Examen Final

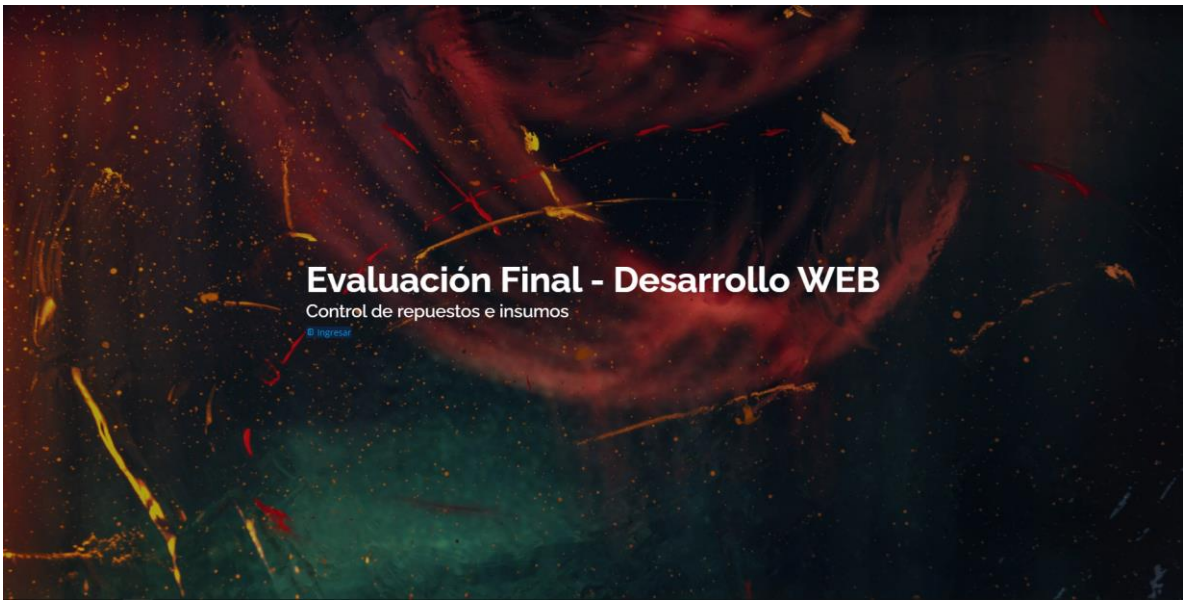
0902-14-20076 ARIEL GERARDO PEINADO BARRIENTOS

