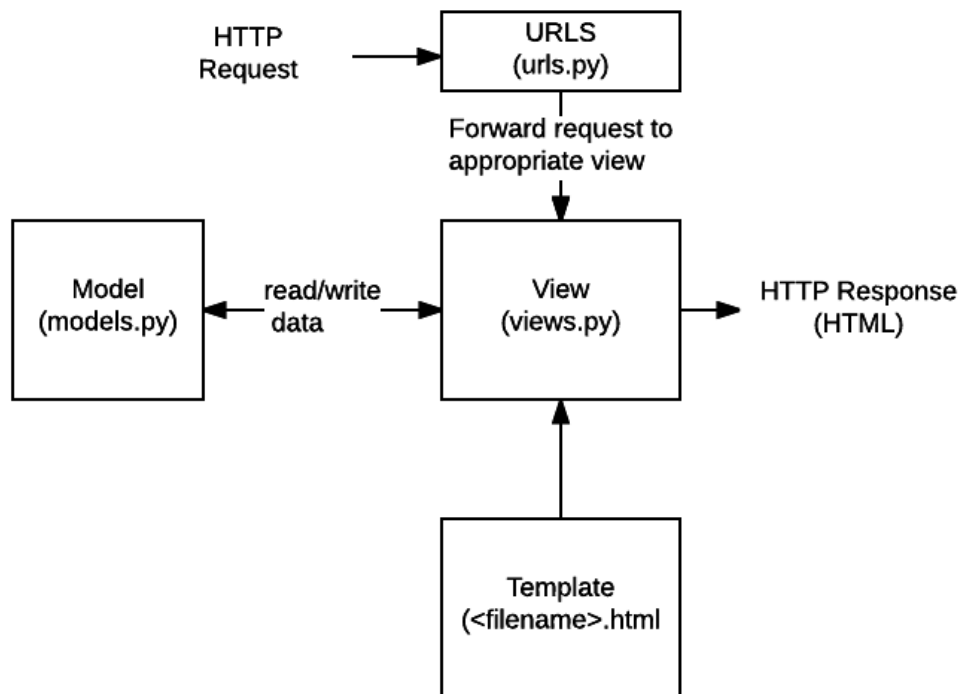


Problema propuesto

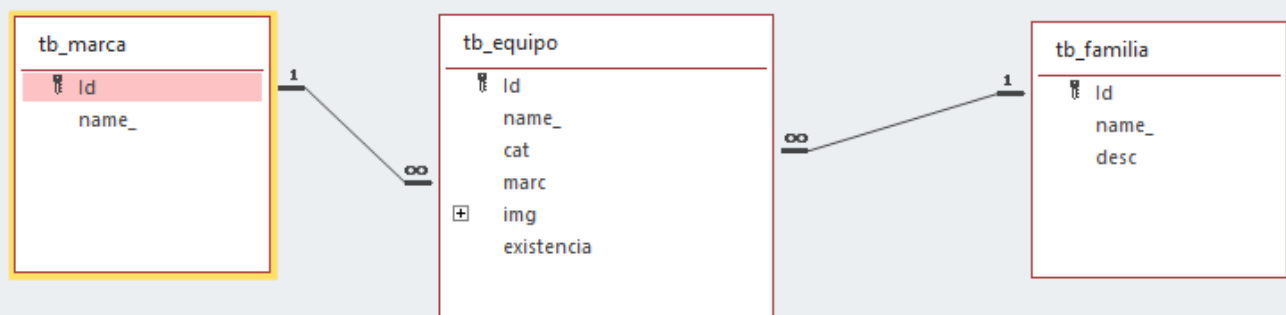
Elaborar una aplicación CRUD para la administración de equipos y suministros disponibles en una tienda de tecnología. Considere que posteriormente se podrán hacer búsquedas por su nombre o descripción. Recuerde que la presentación es importante por lo que debe considerar dicho aspecto.

Solución

Utilizando el patron de diseño MVC y sacándole provecho al proyecto realizado con Django ser realizó, una herramienta para controlar el inventario de los equipos, esto se debe actualizar cada vez que se le dé baja a un equipo, se pueden buscar por marca, familia o características del equipo.



Modelo entidad relación



Creación de modelos

```
from datetime import datetime

from django.db import models
from django.forms import model_to_dict

from config.settings import MEDIA_URL, STATIC_URL
from core.erp.choices import gender_choices
from core.models import BaseModel


class Categoria(models.Model):
    name = models.CharField(max_length=150, verbose_name='Nombre', unique=True)
    desc = models.CharField(max_length=500, null=True, blank=True,
        verbose_name='Descripción')

    def __str__(self):
        return self.name

    def toJSON(self):
        item = model_to_dict(self)
        return item

    class Meta:
        verbose_name = 'Categoria'
        verbose_name_plural = 'Categorías'
        ordering = ['id']


class Marca(models.Model):
    name = models.CharField(max_length=150, verbose_name='Marca', unique=True)

    def __str__(self):
        return self.name

    def toJSON(self):
        item = model_to_dict(self)
        return item

    class Meta:
        verbose_name = 'Marca'
        verbose_name_plural = 'Marcas'
        ordering = ['id']


class Equipos(models.Model):
    name = models.CharField(max_length=150, verbose_name='Nombre', unique=True)
    cat = models.ForeignKey(Categoria, on_delete=models.CASCADE,
        verbose_name='Categoría')
    marc = models.ForeignKey(Marca, on_delete=models.CASCADE, verbose_name='Marca')
    image = models.ImageField(upload_to='product/%Y/%m/%d', null=True, blank=True,
        verbose_name='Imagen')
    existencia = models.DecimalField(default=0.00, max_digits=9, decimal_places=2,
        verbose_name='Existencia')

    def __str__(self):
        return self.name

    def toJSON(self):
        item = model_to_dict(self)
        item['cat'] = self.cat.toJSON()
        item['marc'] = self.marc.toJSON()
        item['image'] = self.get_image()
```

```

        item['existencia'] = format(self.existencia, '.2f')
        return item

    def get_image(self):
        if self.image:
            return '{}{}'.format(MEDIA_URL, self.image)
        return '{}{}'.format(STATIC_URL, 'img/empty.png')

    class Meta:
        verbose_name = 'Producto'
        verbose_name_plural = 'Productos'
        ordering = ['id']

