Universidad Nacional de Loja

POSGRADO

Desarrollo de Software en Ambientes Cloud

Maestría en Ingeniería en Software - UNL

Carrera de Ingeniería en Sistemas / Computación


IN THESAVRIS SAPIENTIÆ GLORIFICATIO VITÆ
unl
Universidad
Nacional
de Loja

Bienvenidos!

Ing. Roberth Figueroa Díaz M.Sc.

roberth.figueroa@unl.edu.ec

Profesor

Agenda

1. Tecnologías del lado del Servidor o Backend
2. Instalando Framework Django
   - Instalar Django
   - Identificar comandos claves
   - Realizar un ejemplo con Django
   - Conclusiones y recomendaciones en Django

INSTALACIÓN DJANGO

3.2.3

Virtual Env

virtualenv → Entorno Python Aislado

Virtualenv

Virtualenv is a tool that lets you create an isolated Python environment for your project. It creates an environment that has its own installation directories, that doesn't share dependencies with other `virtualenv` environments (and optionally doesn't access the globally installed dependencies either). You can even configure what version of Python you want to use for each individual environment. It's very much recommended to use `virtualenv` when dealing with Python applications.

Installation

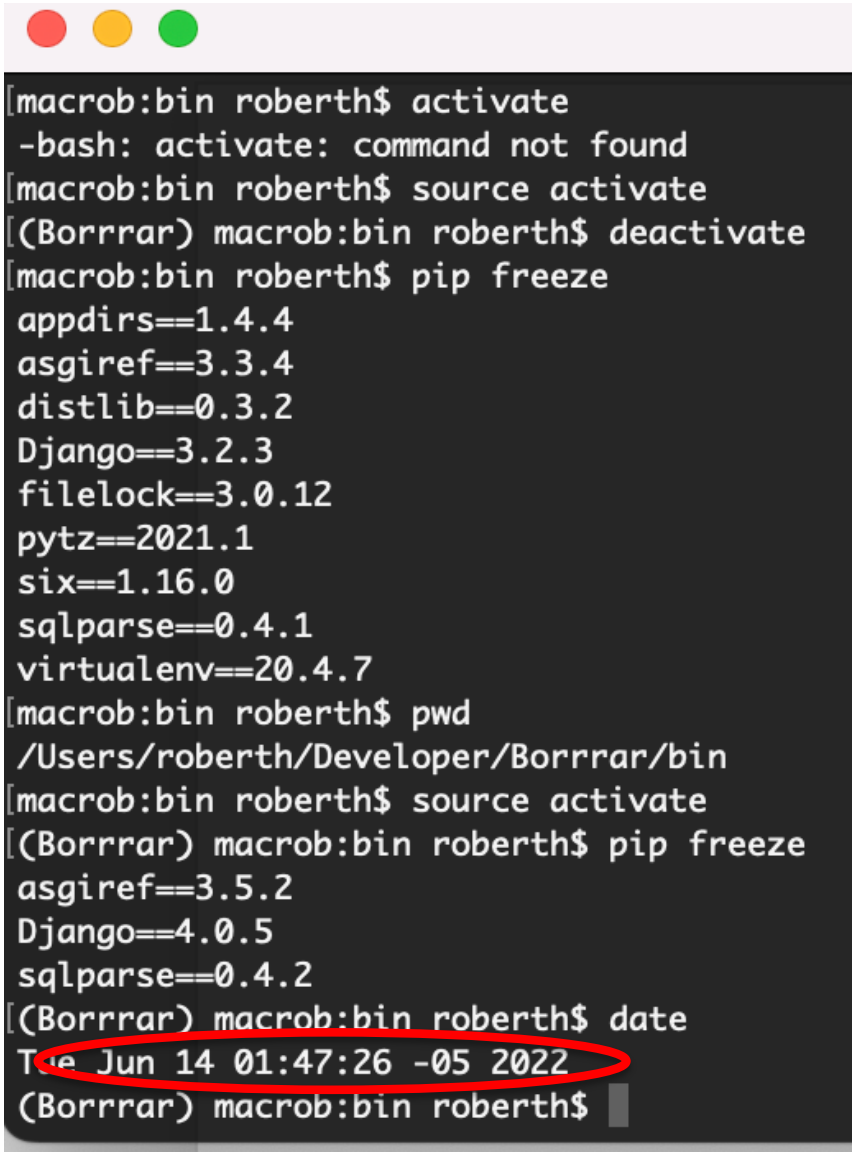
To install `virtualenv` run:

```
pip install virtualenv
```