DESARROLLO WEB-2020

Evaluación Final

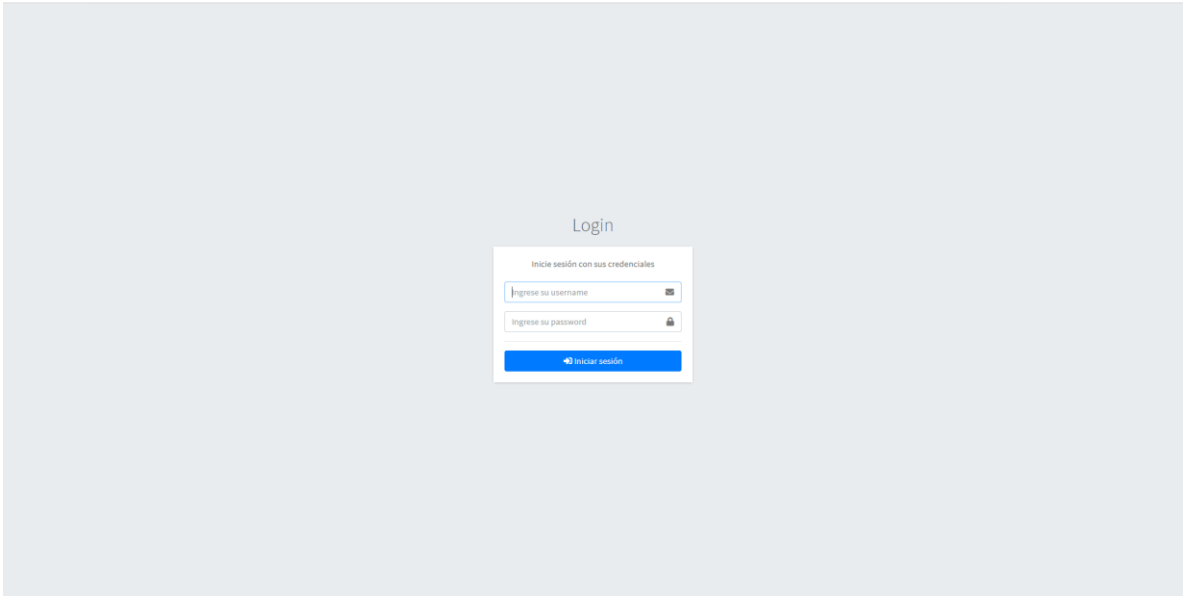
Pantallas

Inicio



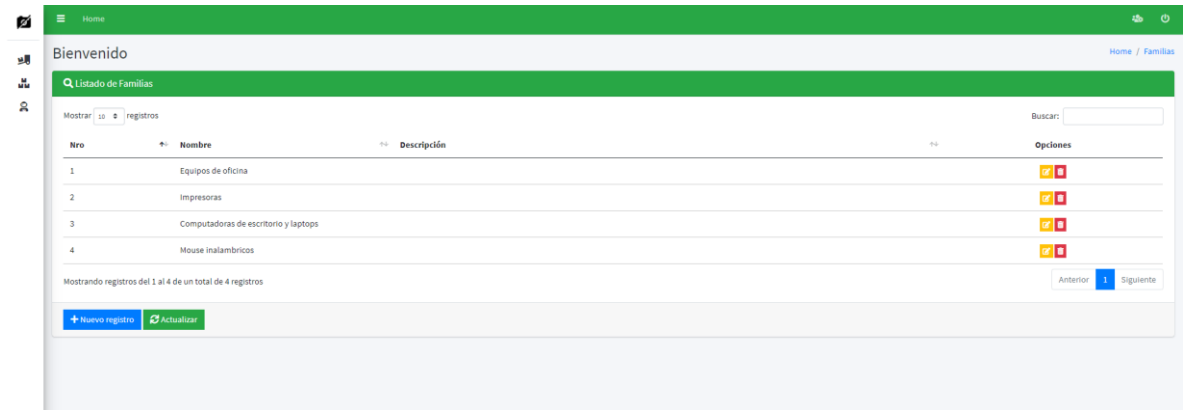
Pagina de aterrizaje para iniciar la aplicación