```

Vistas

```

from django.contrib.auth.mixins import LoginRequiredMixin
from django.http import JsonResponse
from django.urls import reverse_lazy
from django.utils.decorators import method_decorator
from django.views.decorators.csrf import csrf_exempt
from django.views.generic import ListView, CreateView, UpdateView, DeleteView

from core.erp.forms import CategoryForm
from core.erp.mixins import ValidatePermissionRequiredMixin
from core.erp.models import Categoria

class CategoryListView(LoginRequiredMixin, ValidatePermissionRequiredMixin, ListView):
    model = Categoria
    template_name = 'category/list.html'
    permission_required = 'view_categoria'

    @method_decorator(csrf_exempt)
    def dispatch(self, request, *args, **kwargs):
        return super().dispatch(request, *args, **kwargs)

    def post(self, request, *args, **kwargs):
        data = {}
        try:
            action = request.POST['action']
            if action == 'searchdata':
                data = []
                position = 1
                for i in Categoria.objects.all():
                    item = i.toJSON()
                    item['position'] = position
                    data.append(item)
                    position += 1
                # data.append(i.toJSON())
            else:
                data['error'] = 'Ha ocurrido un error'
        except Exception as e:
            data['error'] = str(e)
        return JsonResponse(data, safe=False)

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['title'] = 'Listado de Familias'
        context['create_url'] = reverse_lazy('erp:category_create')
        context['list_url'] = reverse_lazy('erp:category_list')
        context['entity'] = 'Familias'
        return context

```

```

class CategoryCreateView(LoginRequiredMixin, ValidatePermissionRequiredMixin,
CreateView):
    model = Categoria
    form_class = CategoryForm
    template_name = 'category/create.html'
    success_url = reverse_lazy('erp:category_list')
    permission_required = 'add_categoria'
    url_redirect = success_url

    def dispatch(self, request, *args, **kwargs):
        return super().dispatch(request, *args, **kwargs)

    def post(self, request, *args, **kwargs):
        data = {}
        try:
            action = request.POST['action']
            if action == 'add':
                form = self.get_form()
                data = form.save()
            else:
                data['error'] = 'No ha ingresado a ninguna opción'
        except Exception as e:
            data['error'] = str(e)
        return JsonResponse(data)

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['title'] = 'Creación una Familia'
        context['entity'] = 'Familias'
        context['list_url'] = self.success_url
        context['action'] = 'add'
        return context

class CategoryUpdateView(LoginRequiredMixin, ValidatePermissionRequiredMixin,
UpdateView):
    model = Categoria
    form_class = CategoryForm
    template_name = 'category/create.html'
    success_url = reverse_lazy('erp:category_list')
    permission_required = 'change_categoria'
    url_redirect = success_url

    def dispatch(self, request, *args, **kwargs):
        self.object = self.get_object()
        return super().dispatch(request, *args, **kwargs)

    def post(self, request, *args, **kwargs):
        data = {}
        try:
            action = request.POST['action']
            if action == 'edit':
                form = self.get_form()
                data = form.save()
            else:
                data['error'] = 'No ha ingresado a ninguna opción'
        except Exception as e:
            data['error'] = str(e)
        return JsonResponse(data)

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)

```

```

        context['title'] = 'Edición una familia'
        context['entity'] = 'Familias'
        context['list_url'] = self.success_url
        context['action'] = 'edit'
        return context

```

```

class CategoryDeleteView(LoginRequiredMixin, ValidatePermissionRequiredMixin,
DeleteView):
    model = Categoria
    template_name = 'category/delete.html'
    success_url = reverse_lazy('erp:category_list')
    permission_required = 'delete_categoria'
    url_redirect = success_url

    def dispatch(self, request, *args, **kwargs):
        self.object = self.get_object()
        return super().dispatch(request, *args, **kwargs)

    def post(self, request, *args, **kwargs):
        data = {}
        try:
            self.object.delete()
        except Exception as e:
            data['error'] = str(e)
        return JsonResponse(data)

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['title'] = 'Eliminación de una familia'
        context['entity'] = 'Familias'
        context['list_url'] = self.success_url
        return context

```

```

from django.contrib.auth.mixins import LoginRequiredMixin
from django.http import JsonResponse
from django.urls import reverse_lazy
from django.utils.decorators import method_decorator
from django.views.decorators.csrf import csrf_exempt
from django.views.generic import ListView, CreateView, UpdateView, DeleteView

from core.erp.forms import MarcaForm
from core.erp.mixins import ValidatePermissionRequiredMixin
from core.erp.models import Marca