Virtual Env →entorno virtual

entorno principal administrado por Sistema operativo

entorno virtual llamado Borrrar

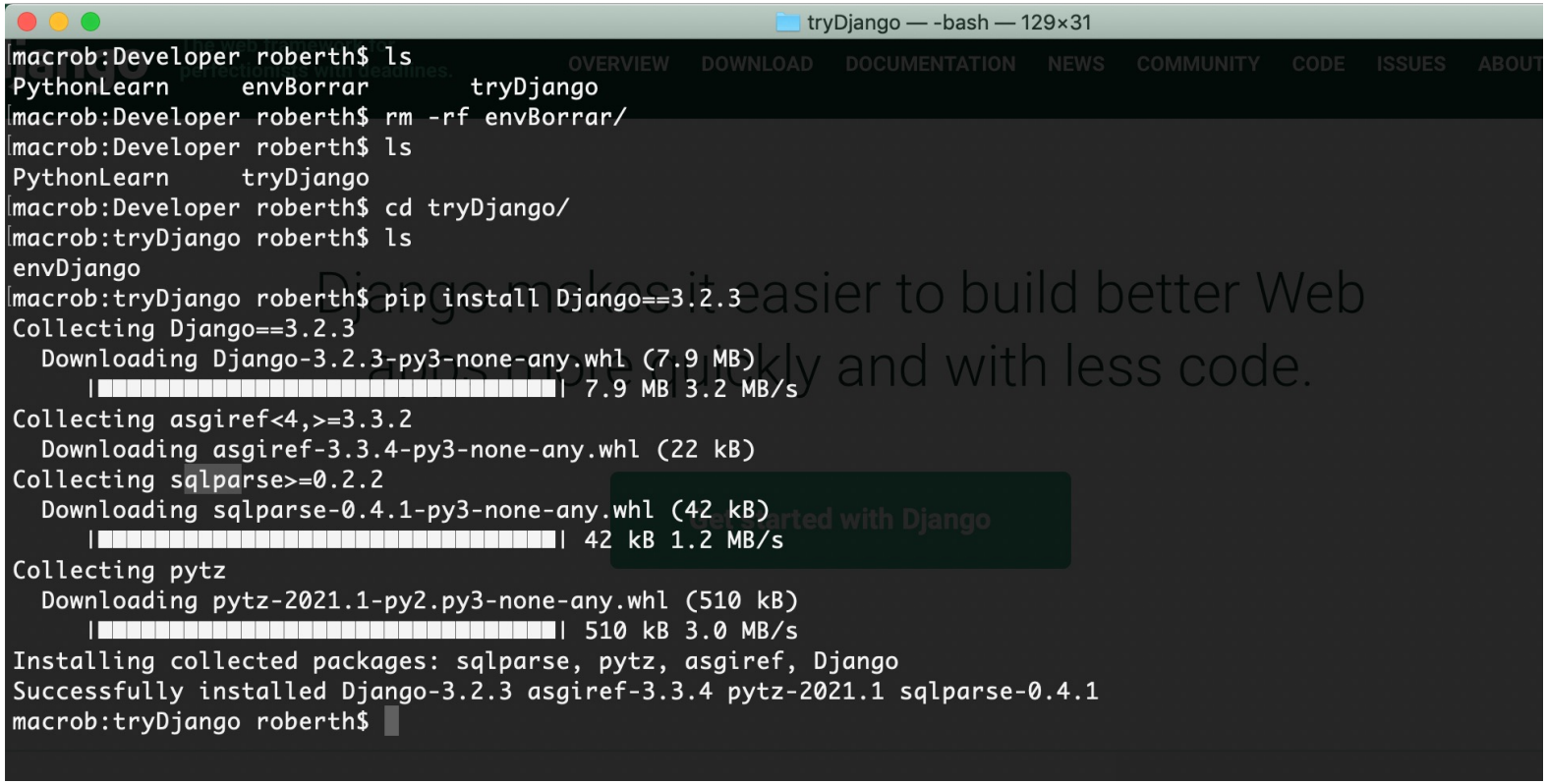


Pasos a seguir

AMBIENTE DESARROLLO

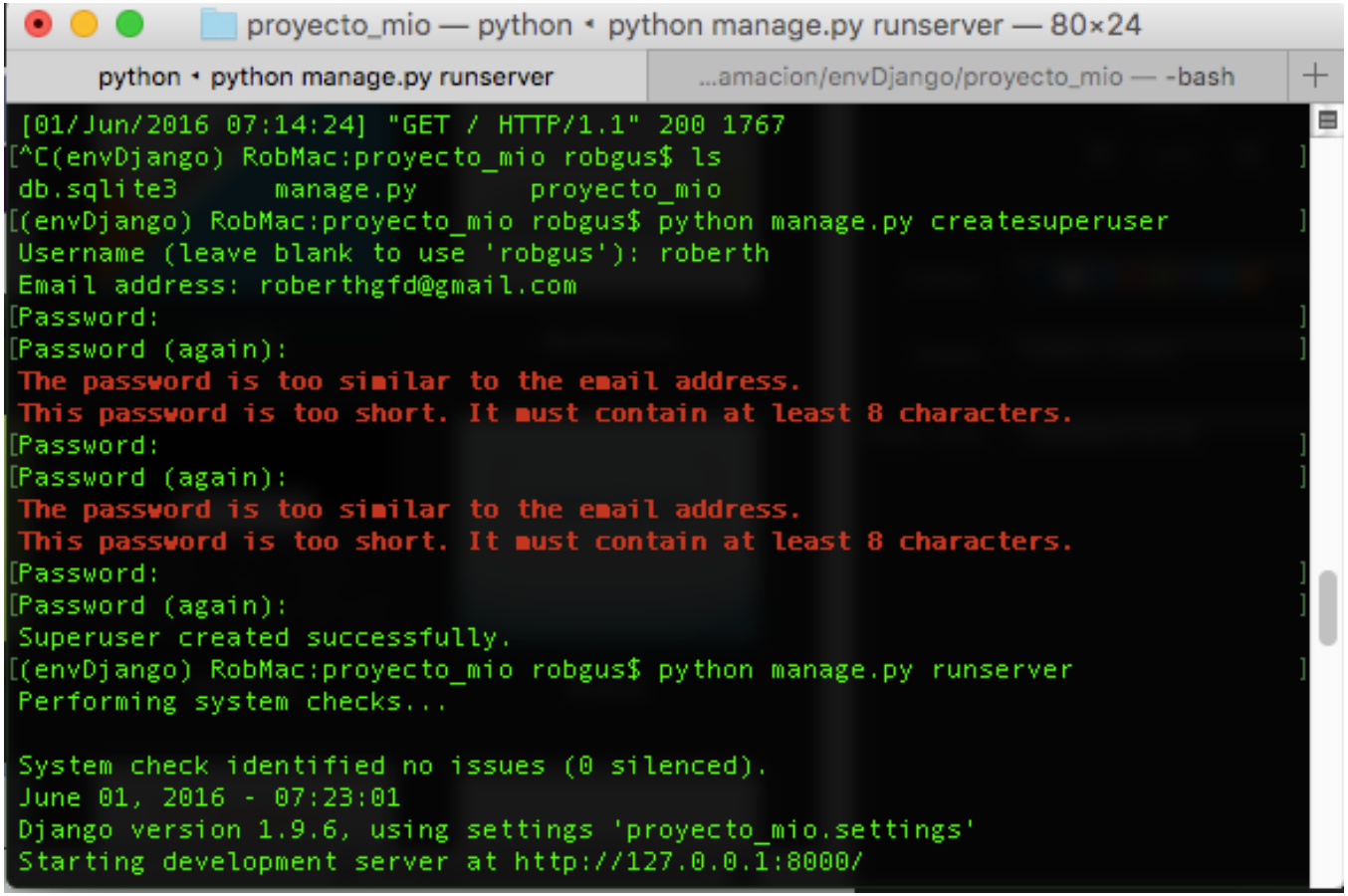
1. Instalando PIP en Python
   - Usar comando → **sudo easy_install pip**
2. Instalando virtual environment
   - Usar comando → **sudo pip install virtualenv**
3. *Crear el entorno virtual deseado*
   - Usar comando → **virtualenv envDjango**
   - O definir la versión de python al entorno → **virtualenv –p python3 .**
   - Se crean carpeta **bin**, **lib** con archivos Python necesarios
4. *Instalar Django*
   - Comando para activar virtualenv → **source bin/activate**
   - Comando para instalar → **pip install Django==3.2.3**
   - Comando para actualizar la version django → **pip install django -- upgrade**
   - Comando para ver la versión de django instalada→ **pip freeze**
5. Crear la aplicación **mi_proyecto** dentro del environment
   - Comando → **django-admin startproject mi_proyecto**
6. Se genera la base de datos (SQlite):
   - Comando → **python manage.py migrate**
7. Se ejecuta el servidor de aplicaciones web:
   - Comando → **python manage.py runserver**

Proceso
Instalación de Django 3.2.3



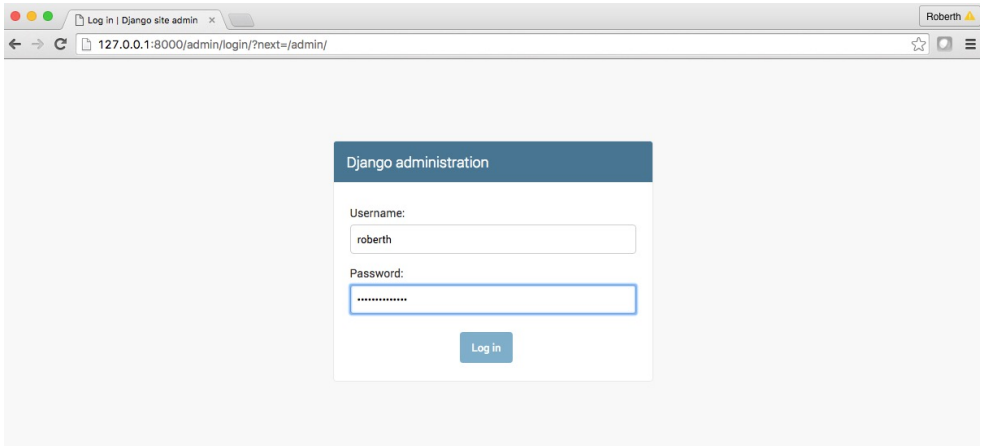
Crear el adminstrador Django

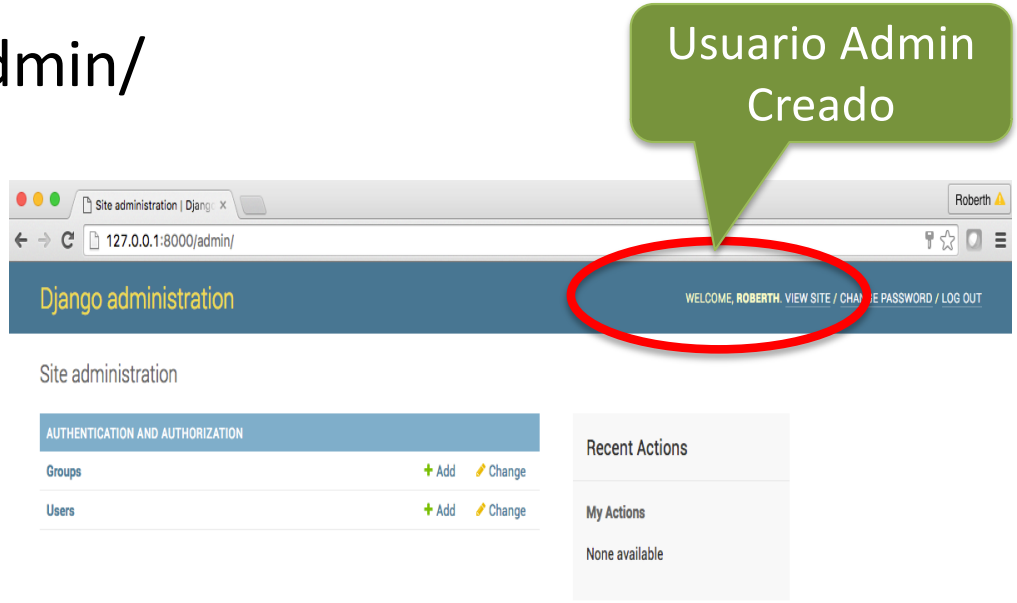
- Crear el usuario
  - Usar comando → **python manage.py createsuperuser**



