*** PARTE I ***
Configuracion de Django

1. Crea un directorio, kresky es solo el nombre de una empresa
   mkdir karin

2. Entra en el

   cd karin

3. Crear el virtual enviroment

   pipenv shell

4. Al activarlo se crea un archivo "Pipfile"
   que puedes chequear con "ls"

5. pipenv install django==2.2

6. $ django-admin startproject kresky_project. (No te olives del punto)

7. python manage.py migrate

8. Crear la App

  $ python manage.py startapp website

9. Registra la App en settings.
   INSTALLED_APPS=[
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'website', #NEW
]

ERROR Las últimas versiones de django no están importando os
si no "from pathlib import Path" es posible que tengas
que importar os, por eso en este manual instalé django==2.2

10. Crea un urls.py para la app (solo copia y pega)

11. modifica el urls.py del poryecto general(no de la app)
from django.contrib import admin
from django.urls import path, include #NEW

urlpatterns = [
    path('admin/', admin.site.urls),
    path('',include('website.urls')), #NEW
]

12. Crea un Template en website por ejemplo:
    mkdir templates <---- En la carpeta de la app(website)
    home.html --> <h1>Hello App<h1>

13. Ahora ve a la urls.py de tu App
from django.urls import path
from . import views

```
urlpatterns = [
    path('', views.home, name="home"),
]

14. Ve a views.py
from django.shortcuts import render

def home(request):
    return render(request, 'home.html', {})



15. GUARDAR TODOS LOS ARCHIVOS, EMACS NO TE AVISA POR DEFECTO
   Para chequear ve al menú buffer

**PART II** static file

add a settings.py

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

]

16. Agregar tag {% load static %} a home.html

17. """Crear  static file """ en el el directorio raiz
      e inmediatamente crear dentro de el la carpeta website
      donde vas a pegar archivos y carpetas como:
       *css *images *fonts
       Y los siguientes archivos html en la carpeta templates
      *index.html *contact.html
       Generalmente vas a tener que cambiar el nombre de index.html
       por home.html que es el que tienes al principio cuando hiciste el
       proyecto



18.  A tener en cuenta cuando use plantillas:
     1. Los errores que muestre por consola(si son muchos no vale la pena)
     2. Que no tenga mucho php
             3. Los ajax con los que ya viene la página
             4. Si vas a usar tu herramienta regex fijate si es img ó images
                  y claro si realamente cubres todas las extenciones de imagenes

             5. Que vaya entres comillas simples ' ' lo que va después de static
                Cierra Bien las Comillas

          CSS <link rel="stylesheet" href="{% static 'website/css/styles.css' %}">

                IMG <img src="{% static 'website/img/perfil.png' %}" alt="">

                JS <script src="{% static 'website/js/main.js' %}"></script>
```

*** Parte III *** Contact Page (como insert to object)(email mira el video de django dentist)

1. Crear el modelo models.py

```
class Contact(models.Model):
    name = models.CharField(max_length=100)
    cargo = models.CharField(max_length=50)
    email = models.EmailField()
    message = models.TextField()
    def __str__(self):
        template = '{0.name} {0.cargo} {0.email}'
        return template.format(self)
```

```
makemigrations && migrate
```

1.2 home.html (the contact form)
OJO: Es posible que tengas que cambiar el nombre del form por que muchos templates que descargas tienen su formulario ya relacionado a un ajax a travez de, ej: main.js

```
 <form action="" method="post" role="form" class="contactFormu">
       {% csrf_token %} <-----------------------------------
                <div class="form-group">
                  <input type="text" name="name"
                             class="form-control" id="name"
                             placeholder="Your Name" data-rule="minlen:4"
                             data-msg="Please enter at least 4 chars"
                   />
                  <div class="validation"></div>
                </div>
                <div class="form-group">
                  <input type="email" class="form-control"
                             name="email" id="email"
                             placeholder="Your Email"
                             data-rule="email"
                             data-msg="Please enter a valid email" />
                  <div class="validation"></div>
                </div>
                <div class="form-group">
                  <input type="text" class="form-control"
                              name="subject" id="subject"
                              placeholder="Subject" data-rule="minlen:4"
                              data-msg="Please enter at least 8 chars of subject"
                   />
                  <div class="validation"></div>
                </div>
                <div class="form-group">
                  <textarea class="form-control" name="message"
                                 rows="5" data-rule="required"
                                 data-msg="Please write something for us"
                                 placeholder="Message">
                  </textarea>
                  <div class="validation"></div>
                </div>

                <div id="sendmessage">Your message has been sent. Thank you!</div>
```

```
                <div id="errormessage"></div>

                <div class="text-center">
          <button type="submit" title="Send Message">Send Message
                            </button>
                </div>
              </form>
            </div
```