Login



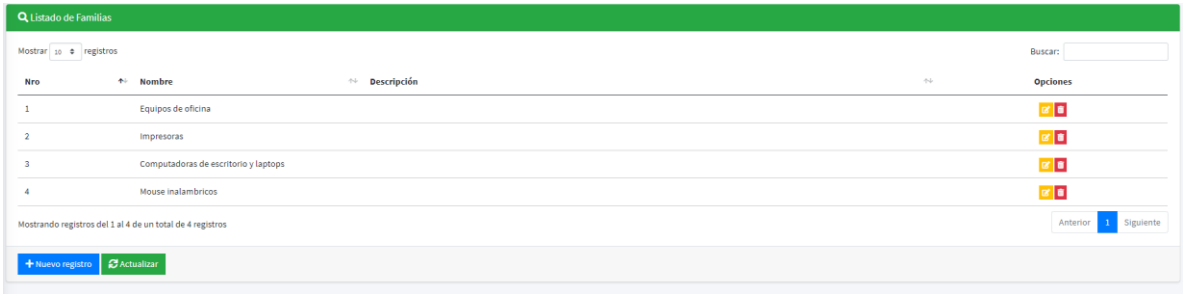
Permite registrarse

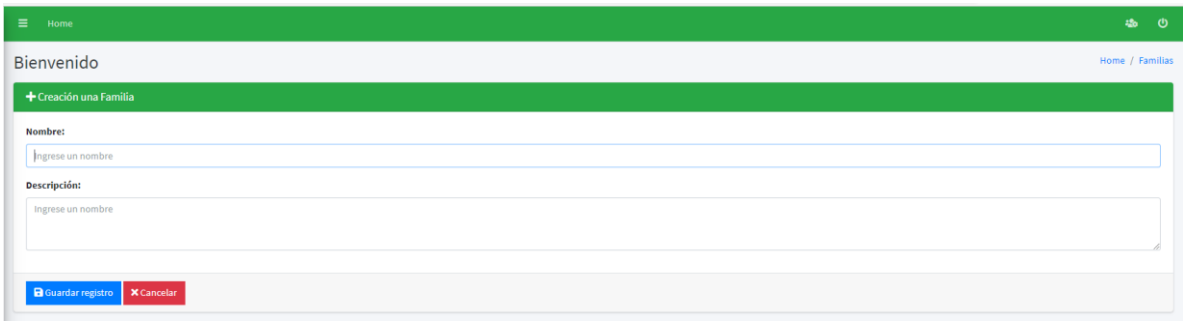
Pagina principal de aplicación

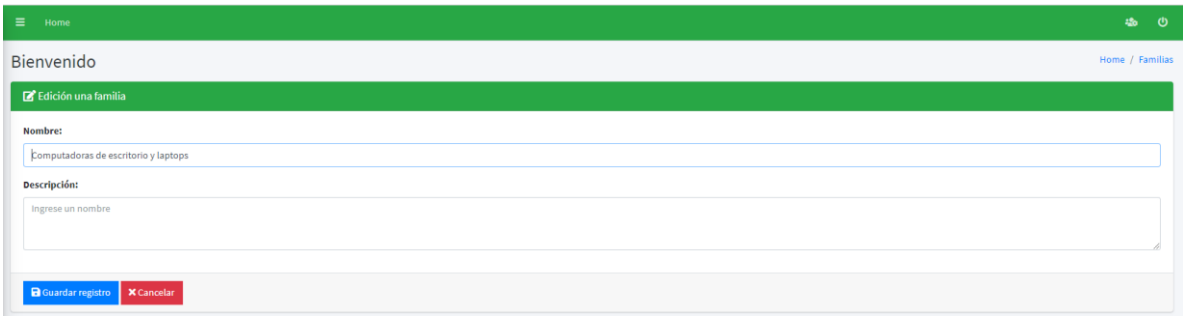


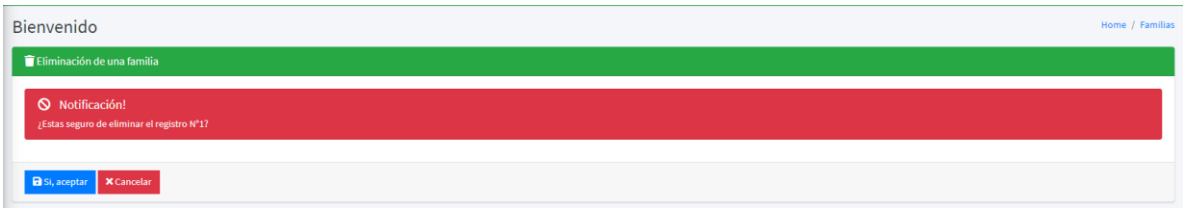
Acá el usuario podrá elegir si dese ingresar las familias de los equipos, los equipos o administrar los usuarios.