```

```

class MarcaListView(LoginRequiredMixin, ValidatePermissionRequiredMixin, ListView):
    model = Marca
    template_name = 'marca/list.html'
    permission_required = 'view_marca'

    @method_decorator(csrf_exempt)
    def dispatch(self, request, *args, **kwargs):
        return super().dispatch(request, *args, **kwargs)

    def post(self, request, *args, **kwargs):
        data = {}
        try:
            action = request.POST['action']
            if action == 'searchdata':
                data = []

```

```

        position = 1
        for i in Marca.objects.all():
            item = i.toJSON()
            item['position'] = position
            data.append(item)
            position += 1
            # data.append(i.toJSON())
    else:
        data['error'] = 'Ha ocurrido un error'
except Exception as e:
    data['error'] = str(e)
return JsonResponse(data, safe=False)

def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    context['title'] = 'Listado de marcas'
    context['create_url'] = reverse_lazy('erp:marca_create')
    context['list_url'] = reverse_lazy('erp:marca_list')
    context['entity'] = 'Marcas'
    return context

```

```

class MarcaCreateView(LoginRequiredMixin, ValidatePermissionRequiredMixin, CreateView):
    model = Marca
    form_class = MarcaForm
    template_name = 'marca/create.html'
    success_url = reverse_lazy('erp:marca_list')
    permission_required = 'add_marca'
    url_redirect = success_url

    def dispatch(self, request, *args, **kwargs):
        return super().dispatch(request, *args, **kwargs)

    def post(self, request, *args, **kwargs):
        data = {}
        try:
            action = request.POST['action']
            if action == 'add':
                form = self.get_form()
                data = form.save()
            else:
                data['error'] = 'No ha ingresado a ninguna opción'
        except Exception as e:
            data['error'] = str(e)
        return JsonResponse(data)

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['title'] = 'Creación una marca'
        context['entity'] = 'Marcas'
        context['list_url'] = self.success_url
        context['action'] = 'add'
        return context

```

```

class MarcaUpdateView(LoginRequiredMixin, ValidatePermissionRequiredMixin, UpdateView):
    model = Marca
    form_class = MarcaForm
    template_name = 'marca/create.html'
    success_url = reverse_lazy('erp:marca_list')
    permission_required = 'change_marca'
    url_redirect = success_url

```

```

def dispatch(self, request, *args, **kwargs):
    self.object = self.get_object()
    return super().dispatch(request, *args, **kwargs)

def post(self, request, *args, **kwargs):
    data = {}
    try:
        action = request.POST['action']
        if action == 'edit':
            form = self.get_form()
            data = form.save()
        else:
            data['error'] = 'No ha ingresado a ninguna opción'
    except Exception as e:
        data['error'] = str(e)
    return JsonResponse(data)

def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    context['title'] = 'Edición una marca'
    context['entity'] = 'Marcas'
    context['list_url'] = self.success_url
    context['action'] = 'edit'
    return context

```

```

class MarcaDeleteView(LoginRequiredMixin, ValidatePermissionRequiredMixin, DeleteView):
    model = Marca
    template_name = 'marca/delete.html'
    success_url = reverse_lazy('erp:marca_list')
    permission_required = 'delete_marca'
    url_redirect = success_url

    def dispatch(self, request, *args, **kwargs):
        self.object = self.get_object()
        return super().dispatch(request, *args, **kwargs)

    def post(self, request, *args, **kwargs):
        data = {}
        try:
            self.object.delete()
        except Exception as e:
            data['error'] = str(e)
        return JsonResponse(data)

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['title'] = 'Eliminación de una marca'
        context['entity'] = 'Marcas'
        context['list_url'] = self.success_url
        return context

```

```

from django.contrib.auth.decorators import login_required
from django.contrib.auth.mixins import LoginRequiredMixin
from django.http import JsonResponse
from django.urls import reverse_lazy
from django.utils.decorators import method_decorator
from django.views.decorators.csrf import csrf_exempt
from django.views.generic import ListView, CreateView, UpdateView, DeleteView

```



```

from core.erp.forms import ProductForm
from core.erp.mixins import ValidatePermissionRequiredMixin
from core.erp.models import Equipos

class ProductListView(LoginRequiredMixin, ValidatePermissionRequiredMixin, ListView):
    model = Equipos
    template_name = 'product/list.html'
    permission_required = 'view_equipos'

    @method_decorator(csrf_exempt)
    def dispatch(self, request, *args, **kwargs):
        return super().dispatch(request, *args, **kwargs)

    def post(self, request, *args, **kwargs):
        data = {}
        try:
            action = request.POST['action']
            if action == 'searchdata':
                data = []
                for i in Equipos.objects.all():
                    data.append(i.toJSON())
            else:
                data['error'] = 'Ha ocurrido un error'
        except Exception as e:
            data['error'] = str(e)
        return JsonResponse(data, safe=False)

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['title'] = 'Listado de equipos'
        context['create_url'] = reverse_lazy('erp:product_create')
        context['list_url'] = reverse_lazy('erp:product_list')
        context['entity'] = 'Equipos'
        return context

class ProductCreateView(LoginRequiredMixin, ValidatePermissionRequiredMixin, CreateView):
    model = Equipos
    form_class = ProductForm
    template_name = 'product/create.html'
    success_url = reverse_lazy('erp:product_list')
    permission_required = 'add_equipos'
    url_redirect = success_url

    def dispatch(self, request, *args, **kwargs):
        return super().dispatch(request, *args, **kwargs)

    def post(self, request, *args, **kwargs):
        data = {}
        try:
            action = request.POST['action']
            if action == 'add':
                form = self.get_form()
                data = form.save()
            else:
                data['error'] = 'No ha ingresado a ninguna opción'
        except Exception as e:
            data['error'] = str(e)
        return JsonResponse(data)

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)