2. urls.py
   Generalmente se haría una contact.html en urls (en este caso no por ser onepage)

```
from django.urls import path
from . import views

urlpatterns = [
    path(' ', views.home, name="home"),
    path('contact.html', views.home, name="contact"),
]
```

3. Views.py (Esto para insertatr en objeto, si quieres enviar correo ve mas abajo)

```
from django.shortcuts import render
from .models import Contact  <----- Tienes que llamar al modelo que vas a usar

def home(request):
    if request.method =="POST":
        name_form = request.POST['namef']
        email_form = request.POST['emailf']
        subject_form = request.POST['subjectf']
        message_form = request.POST['messagef']
        #Insertar el objeto
        Contact.objects.create(name=name_form,
                                            email=email_form,
                                            subject=subject_form,
                                             message=message_form)

        return render(request, 'home.html',{})

    else:

        return render(request, 'home.html',{})

    return render(request, 'home.html', {})
```

3.5 Views.py(email)

Pero primero la confiuración de settings.py

```
#email settings
EMAIL_HOST ='smtp.gmail.com'
EMAIL_PORT = 587
EMAIL_HOST_USER = 'ricardo9285@gmail.com'
EMAIL_HOST_PASSWORD = '#a**********%'
EMAIL_USE_TLS = True
#EMAIL_USE_SSS = False
```

```
def home(request):

    if request.method =="POST" and request.is_ajax():
        name_form = request.POST['name_form']
        email_form = request.POST['email_form']
        message_form = request.POST['message_form']

        # send an email

        send_mail(name_form ,message_form,email_form,['ricardo9285@gmail.com'])

        return JsonResponse({"name_form":name_form}, status=200)
        return render(request, 'home.html', {"name_form" : name_form})

    else:
        return render(request, 'home.html',{})

    return render(request, 'home.html',{})


*** PARTE OPCIONAL ***  *Utilización de AJAX*  → home.html
<!--===== CONTACT =====-->
<section class="contact section" id="contact">

<h2 class="section-title">Contact</h2>

<div class="contact__container bd-grid">
<form id="contact-form"
      role="form"
      action="{% url 'home' %}"
      method="POST"
      class="contact__form"
>

{% csrf_token %}

<input type="text" name="namef" placeholder="Name" class="contact__input">

<input type="text" name="emailf" placeholder="Email" class="contact__input">

<textarea name="messagef" id="" cols="0" rows="10" class="contact__input">
</textarea>

<input type="submit" value="Enviar" class="contact__button button">

</form>

</div>
</section>
```

{% block javascript %}

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js">
</script>

<script>
$(document).ready(function () {
// catch the form's submit event
$('#contact-form').submit(function () {
// create an AJAX call
$.ajax({
data: $(this).serialize(), // get the form data
type: $(this).attr('method'), // GET or POST
url: "{% url 'home' %}",
success: function (response) {
alert("Gracias " +response.name_form +" Recibimos tu email, responderemos
pronto!");
clearMe()
//alert("Registro Agregado")
},
error: function (response) {
// alert the error if any error occured
alert(response.responseJSON.errors);
console.log(response.responseJSON.errors)
}
});
return false;
});
})
function clearMe() {
document.getElementById("contact-form").reset();
}
</script>
```

{% endblock javascript %}

Views.py

```
from .models import Contact
from django.http import JsonResponse

def home(request):
    if request.method =="POST":
    name_form = request.POST['namef']
    email_form = request.POST['emailf']
    message_form = request.POST['messagef']
```

```
#Insertar el objeto
    Contact.objects.create(name=name_form,email=email_form,message=message_form)
    return JsonResponse({"name_form":name_form}, status=200)
    return render(request, 'home.html', {"name_form":name_form})
else:
   return render(request, 'home.html',{})

return render(request, 'home.html', {})




4.***PARTE IV ****
  *POSGRESQL*

  Ya que estas haciendo modelos debes configurar postgresql antes de que te de
problemas
  por que heroku no soporta sqlite3