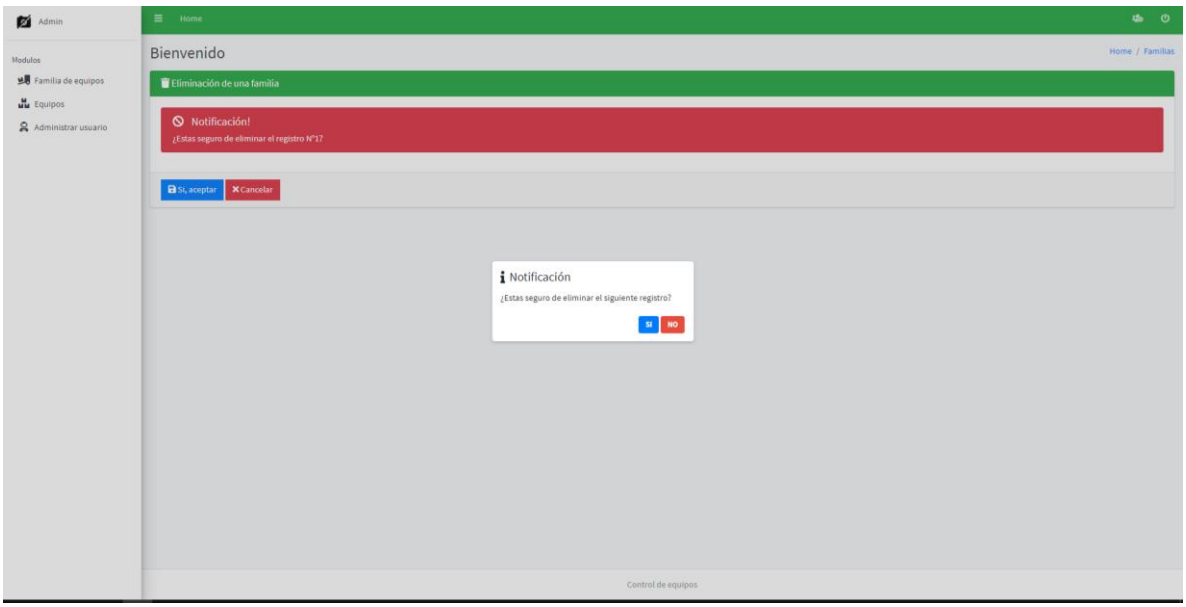
Módulo Familia de equipos





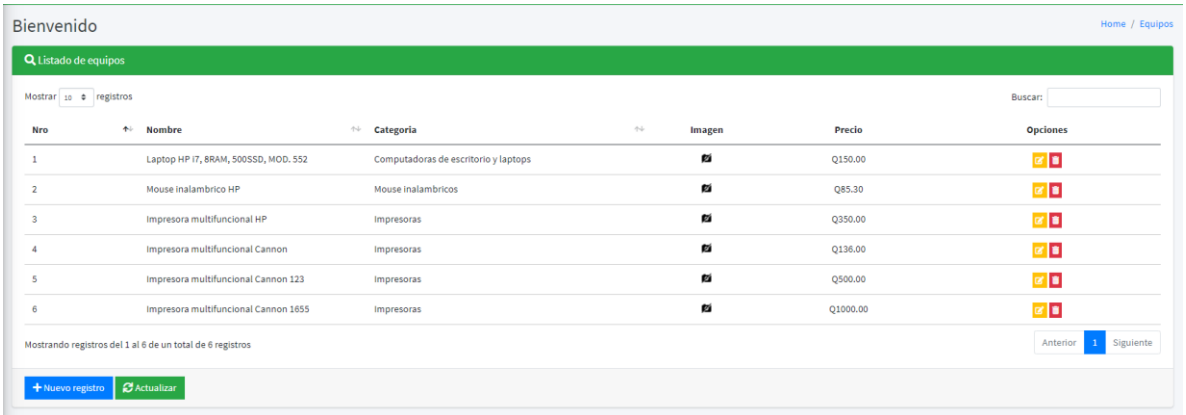




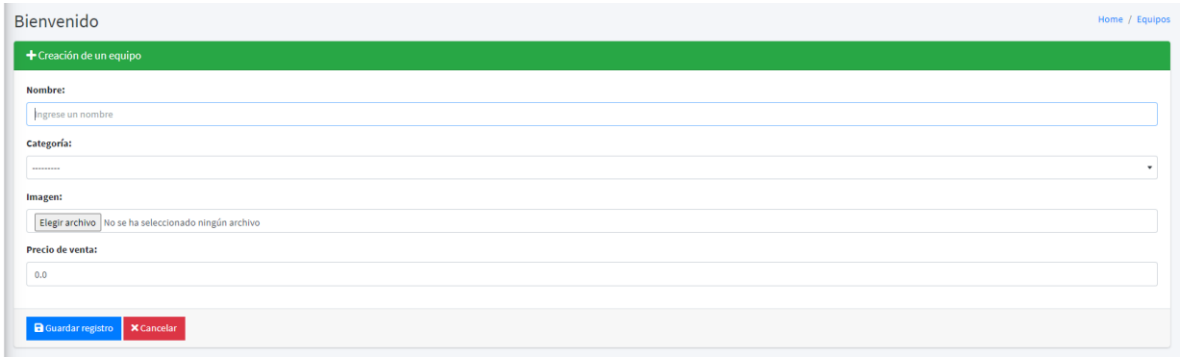


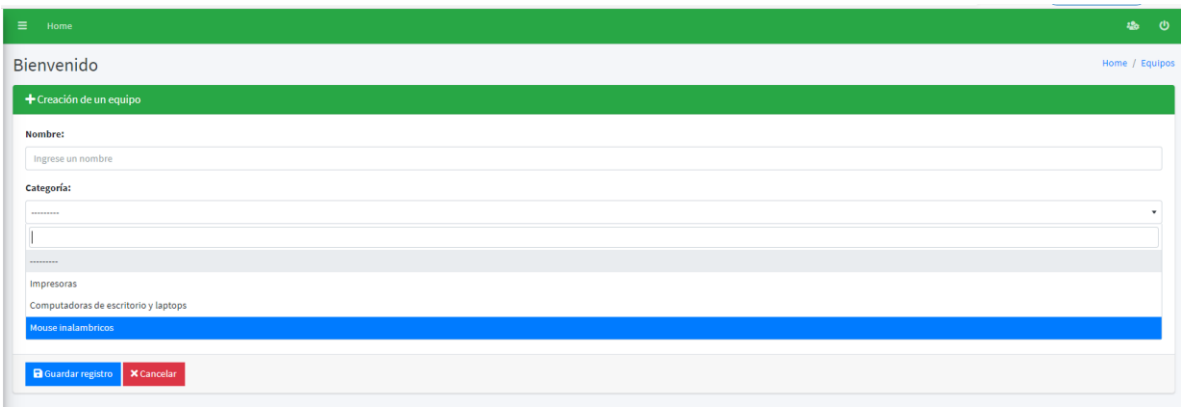
Permite al usuario decidir si está seguro de ejecutar la instrucción.