Crear el adminstrador Django

- Se ejecuta el servidor de aplicaciones :
  - Comando → **python manage.py runserver**
- Ir al navegador a:
  - http://127.0.0.1:8000/admin/

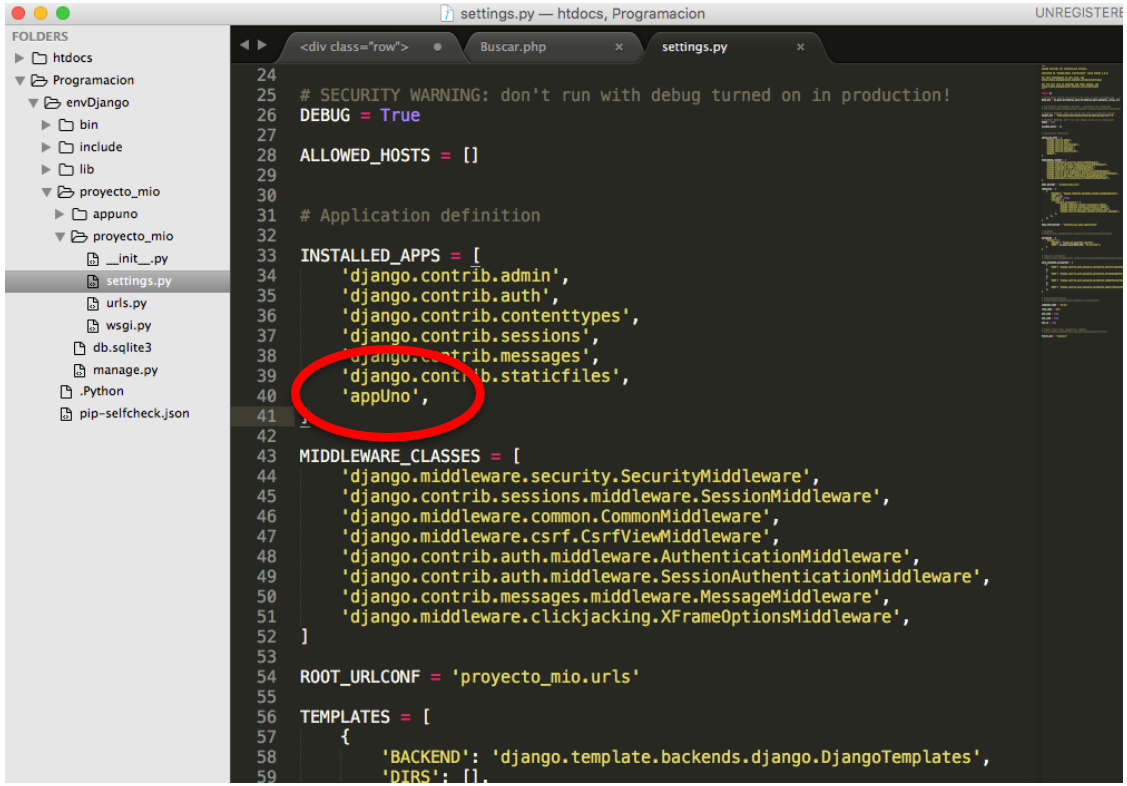




Creación de aplicaciones

- Crear el usuario
  - Usar comando → **python manage.py startapp appUno**
  - Agregar la appUno al settings.py→ **sudo vim** settings.py

settings.py



CREACIÓN DE MÁS APLICACIONES EN DJANGO

Creación de Aplicaciones Django

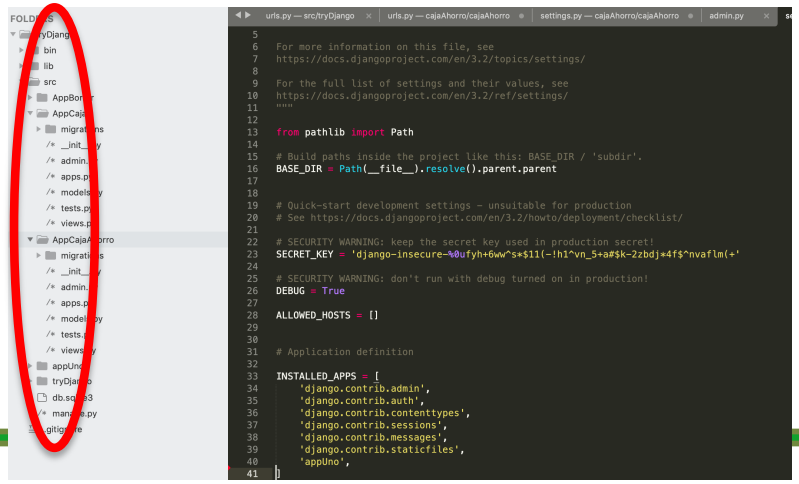
- Ubicarse en el directorio de instalación de Django

En nuestro caso:

→ /Users/roberth/Developer/tryDjango/src

```
src — -bash — 80×24
bin             cajaAhorro      lib             pyvenv.cfg      src
(tryDjango) macrob:tryDjango roberth$ ls
bin             cajaAhorro      lib             pyvenv.cfg      src
(tryDjango) macrob:tryDjango roberth$
(tryDjango) macrob:tryDjango roberth$ cd ..
(tryDjango) macrob:Developer roberth$ ls
PythonLearn     tryDjango
(tryDjango) macrob:Developer roberth$ pwd
/Users/roberth/Developer
(tryDjango) macrob:Developer roberth$ cd tryDjango/
(tryDjango) macrob:tryDjango roberth$ ls
bin             cajaAhorro      lib             pyvenv.cfg      src
(tryDjango) macrob:tryDjango roberth$ cd src/
(tryDjango) macrob:src roberth$ ls
Banco           appUno          db.sqlite3      manage.py       tryDjango
(tryDjango) macrob:src roberth$ python manage.py startapp AppBorrar
(tryDjango) macrob:src roberth$ python manage.py startapp AppCaja
(tryDjango) macrob:src roberth$ python manage.py startapp AppCajaAhorro
(tryDjango) macrob:src roberth$ python manage.py startapp AppCajaAhorro2
(tryDjango) macrob:src roberth$ python manage.py startproject MiProyecto
(tryDjango) macrob:src roberth$ python manage.py startapp AppCajaAhorro20
(tryDjango) macrob:src roberth$ pwd
/Users/roberth/Developer/tryDjango/src
(tryDjango) macrob:src roberth$
```