  1. pipenv install psycopg2-binary



  2.  Vas a tener que configurar settings.py (Base de Datos)
      Por supuesto borra la otra de sqlite3

  DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'karin_db',
        'USER': 'karin_user',
        'PASSWORD': '123456',
        'HOST': 'localhost',
        'PORT': '5432',
    }
   }

  3. Crear la bd y usuario:

     mint:~$ sudo -u postgres psql <-- te va a pedir contraseña del sistema linux

     postgres=# create database karin_db;
     CREATE DATABASE
     postgres=#  \c  karin_db;
     You are now connected to database "karin_db" as user "postgres".
     karin_db=# CREATE USER karin_user WITH PASSWORD '123456';
     CREATE ROLE


  4. Hacer migrate
     (karin) ricardo@ricardo-mint:~/DjangoProjects/karin$ python manage.py migrate
     (Si todo sale bien te debe dar el siguiente mensaje)
     Operations to perform:
     Apply all migrations: admin, auth, contenttypes, sessions, website
     Running migrations:
     Applying contenttypes.0001_initial... OK
```

```
 *** PARTE V ***
  *Configuración Admin*

  5. Retomando el modelo contact

     1.Crea un super usuario para poder manipular los modelos:

       python manage.py createsuperuser

     2.Tienes que registrarlo en admin.py

       from django.contrib import admin
       from .models import Contact

       admin.site.register(Contact)

     3. python manage.py runserver


*** PARTE VI ***
Continuación Contact Page
En este caso action apunta a espacio por que
en urls.py esta configurado de esa forma,
por favor coloca method="post" muchos templates
no lo traen por defecto

  <form action="" method="post"  role="form" class="contactFormu">
   {% csrf_token %}
     <div class="contact__inputs">
       <input type="text" name="namef" placeholder="Name"  class="contact__input">
       <input type="text" name="emailf" placeholder="Email"
              class="contact__input">
     </div>

     <textarea name="messagef" id="" cols="0" rows="10"
         class="contact__input"></textarea>


          <input type="submit" value="Enviar" class="contact__button">
  </form>

urlpatterns = [
    path('', views.home, name="home"),
]

*OTRO EJEMPLO con otro action*
<h1>Create a Post </h1>
<form action="createpost" method="POST">

{% csrf_token %}

Title: <input type="text" name="title"/><br/>
Content: <br/>
<textarea cols="35" rows="8" name="content">
</textarea><br/>
```

```
<input type="submit" value="Ingresar"/>

</form>

urls.py
path('createpost', views.createpost),


*** PARTE VII ***

Extends Base.html For Navbars

1. Vas a crear y guardar en templates
   un archivo llamado base.html.
   Aquí es donde se quedara insertado el
   código de otras páginas

   {% block content %}

   {% endblock %}


En este ejemplo se coloca primero el Header
{% block content %}
{% endblock %}
El Footer

Ojo : Esto va a depender de la plantilla
debes revisar la pág con inspect element

{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">

<!-- ===== CSS ===== -->
<link rel="stylesheet" href="{% static 'website/assets/css/styles.css' %}">

<!-- ===== BOX ICONS ===== -->
<link href='https://cdn.jsdelivr.net/npm/boxicons@2.0.5/css/boxicons.min.css'
rel='stylesheet'>

<title>Karin Peña</title>
</head>
<body>
<!--===== HEADER =====-->
<header class="l-header">
<nav class="nav bd-grid">
<div>
<a href="#" class="nav__logo">Karin Peña</a>
</div>

<div class="nav__menu" id="nav-menu">
<ul class="nav__list">
<li class="nav__item"><a href="#home" class="nav__link active">Home</a></li>
<li class="nav__item"><a href="#about" class="nav__link">About</a></li>
<li class="nav__item"><a href="#skills" class="nav__link">Skills</a></li>
```

```
<li class="nav__item"><a href="#portfolio" class="nav__link">Portfolio</a></li>
<li class="nav__item"><a href="#contact" class="nav__link">Contact</a></li>
</ul>
</div>

<div class="nav__toggle" id="nav-toggle">
<i class='bx bx-menu'></i>
</div>
</nav>
</header>