```

```
context['title'] = 'Creación de un equipo'
context['entity'] = 'Equipos'
context['list_url'] = self.success_url
context['action'] = 'add'
return context
```

```
class ProductUpdateView(LoginRequiredMixin, ValidatePermissionRequiredMixin, UpdateView):
    model = Equipos
    form_class = ProductForm
    template_name = 'product/create.html'
    success_url = reverse_lazy('erp:product_list')
    permission_required = 'change_equipos'
    url_redirect = success_url
```

```
def dispatch(self, request, *args, **kwargs):
    self.object = self.get_object()
    return super().dispatch(request, *args, **kwargs)

def post(self, request, *args, **kwargs):
    data = {}
    try:
        action = request.POST['action']
        if action == 'edit':
            form = self.get_form()
            data = form.save()
        else:
            data['error'] = 'No ha ingresado a ninguna opción'
    except Exception as e:
        data['error'] = str(e)
    return JsonResponse(data)

def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    context['title'] = 'Edición de un equipo'
    context['entity'] = 'Equipos'
    context['list_url'] = self.success_url
    context['action'] = 'edit'
    return context
```

```
class ProductDeleteView(LoginRequiredMixin, ValidatePermissionRequiredMixin, DeleteView):
    model = Equipos
    template_name = 'product/delete.html'
    success_url = reverse_lazy('erp:product_list')
    permission_required = 'delete_equipos'
    url_redirect = success_url
```

```
@method_decorator(login_required)
def dispatch(self, request, *args, **kwargs):
    self.object = self.get_object()
    return super().dispatch(request, *args, **kwargs)

def post(self, request, *args, **kwargs):
    data = {}
    try:
        self.object.delete()
    except Exception as e:
        data['error'] = str(e)
    return JsonResponse(data)

def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
```

```

context['title'] = 'Eliminación de un equipo'
context['entity'] = 'Equipos'
context['list_url'] = self.success_url
return context

```

Formularios

```

from datetime import datetime

from django.forms import *

from core.erp.models import Categoria, Equipos, Marca

class CategoryForm(ModelForm):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        # for form in self.visible_fields():
        #     form.field.widget.attrs['class'] = 'form-control'
        #     form.field.widget.attrs['autocomplete'] = 'off'
        self.fields['name'].widget.attrs['autofocus'] = True

    class Meta:
        model = Categoria
        fields = '__all__'
        widgets = {
            'name': TextInput(
                attrs={
                    'placeholder': 'Ingrese un nombre',
                }
            ),
            'desc': Textarea(
                attrs={
                    'placeholder': 'Ingrese un nombre',
                    'rows': 3,
                    'cols': 3
                }
            ),
        }

    def save(self, commit=True):
        data = {}
        form = super()
        try:
            if form.is_valid():
                form.save()
            else:
                data['error'] = form.errors
        except Exception as e:
            data['error'] = str(e)
        return data

class MarcaForm(ModelForm):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        # for form in self.visible_fields():
        #     form.field.widget.attrs['class'] = 'form-control'
        #     form.field.widget.attrs['autocomplete'] = 'off'
        self.fields['name'].widget.attrs['autofocus'] = True

    class Meta:

```

```

model = Marca
fields = '__all__'
widgets = {
    'name': TextInput(
        attrs={
            'placeholder': 'Ingrese un marca nueva',
        }
    ),
}

def save(self, commit=True):
    data = {}
    form = super()
    try:
        if form.is_valid():
            form.save()
        else:
            data['error'] = form.errors
    except Exception as e:
        data['error'] = str(e)
    return data

class ProductForm(ModelForm):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.fields['name'].widget.attrs['autofocus'] = True

class Meta:
    model = Equipos
    fields = '__all__'
    widgets = {
        'name': TextInput(
            attrs={
                'placeholder': 'Ingrese un nombre',
            }
        ),
        'cat': Select(
            attrs={
                'class': 'select2',
                'style': 'width: 100%'
            }
        ),
        'marc': Select(
            attrs={
                'class': 'select2',
                'style': 'width: 100%'
            }
        ),
    }

def save(self, commit=True):
    data = {}
    form = super()
    try:
        if form.is_valid():
            form.save()
        else:
            data['error'] = form.errors
    except Exception as e:
        data['error'] = str(e)

```

```

        return data

from django import forms

# para recuperar contraseña
from core.user.models import User

# formulario para enviar usuario para resetear
class ResetPasswordForm(forms.Form):
    username = forms.CharField(widget=forms.TextInput(attrs={
        'placeholder': 'Ingrese su nombre de usuario',
        'class': 'form-control',
        'autocomplete': 'off'
    })))

    # validar si existe el usuario
    def clean(self):
        cleaned = super().clean()
        if not User.objects.filter(username=cleaned['username']).exists():
            # modificar mensaje de errores
            # self._errors['error'] = self._errors.get('error', self.error_class())
            # self._errors['error'].append('El usuario no existe')
            raise forms.ValidationError('El usuario no existe')
        return cleaned

# saber que usuario
    def get_user(self):
        username = self.cleaned_data.get('username')
        return User.objects.get(username=username)

# formulario para resetear contraseña
class ChangePasswordForm(forms.Form):
    password = forms.CharField(widget=forms.PasswordInput(attrs={
        'placeholder': 'Ingrese contraseña nueva',
        'class': 'form-control',
        'autocomplete': 'off'
    })))

    confirmPassword = forms.CharField(widget=forms.PasswordInput(attrs={
        'placeholder': 'Repita la contraseña',
        'class': 'form-control',
        'autocomplete': 'off'
    })))

    def clean(self):
        cleaned = super().clean()
        password = cleaned['password']
        confirmPassword = cleaned['confirmPassword']
        if password != confirmPassword:
            # self._errors['error'] = self._errors.get('error', self.error_class())
            # self._errors['error'].append('El usuario no existe')
            raise forms.ValidationError('Las contraseñas deben ser iguales')
        return cleaned