Modulo Equipos

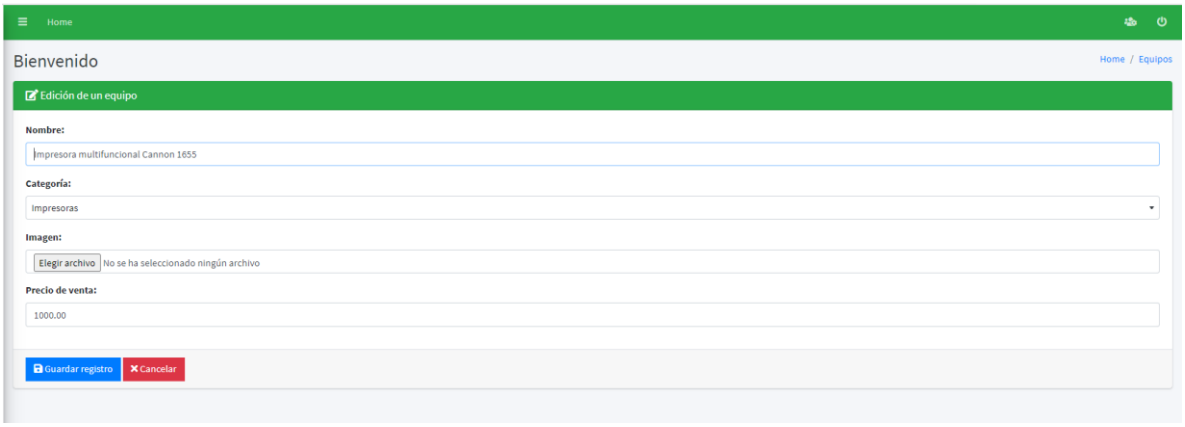


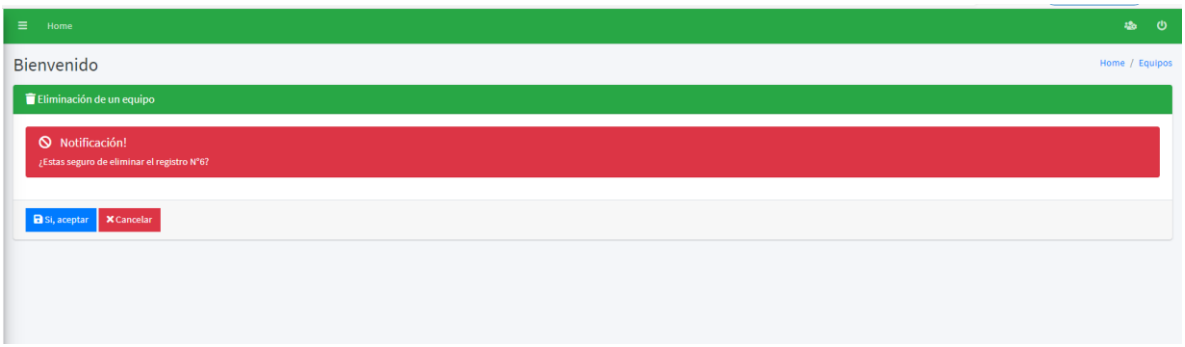
Agregar un equipo, esto me permite agregar un nuevo equipo especificando el nombre el precio por unidad y la fotografía si es necesario.