- Identificar carpetas y archivos principales de cada aplicación



Creación de aplicación con **startapp**

- Crear la aplicación CajaAhorro
  - Usar comando → **python manage.py startapp appCajaAhorro**
  - Agregar la appCajaAhorro al **settings.py** → **sudo vim** settings.py

settings.py

```
31
32  # Application definition
33
34  INSTALLED_APPS = [
35      'django.contrib.admin',
36      'django.contrib.auth',
37      'django.contrib.contenttypes',
38      'django.contrib.sessions',
39      'django.contrib.messages',
40      'django.contrib.staticfiles',
41      'appUno',
42      'CajaAhorro',
43  ]
44
45  MIDDLEWARE = [
46      'django.middleware.security.SecurityM
47      'django.contrib.sessions.middleware.S
48      'django.middleware.common.CommonMiddl
49      'django.middleware.csrf.CsrfViewMiddl
50      'django.contrib.auth.middleware.Authe
51      'django.contrib.messages.middleware.M
52      'django.middleware.clickjacking.XFram
53  ]
54
55  ROOT_URLCONF = 'DemoDjango.urls'
56
57  TEMPLATES = [
58      {
59          'BACKEND': 'django.template.backe
60          'DIRS': [BASE_DIR, "templates"],
61          'APP_DIRS': True,
62          'OPTIONS': {
```

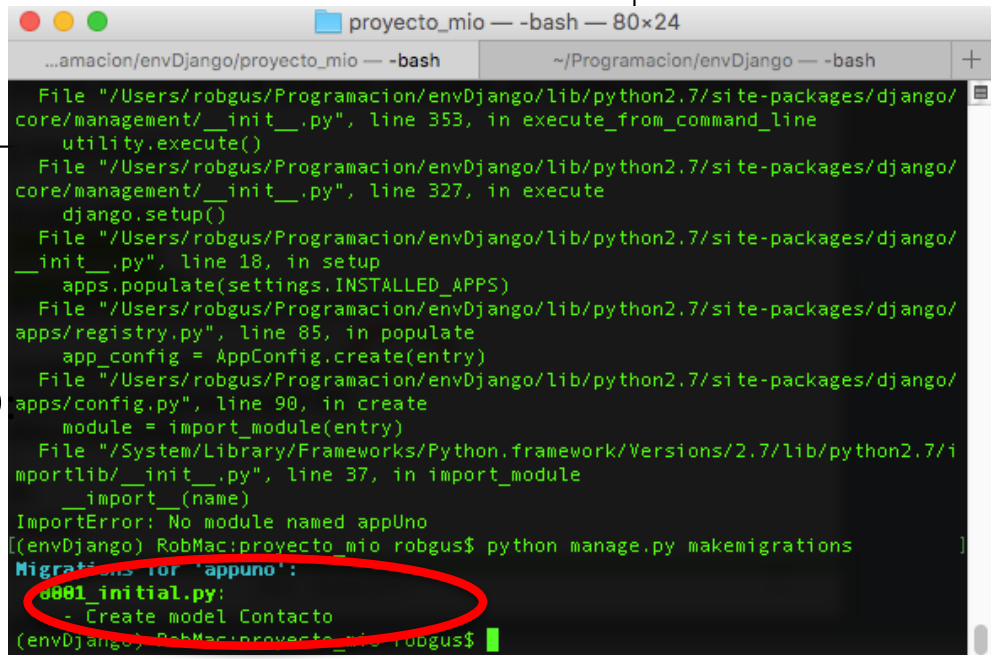
Comando **makemigrations**

- Codificar el **models.py,** con la clase Contacto:

```
from django.db import models

class Contacto(models.Model):

     nombres=models.CharField(max_length=30,blank=True)
     apellidos=models.CharField(max_length=30)
     cedula=models.CharField(max_length=30)
     email=models.EmailField()

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

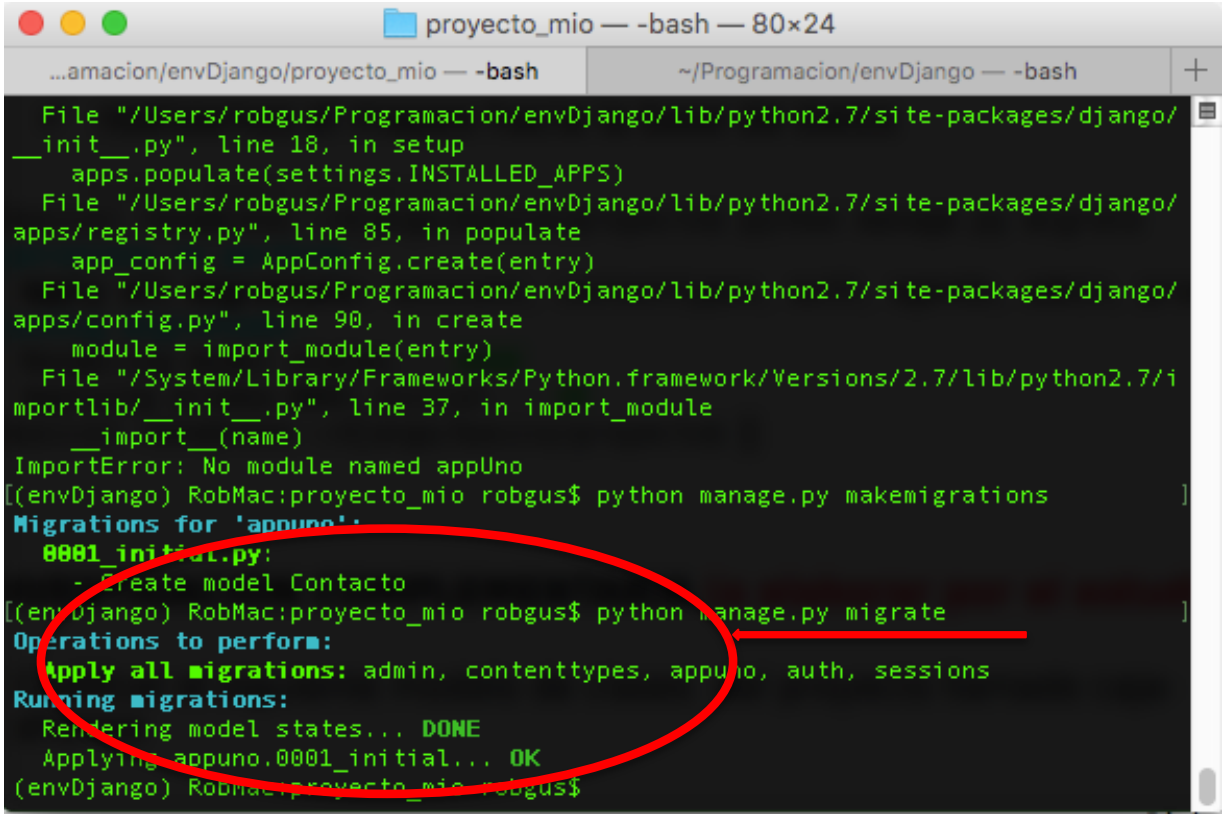
- Aplicar el modelo al proyecto

usar→ **python manage.py makemigrations**



Comando **migrate**

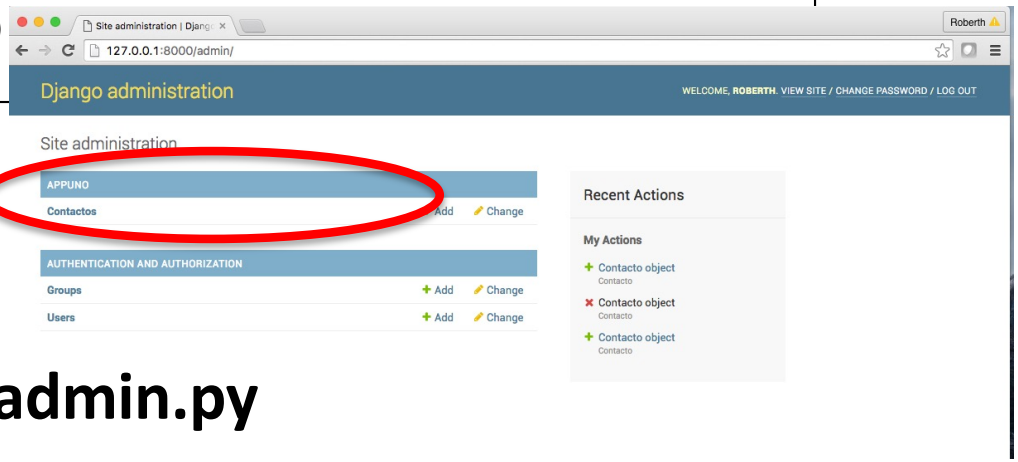
- Mapear hacia la base de datos el modelo
  - usar→ **python manage.py migrate**



Anclar el modelo al **admin.py**

- Realizar cambios al **admin.py**