```

URLS

```
from django.urls import path
from core.erp.views.category.views import *
from core.erp.views.marca.views import *
from core.erp.views.product.views import *

app_name = 'erp'

urlpatterns = [

    path('category/list/', CategoryListView.as_view(), name='category_list'),
    path('category/add/', CategoryCreateView.as_view(), name='category_create'),
    path('category/update/<int:pk>/', CategoryUpdateView.as_view(),
name='category_update'),
    path('category/delete/<int:pk>/', CategoryDeleteView.as_view(),
name='category_delete'),

    path('product/list/', ProductListView.as_view(), name='product_list'),
    path('product/add/', ProductCreateView.as_view(), name='product_create'),
    path('product/update/<int:pk>/', ProductUpdateView.as_view(), name='product_update'),
    path('product/delete/<int:pk>/', ProductDeleteView.as_view(), name='product_delete'),

    path('marca/list/', MarcaListView.as_view(), name='marca_list'),
    path('marca/add/', MarcaCreateView.as_view(), name='marca_create'),
    path('marca/update/<int:pk>/', MarcaUpdateView.as_view(), name='marca_update'),
    path('marca/delete/<int:pk>/', MarcaDeleteView.as_view(), name='marca_delete'),

    path('dashboard/', CategoryListView.as_view(), name='dashboard'),

]
```

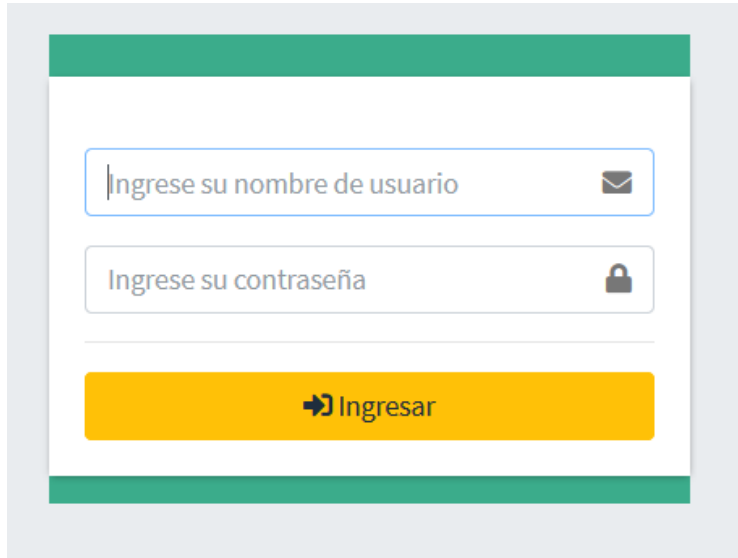
```
from django.urls import path

from core.login.views import *

urlpatterns = [
    path('', LoginFormView.as_view(), name='login'),
    path('logout/', LogoutView.as_view(), name='logout'),
    path('reset/password/', ResetPasswordView.as_view(), name='reset_password'),
    path('change/password/<str:token>/', ChangePasswordView.as_view(),
name='change_password')
]
```