Puedo modificar los equipos ya ingresados





Puedo eliminar los equipos ingresados.

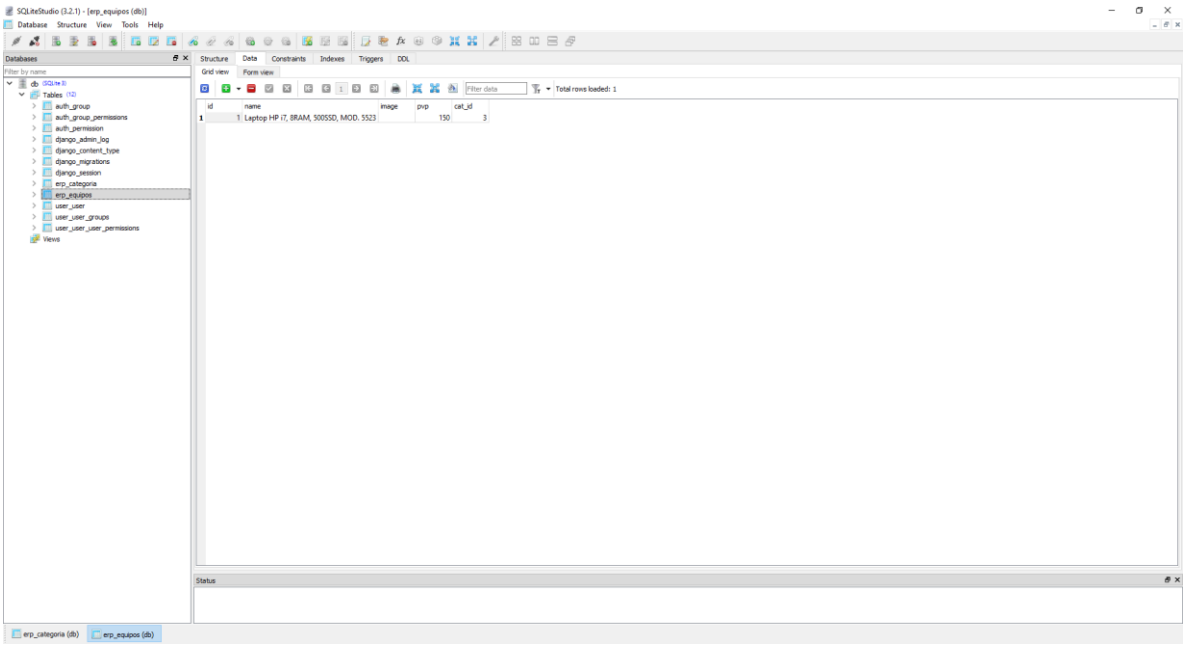
Código Principal

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 = 'Categorias'
        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')
    image = models.ImageField(upload_to='product/%Y/%m/%d', null=True,
blank=True, verbose_name='Imagen')
    pvp = models.DecimalField(default=0.00, max_digits=9,
decimal_places=2, verbose_name='Precio de venta')

    def __str__(self):
        return self.name

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

Los modelos definen las tablas de mi base de datos. Por la facilidad de Django utilicé SQLlite como gestor de base de datos.



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
```

Las vistas me permiten realizar los procesos GET y POST de mi aplicación.

Controlador

```
{% extends 'layout.html' %}
{% load widget_tweaks %}
{% block head %}
    {% block head_form %}

    {% endblock %}
{% endblock %}
{% block content %}
    <form method="post" action="." enctype="multipart/form-data">
        <div class="card card-green">
            <div class="card-header">
                <h3 class="card-title">
                    {% if action == 'add' %}
                        <i class="fas fa-plus"></i>
                    {% else %}
                        <i class="fas fa-edit"></i>
                    {% endif %}