{% block content %}

{% endblock %}

<!--===== FOOTER =====-->
<footer class="footer section">
<div class="footer__container bd-grid">
<div class="footer__data">
<h2 class="footer__title">Karin Peña</h2>
<p class="footer__text">Soy Karin Peña y esta es mi página web</p>
</div>

<div class="footer__data">
<h2 class="footer__title">EXPLORE</h2>
<ul>
<li><a href="#home" class="footer__link">Home</a></li>
<li><a href="#about" class="footer__link">About</a></li>
<li><a href="#skills" class="footer__link">Skills</a></li>
<li><a href="#portfolio" class="footer__link">Portfolio</a></li>
<li><a href="#Contact" class="footer__link">Contact</a></li>
</ul>
</div>
<div class="footer__data">
<h2 class="footer__title">FOLLOW</h2>
<a href="#" class="footer__social"><i class='bx bxl-facebook' ></i></a>
<a href="#" class="footer__social"><i class='bx bxl-instagram' ></i></a>
<a href="#" class="footer__social"><i class='bx bxl-twitter' ></i></a>
</div>
</div>
</footer>

<!--===== SCROLL REVEAL =====-->

<script src="https://unpkg.com/scrollreveal"></script>

<!--===== MAIN JS =====-->
<script src="{% static 'website/assets/js/main.js' %}"></script>
</body>
</html>
```

Después de home.html llamar a extends ‘base.html’

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

{% block content %}
<main class="l-main">
```

```
<!--===== HOME =====-->
<section class="home" id="home">
<div class="home__container bd-grid">
<h1 class="home__title"><span>HO</span><br>LA&hearts;</h1>

<div class="home__scroll">
<a href="#about" class="home__scroll-link"><i class='bx bx-up-arrow-alt'
></i>Información</a>
</div>
<img src="{% static 'website/assets/img/perfilo.png' %}" alt="" class="home__img">
</div>
</section>
<!--===== ABOUT =====-->
<section class="about section" id="about">
</div>

</section>

<!--===== CONTACT =====-->
<section class="contact section" id="contact">
<span class="contact__text">info.mail.com</span>
<h3 class="contact__subtitle">PHONE</h3>
<span class="contact__text">+20 999-999</span>
<h3 class="contact__subtitle">ADRESS</h3>


</section>
</main>

{% endblock %}



** OTRO Ejemplo **
Para la página details.html

{% extends 'base.html' %}
{% load static %}

{% block content %}

<section class="portfolio section" id="portfolio">
<h2 class="section-title">Portfolio</h2>

</section>

{% endblock %}


Configurar views.py

def details(request):
    return render(request, 'details.html', {})

Configurar urls.py

urlpatterns = [
path('', views.home, name="home"),
path('details.html', views.details, name="details"), #new
```

]

Como hacer un tag configurable?? title (el titulo del tag puede ser el que quieras)

<title>{% block title %} Karin Peña {% endblock %}</title>

En details.html

{% load static %}
{% block title %} Details {% endblock %}
{% block content %}

Otro ejemplo base.html meta tag

<meta name="description" content="{% block meta_description %}karin pena
rentahouse{% endblock %}">

details.html
{% block title %} Details {% endblock %}
{% block meta_description %}karin pena details rentahouse{% endblock %}

Cuando hagas un enlace sería:
{% url ‘ contact’ %}

PARTE VIII
Hacer un posting para que el cliente haga sus propias publicaciones
de lo que va a vender.

1. Tienes que preparar un objeto en modelo.py para después
   mostrarlo con un **for** en template

     a.python manage.py **makemigrations**

       python manage.py **migrate**

     b. Registrarlo en admin.py y views.py
        y coloca algunos registros para probar

2. Tienes que acondicionar el html para hacer el for loop

3. Views.py (declarar las variables que vas a lanzar a html)

4. Imagen

views.py
def home(request):
  items = Post.objects.all()

  ----
  return render(request, 'home.html', {'items':items})

home.html
{% for **item** in items %}
  <img src="{% static **item.img** %}" alt="">

   View details

```
    </div>
  {% endfor %}
```
Para url , en este caso no use la palabra url por ser
un enlace directo a internet

```
{% for item in items %}
    <div class="portfolio__img">
    <img src="{% static item.img %}" alt="">
    <div class="portfolio__link">
    <a href="{{item.details}}" class="portfolio__link-name">View details</a>
    </div>
    </div>
{% endfor %}
```