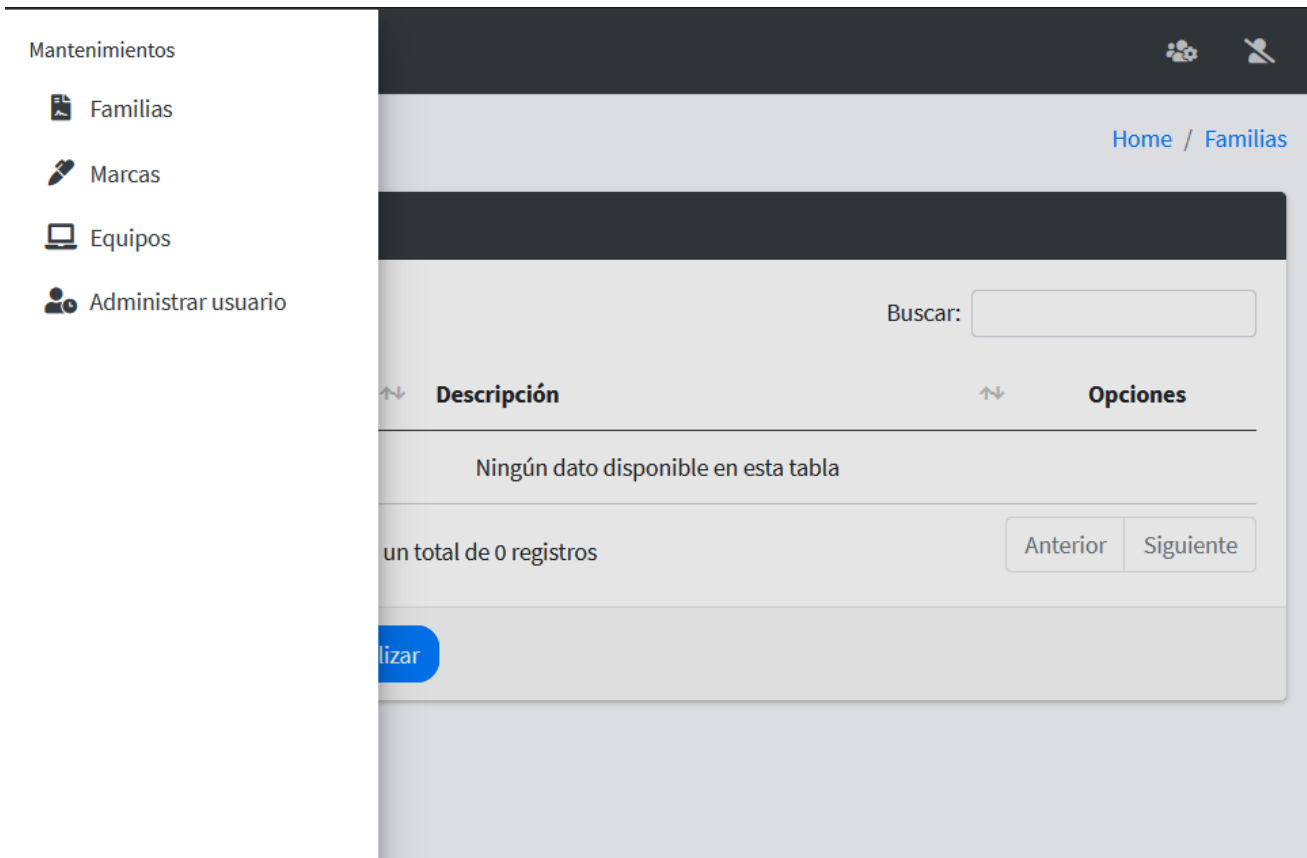
Funcionamiento

Login



A login form with a green header and footer. It contains two input fields: "Ingrese su nombre de usuario" with an envelope icon and "Ingrese su contraseña" with a lock icon. Below the fields is a yellow button labeled "Ingresar" with a right-pointing arrow icon.

Menu Principal



The main menu on the left lists: "Mantenimientos", "Familias" (with a family icon), "Marcas" (with a pencil icon), "Equipos" (with a laptop icon), and "Administrar usuario" (with a user icon). The main content area shows a breadcrumb "Home / Familias", a search bar labeled "Buscar:", and a table with columns "Descripción" and "Opciones". The table is empty, displaying "Ningún dato disponible en esta tabla". At the bottom, it says "un total de 0 registros" and has "Anterior" and "Siguiete" buttons. A blue button labeled "lizar" is partially visible at the bottom left.

Formulario para ingreso de familias

Inicio

Home / Familias

Bienvenido

+ Creación una Familia

Nombre:

Ingrese un nombre

Descripción:

Ingrese un nombre

Guardar registro

Cancelar

Confirmación para ingreso de familias

+ Creación una Familia

Nombre:

Antivirus

Descripción:

Ingrese un nombre

Guardar registro

Cancelar

Notificación

¿Estas seguro de realizar la siguiente acción?

Si

No

Informe de ingreso de familias

Bienvenido

Home / Familias

+ Creación una Familia

Nombre:

Antivirus

Descripción:

Ingrese un nombre

Guardar registro

Exito

Registro agregado correctamente

Lista de familias

Q

Listado de Familias

Mostrar

10

registros

Buscar:

Nro	↑↓	Nombre	↑↓	Descripción	↑↓	Opciones
1		Antivirus				<div><div></div><div></div></div>

Mostrando registros del 1 al 1 de un total de 1 registros

Anterior

1

Siguiente


+ Nuevo registro

Actualizar

Eliminar familias


Bienvenido


[Home](#) / [Familias](#)

 Eliminación de una familia

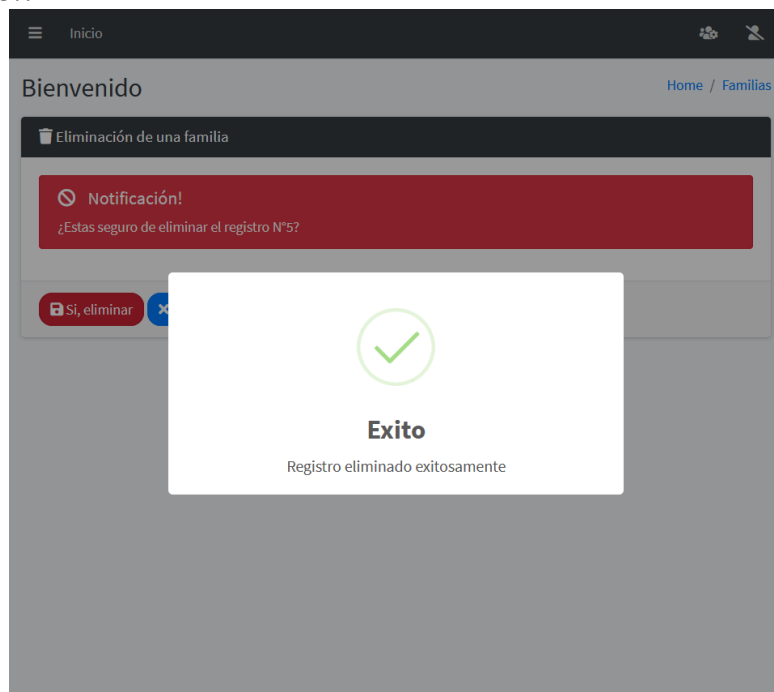
 **Notificación!**

¿Estas seguro de eliminar el registro N°5?