```
from django.contrib import admin
from .models import Contacto

class AdminContacto(admin.ModelAdmin):
     list_display = ["__str__","nombres","apellidos","email"]
     class Meta(object):
          model = Contacto

admin.site.register(Contacto,AdminContacto)
```



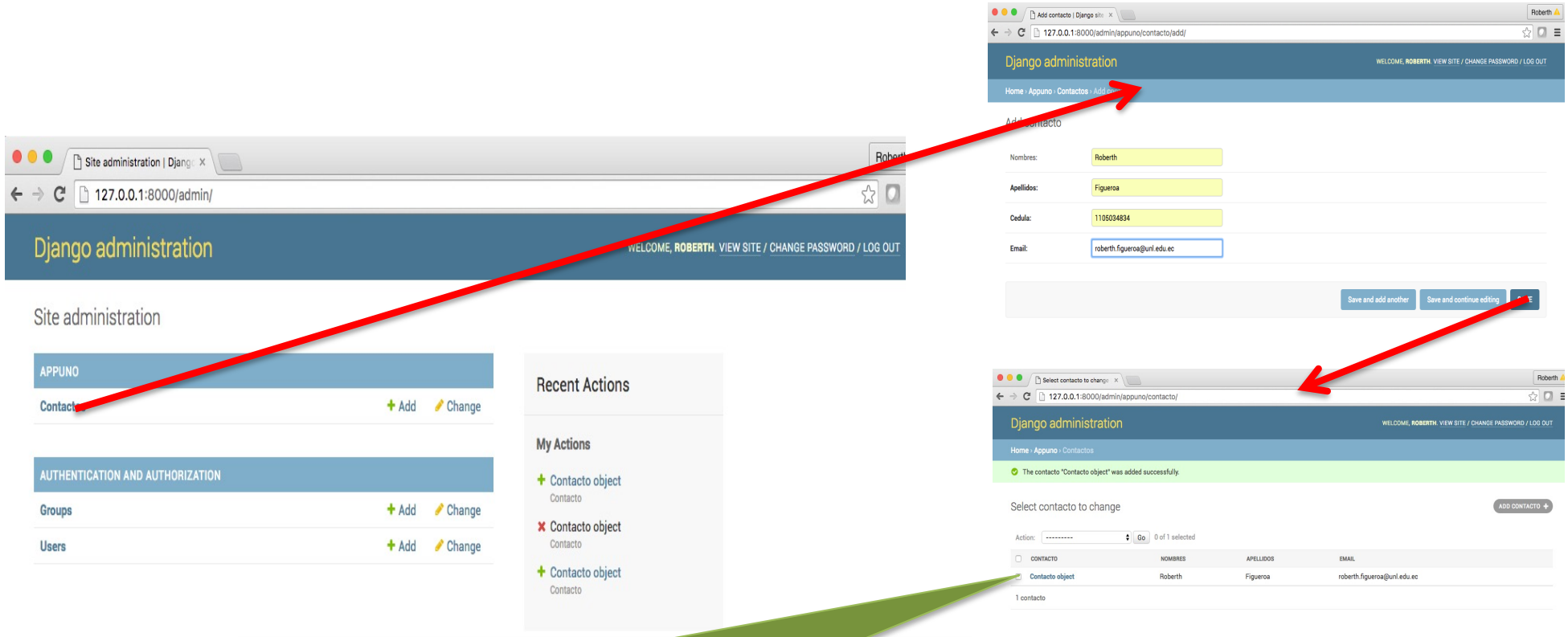
- Aplicar el anclaje al **admin.py**

usar→ **python manage.py migrate**

Levantar el servidor de aplicaciones → **python manage.py runserver**

Resultado luego anclaje

- Pantalla modelo creado en código python.



el __str__ hará de índice en la tabla y el objeto

Uso de Views

Modificar View

- Realizar cambios al **view.py**

```
#from django.http import HttpResponse
from django.shortcuts import render

# Create your views here.

def home_view(request, *args,**kwargs):
     print (args, kwargs)
     print (request)
     print (request.user)
     return render(request, "home.html", {})
     #return HttpResponse("<h1>Hola Bienvenido al Sistema appUno</h1>")

def about_view(request):
     return render(request, "about.html", {})

def contact_view(request,):
     return render(request, "contact.html", {})
```

Habilitar para enviar HTML directamente en el return

```
1  #from django.http import HttpResponse
2  from django.shortcuts import render
3
4  # Create your views here.
5
6  def home_view(request, *args,**kwargs):
7      print (args, kwargs)
8      print (request)
9      print (request.user)
10     return render(request, "home.html", {})
11     #return HttpResponse("<h1>Hola Bienvenido al Sistema appUno</h1>")
12
13 def about_view(request):
14     return render(request, "about.html", {})
15
16 def contact_view(request,):
17     return render(request, "contact.html", {})
18
19 def demo_view(request,):
20     return render(request, "demo.html", {})
```

Agregar un elemento nuevo a la lista urlpatterns en **urls.py**

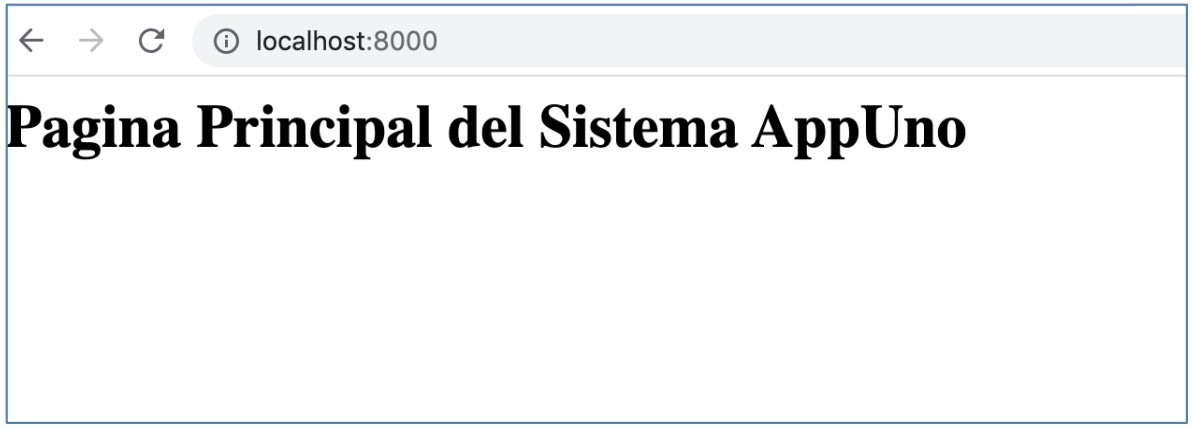
- Agregar la url deseada archivo **urls.py**

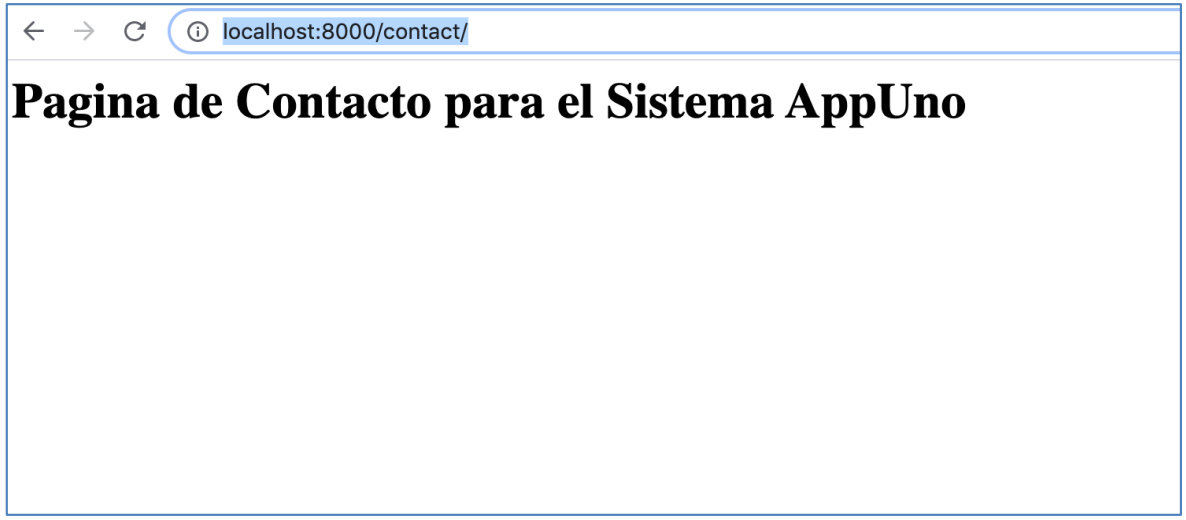
  `path('about/',about_view),`

- **`about_view`** : es la vista donde se ubica la función en **views.py**