                    {{ title }}
                </h3>
            </div>
            <div class="card-body">
                {% csrf_token %}
                <input type="hidden" name="action" value="{{ action }}">
                {% if form.errors %}
                    <div class="alert alert-danger alert-dismissible">
                        <button type="button" class="close" data-
dismiss="alert" aria-hidden="true">×</button>
                        <h5><i class="icon fas fa-ban"></i> Ha ocurrido
un error al querer guardar el registro</h5>
                        <ul>
                            {% for field in form %}
                                {% for error in field.errors %}
                                    <li>{{ error }}</li>
                                {% endfor %}
                            {% endfor %}
                        </ul>
                    </div>
                {% endif %}
                {% for field in form.visible_fields %}
                    <div class="form-group">
                        <label for="email">{{ field.label }}:</label>
                        {{ field|add_class:'form-
control'|attr:'autocomplete:off' }}
                    </div>
                {% endfor %}
            </div>
            <div class="card-footer">
                <button type="submit" class="btn btn-primary btn-flat">
                    <i class="fas fa-save"></i> Guardar registro
                </button>
                <a href="{{ list_url }}" class="btn btn-danger btn-flat">
                    <i class="fas fa-times"></i> Cancelar
                </a>
            </div>
```

```
        </div>
    </form>
    <script>
        {% if form.errors %}
            var errors = '';
            {% for field in form %}
                {% for error in field.errors %}
                    errors += '{{ error }}\n';
                {% endfor %}
            {% endfor %}
            {% for error in form.non_field_errors %}
                errors += '{{ error }}\n';
            {% endfor %}
            Swal.fire({
                title: 'Error!',
                text: errors,
                icon: 'error'
            });
        {% endif %}

        $('form').on('submit', function (e) {
            e.preventDefault();
            var parameters = new FormData(this);

            submit_with_ajax(window.location.pathname, 'Notificación',
'¿Estas seguro de realizar la siguiente acción?', parameters, function ()
{
                location.href = '{{ list_url }}';

            });
        });
    </script>
{% endblock %}
```

```
{% extends 'layout.html' %}
{% load static %}
{% block head %}
    <link rel="stylesheet" href="{% static 'lib/datatables-
1.10.20/css/dataTables.bootstrap4.min.css' %}"/>
    <link rel="stylesheet"
          href="{% static 'lib/datatables-1.10.20/plugins/responsive-
2.2.3/css/responsive.bootstrap4.min.css' %}"/>
    <script src="{% static 'lib/datatables-
1.10.20/js/jquery.dataTables.js' %}"></script>
    <script src="{% static 'lib/datatables-
1.10.20/js/dataTables.bootstrap4.min.js' %}"></script>
    <script src="{% static 'lib/datatables-1.10.20/plugins/responsive-
2.2.3/js/dataTables.responsive.min.js' %}"></script>
    <script src="{% static 'lib/datatables-1.10.20/plugins/responsive-
2.2.3/js/responsive.bootstrap4.min.js' %}"></script>

    {% block head_list %}

    {% endblock %}
```

```
{% endblock %}

{% block content %}
    <div class="card  card-green">
        <div class="card-header">
            <h3 class="card-title">
                <i class="fas fa-search"></i>
                {{ title }}
            </h3>
        </div>
        <div class="card-body">
            <table class="table table-bordered" id="data">
                <thead>
                {% block columns %}

                {% endblock %}
                </thead>
                <tbody>
                {% block rows %}

                {% endblock %}
                </tbody>
            </table>
        </div>
        <div class="card-footer">
            {% block buttons_list %}
                <a href="{{ create_url }}" class="btn btn-primary btn-
flat btnTest">
                    <i class="fas fa-plus"></i> Nuevo registro
                </a>
                <a href="{{ list_url }}" class="btn btn-success btn-
flat">
                    <i class="fas fa-sync"></i> Actualizar
                </a>
            {% endblock %}
        </div>
    </div>
{% endblock %}

{% block javascript %}
    <script type="application/javascript">
        $(function () {
            $('#data').DataTable({
                responsive: true,
                autoWidth: false
            });

            $('.btnTest').on('click', function () {
                $.ajax({
                    url: '{% url 'erp:category_list' %}',
                    type: 'POST',
                    data: {id: 1},
                    dataType: 'json'
                }).done(function (data) {
                    console.log(data);
                }).fail(function (jqXHR, textStatus, errorThrown) {
```

```
                    alert(textStatus + ': ' + errorThrown);
                }).always(function (data) {

                });
            });
        });
    </script>
{% endblock %}
```

En este caso los templates hacen la vista y le facilitan al usuario su trabajo.