 Si, eliminar

 Cancelar eliminacion

Informe de eliminación




Formulario para ingreso de marcas

Lista de registros

Bienvenido

[Home](#) / [Marcas](#)

 Listado de marcas


Mostrar registros


Buscar:

Nro	↕	Nombre	↕	Opciones
1		Avast		 


Mostrando registros del 1 al 1 de un total de 1 registros

[Anterior](#) [1](#) [Siguiete](#)

 Nuevo registro

 Actualizar


Formulario para registros

 Inicio





Bienvenido

[Home](#) / [Marcas](#)

 Creación una marca

Marca:

 Guardar registro

 Cancelar

Confirmación de ingreso de registro

Bienvenido Home / Marcas

+ Creación una marca

Marca:

Avast

Guardar registro

Cancelar

Notificación

¿Estas seguro de realizar la siguiente acción?

Si

No

Eliminar registro

Bienvenido Home / Marcas

Eliminación de una marca

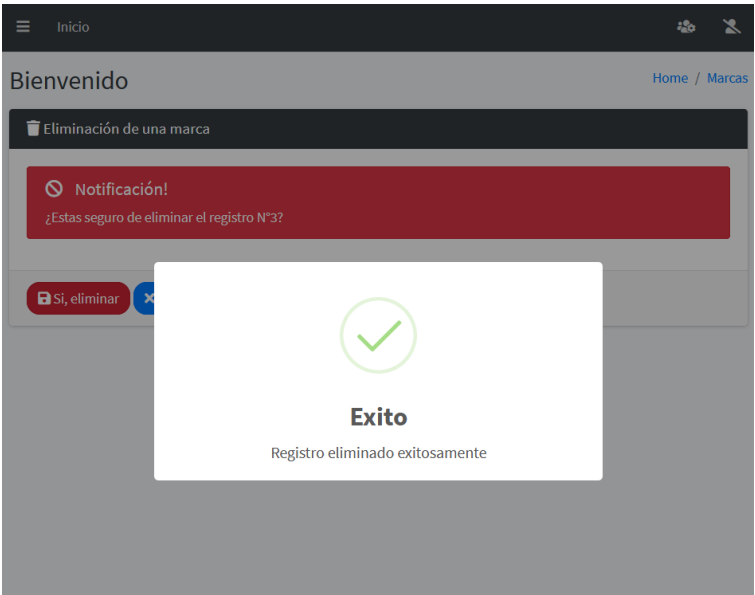
Notificación!

¿Estas seguro de eliminar el registro N°3?