```
from django.contrib import admin
from django.urls import path
from appUno.views import home_view, contact_view, about_view

urlpatterns = [
    path('',home_view, name='home'),
    path('about/',about_view),
    path('contact/',contact_view),
    path('admin/', admin.site.urls),
]
```

Acceder a las nuevas url’s





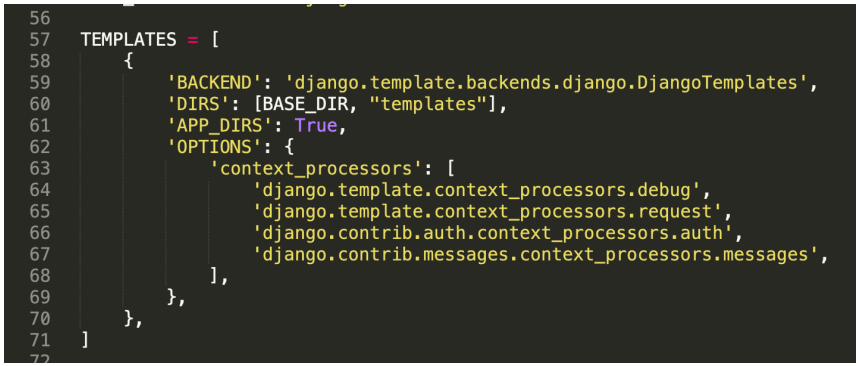
Uso de Templates

Configurar los **templates** en **settings.py**

- Directorio de templates:

  modificar a la ubicación →'DIRS': [ ],

- Quedando así: 'DIRS'

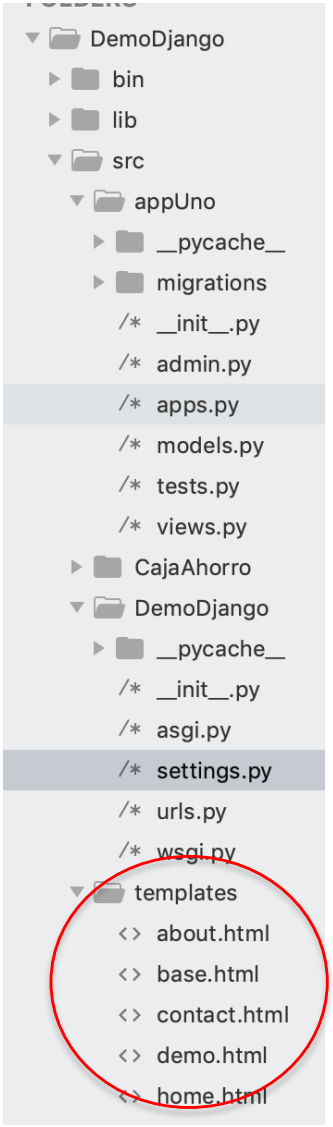


```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR,"templates"],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

Modifique las **views** para cada **templates**

```python
from django.http import HttpResponse
from django.shortcuts import render


def home_view(request):

    return HttpResponse("<h1>Hola Bienvenido Señor@s Estudiantes 5A</h1>")

def about_view(request):
    return render(request, "about.html", {})

def contact_view(request):
    return render(request, "contact.html", {})

def demo_view(request):
    return render(request, "demo.html", {})
```

Jerarquía de carpetas para **templates**




```
12
13  from pathlib import Path
14
15  # Build paths inside the project like this: BASE_DIR / 'subdir'.
16  BASE_DIR = Path(__file__).resolve().parent.parent
17  #/Users/roberth/Developer/DemoDjango/src
18  print(BASE_DIR)
19
20  # Quick-start development settings - unsuitable for production
21  # See https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/
22
23  # SECURITY WARNING: keep the secret key used in production secret!
24  SECRET_KEY = 'django-insecure-!w*p%m(4!*3q6zefyue0k@50&z3583ncweem(v7)3v1&^rgat*'
25
26  # SECURITY WARNING: don't run with debug turned on in production!
27  DEBUG = True
28
29  ALLOWED_HOSTS = []
```

Hay que recordar que la ruta de su proyecto de Django la puede ver mediante la variable **BASE_DIR,** que se ubica en el archivo **settings.py**

Creamos el arhivo html
contact.html

Se guarda el html en el directorio de **templates**:

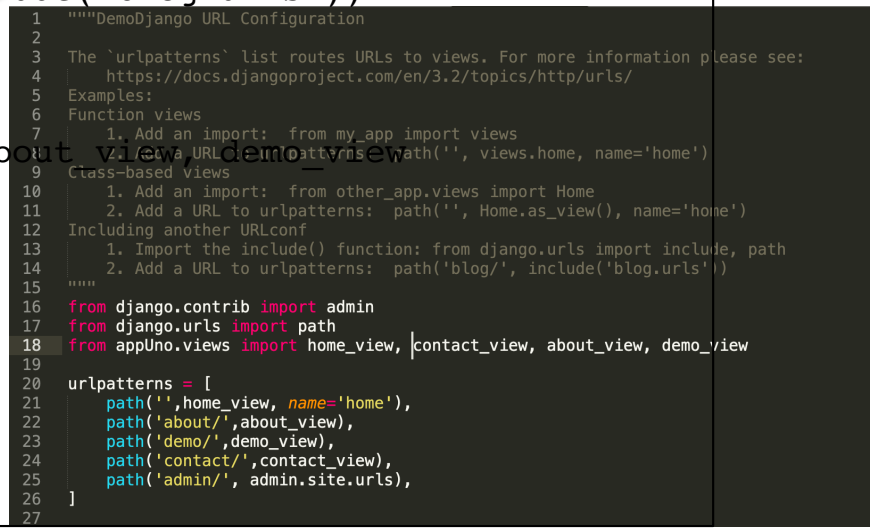
```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8">
5      <title>Conctacto pagina</title>
6  </head>
7  <body>
8  <h1>Pagina de contacto con plantilla DBP</h1>
9  </body>
10 </html>
```