Si, eliminar

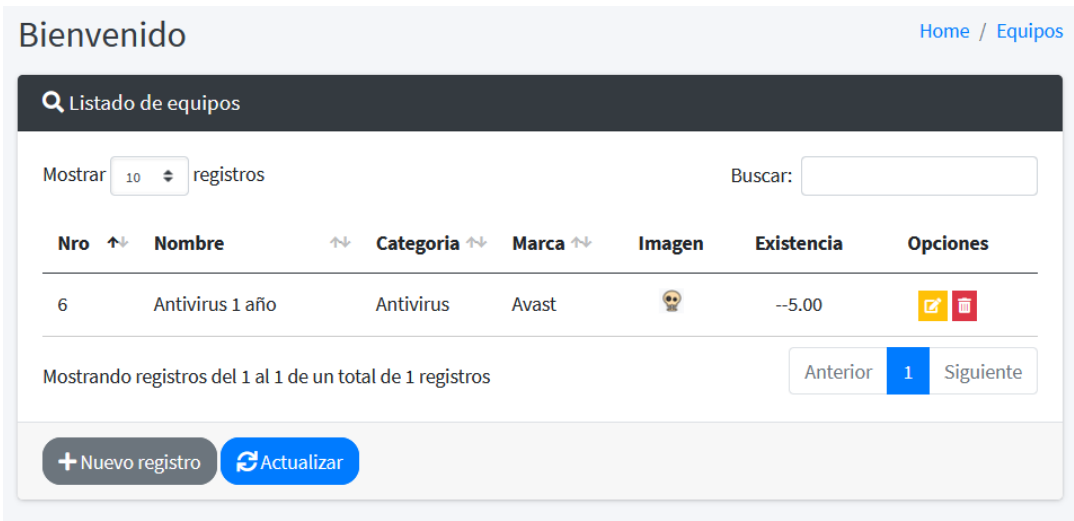
Cancelar eliminacion

Confirmación de eliminación

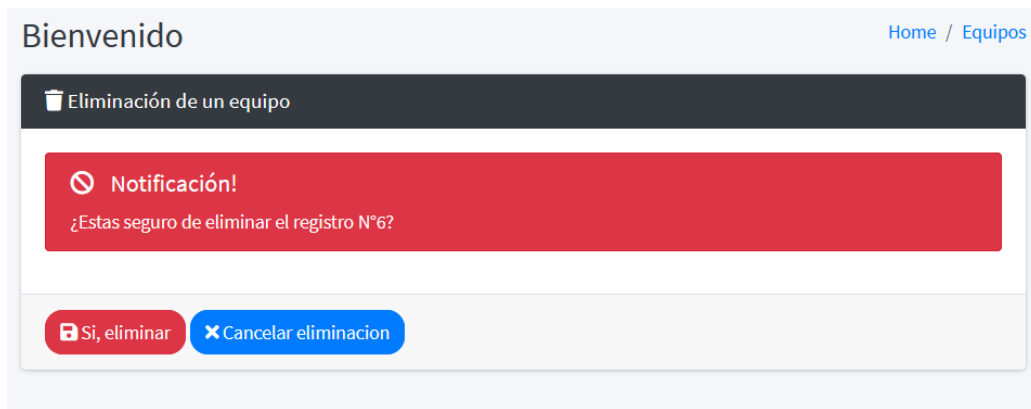


Formulario para ingreso de equipos

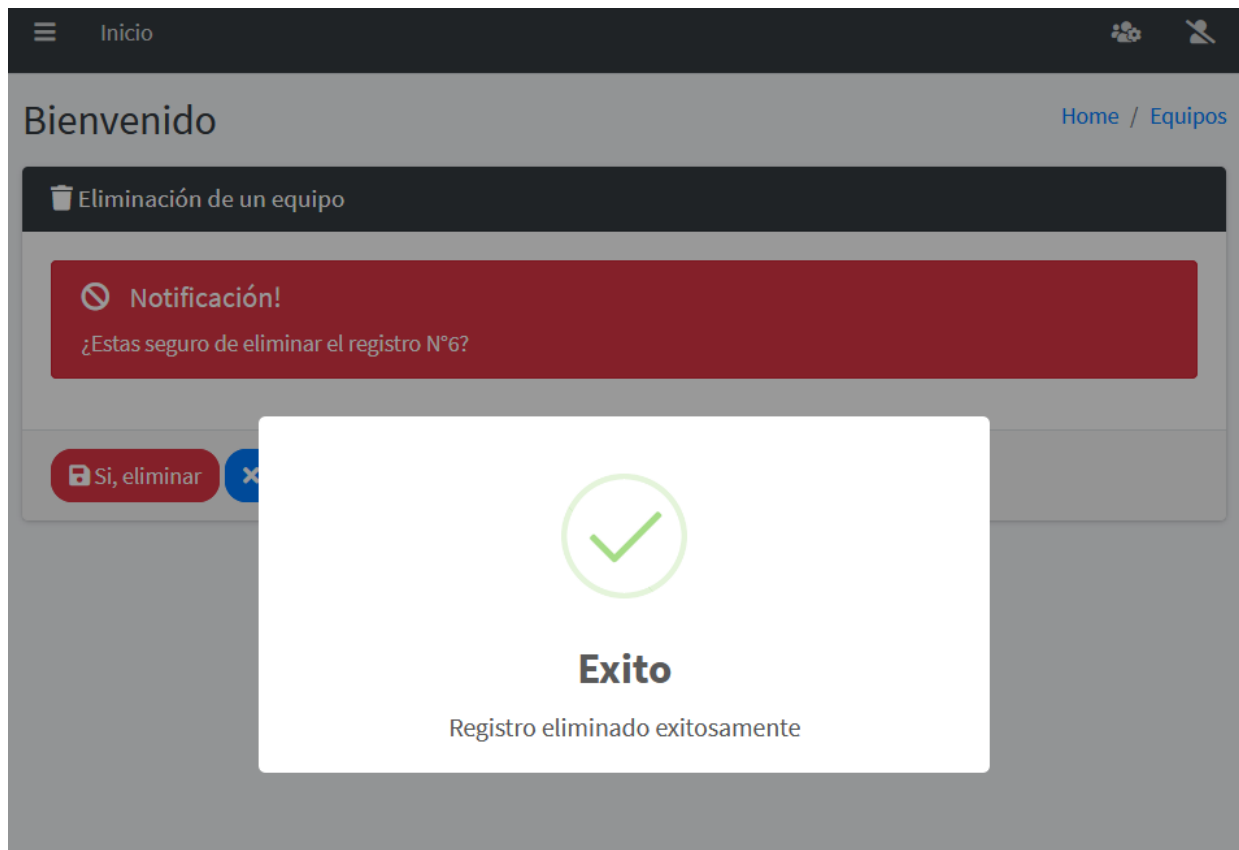
Lista de registros



Eliminar registro



Confirmación de eliminación



Términos Empleados

URLs: Aunque es posible procesar peticiones de cada URL individual vía una función individual, es mucho más sostenible escribir una función de visualización separada para cada recurso. Se usa un mapeador URL para redirigir las peticiones HTTP a la vista apropiada basándose en la URL de la petición. El mapeador URL se usa para redirigir las peticiones HTTP a la vista apropiada basándose en la URL de la petición. El mapeador URL puede también emparejar patrones de cadenas o dígitos específicos que aparecen en una URL y los pasan a la función de visualización como datos.

Vista (View): Una vista es una función de gestión de peticiones que recibe peticiones HTTP y devuelve respuestas HTTP. Las vistas acceden a los datos que necesitan para satisfacer las peticiones por medio de modelos, y delegan el formateo de la respuesta a las plantillas ("templates").

Modelos (Models): Los Modelos son objetos de Python que definen la estructura de los datos de una aplicación y proporcionan mecanismos para gestionar (añadir, modificar y borrar) y consultar registros en la base de datos.

Plantillas (Templates): una plantilla (template) es un fichero de texto que define la estructura o diagrama de otro fichero (tal como una página HTML), con marcadores de posición que se utilizan para representar el contenido real. Una vista puede crear dinámicamente una página usando una plantilla, rellenandola con datos de un modelo. Una plantilla se puede usar para definir la estructura de cualquier tipo de fichero; ¡no tiene porqué ser HTML!