Se crean más archivos html en el directorio de **templates**, según la necesidad

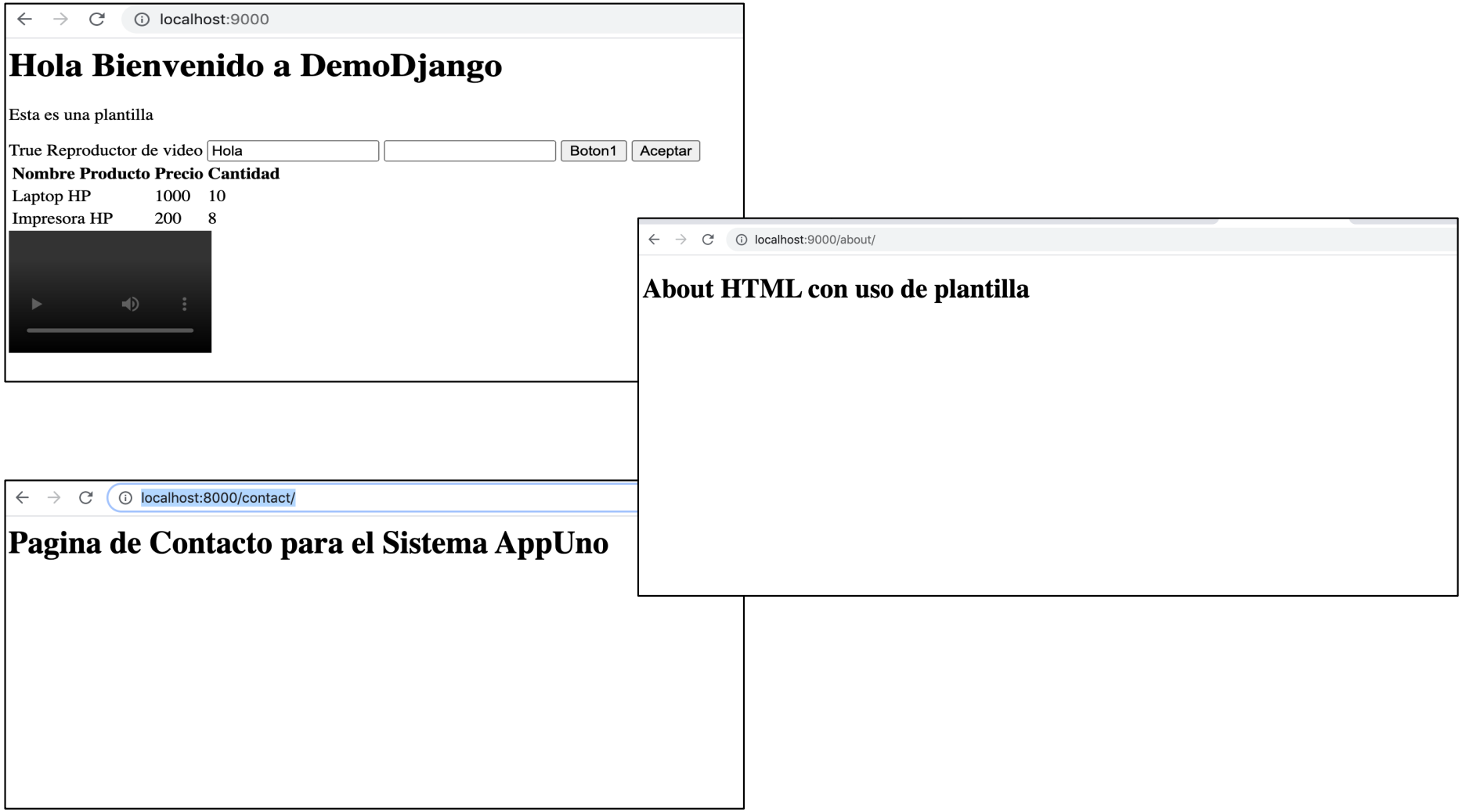
Agregar un elemento nuevo a la lista urlpatterns en **url.py** de ser necesario

```
"""DemoDjango URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/3.2/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
from appUno.views import home_view, contact_view, about_view, demo_view

urlpatterns = [
    path('',home_view, name='home'),
    path('about/',about_view),
    path('demo/',demo_view),
    path('contact/',contact_view),
    path('admin/', admin.site.urls),
]
```

importamos las vistas



Ver el resultado de **uso de templates**



Agregando código

Django versión 3.2.3

Agregar código python en templates

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

{% block content %}


<h1>Hola</h1>
<p>Esto es una plantilla </p>

{% endblock %}

{% block content %}

{% endblock %}
```

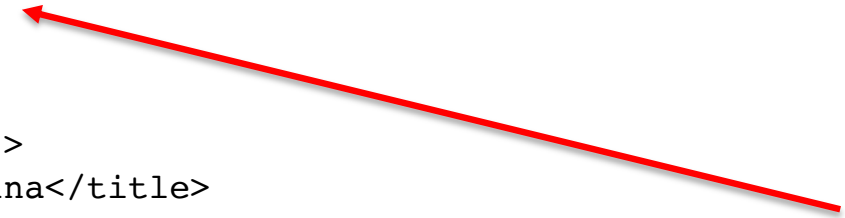
Herencia de plantilla

Código para Verificar el usuario activo:

```
{{request.user.is_authenticated}}
```

Modicando **contact.html**

```
{% include "base.html" %}
<!DOCTYPE html>
<html>
<head>
     <meta charset="utf-8">
     <title>Conctacto pagina</title>
</head>
<body>
<h1>Pagina de contacto con plantilla DBP</h1>

  {{request.user.is_authenticated}}

</body>
</html>
```

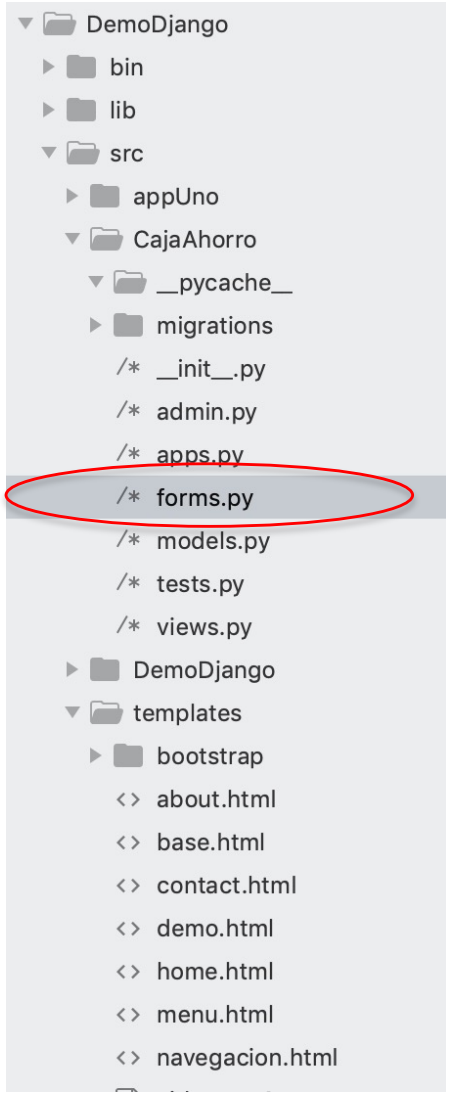


Se importa base.html

Agregando formularios

Django versión 3.2.3

Creamos el archivo **forms.py** dentro de la app



Definir el formulario **forms.py**

```
from django import forms
from .models import Contacto

class Formulario(forms.ModelForm):
     class Meta:
          model = Contacto
          fields=["nombres", "apellidos","cedula","email"]
```

```
1  from django import forms
2  from .models import Contacto
3
4  class Formulario(forms.ModelForm):
5      class Meta:
6          model = Contacto
7          fields=["nombres", "apellidos","cedula","email"]
```

Modificamos el archivo **view.py**

```
from django.shortcuts import render
from django.shortcuts import redirect
from .forms import Formulario
from .models import Contacto

def home_view(request):
     f = Formulario(request.POST or None)
     if request.method == 'POST':
          if f.is_valid():
               datos = f.cleaned_data
               c = Contacto()
               c.nombres = datos.get("nombres")
               c.apellidos = datos.get("apellidos")
               c.cedula = datos.get("cedula")
               c.email = datos.get("email")
               if c.save() != True:
                    print('Imprimo en pantalla y guardo data en BD')
                    print(f.cleaned_data)
                    return redirect(home_view)
     context = {
          "form":f,
     }
     return render(request,"home.html",context)

def contact_view(request):
     return render(request,"contact.html",{})

def about_view(request):
     return render(request,"about.html",{})
```

f es un objeto de la Petición POST

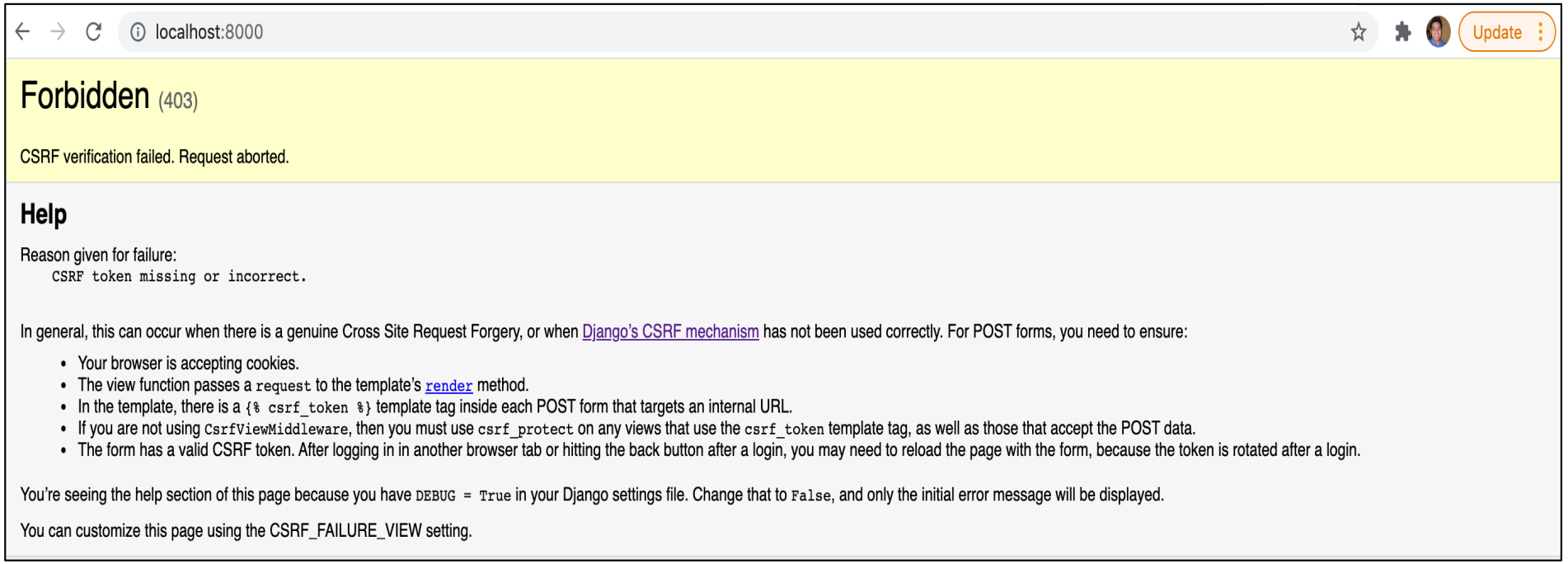
Validar los campos de entradas:

- Agregamos los requisitos de validación de POST
  - **request.POST**
- Agregamos None para que no se visualice los mensajes de
  - **request.POST or None**

El **view.py** modificado

```python
from django.shortcuts import render
from django.shortcuts import redirect
from .forms import Formulario
from .models import Contacto

def home_view(request):
    f = Formulario(request.POST or None)
    if request.method == 'POST':
        if f.is_valid():
            datos = f.cleaned_data
            c = Contacto()
            c.nombres = datos.get("nombres")
            c.apellidos = datos.get("apellidos")
            c.cedula = datos.get("cedula")
            c.email = datos.get("email")
            if c.save() != True:
                print('Imprimo en pantalla y guardo data en BD')
                print(f.cleaned_data)
                return redirect(home_view)
    context = {
        "form":f,
    }
    return render(request,"home.html",context)

def contact_view(request):
    return render(request,"contact.html",{})

def about_view(request):
    return render(request,"about.html",{})

def demo_view(request):
    return render(request,"demo.html",{})
```

Vulnerabilidad Cross Site Request Forgery ***CSRF*** en peticiones **POST**



Modificamos la plantilla **home.html**

- **{{form}}** es la variable context enviada por view a la plantilla.
- **form.as_p** → muestra el formulario en párrafo en html usando la etiqueta <p> por defecto.

```
<h1>Pagina de Inicio</h1>
<form method="post" action=""> {% csrf_token %}
    {{form.as_p}}
    <input type="submit" value="Aceptar">
</form>
<h1>Despues del formulario</h1>
```

- **Agregamos una seguridad web en nuestro formulario** **{% csrf_token %}**

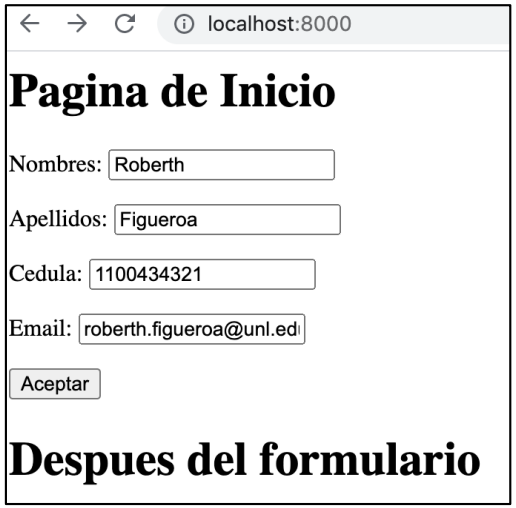
Nota: Un ataque CSRF obliga al navegador de una víctima autenticada a enviar una petición HTTP falsificado, incluyendo la sesión del usuario y cualquier otra información de autenticación incluida automáticamente, a una aplicación web vulnerable. Esto permite al atacante forzar al navegador de la víctima para generar pedidos que la aplicación vulnerable piensa son peticiones legítimas provenientes de la víctima.

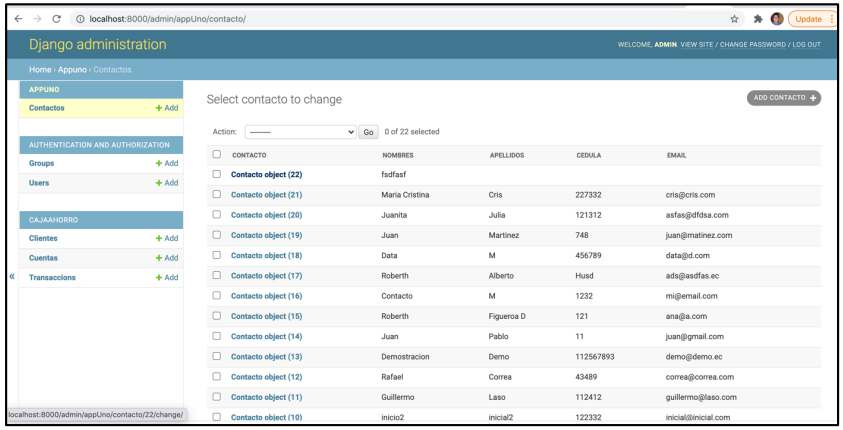
El archivo **home.html**

```
1  <h1>Pagina de Inicio</h1>
2  <form method="post" action=""> {% csrf_token %}
3      {{form.as_p}}
4      <input type="submit" value="Aceptar">
5  </form>
6  <h1>Despues del formulario</h1>
```

Reiniciar el Servidor Web **wsgi**

python manage.py runserver





Verificar que los datos se guarden en la base de datos de la aplicación

Redireccionar una App específica modificando las urls

Django versión 3.2.3

Accediendo desde otro Host externo o computador a mi aplicación en Django

Django versión 3.2.3

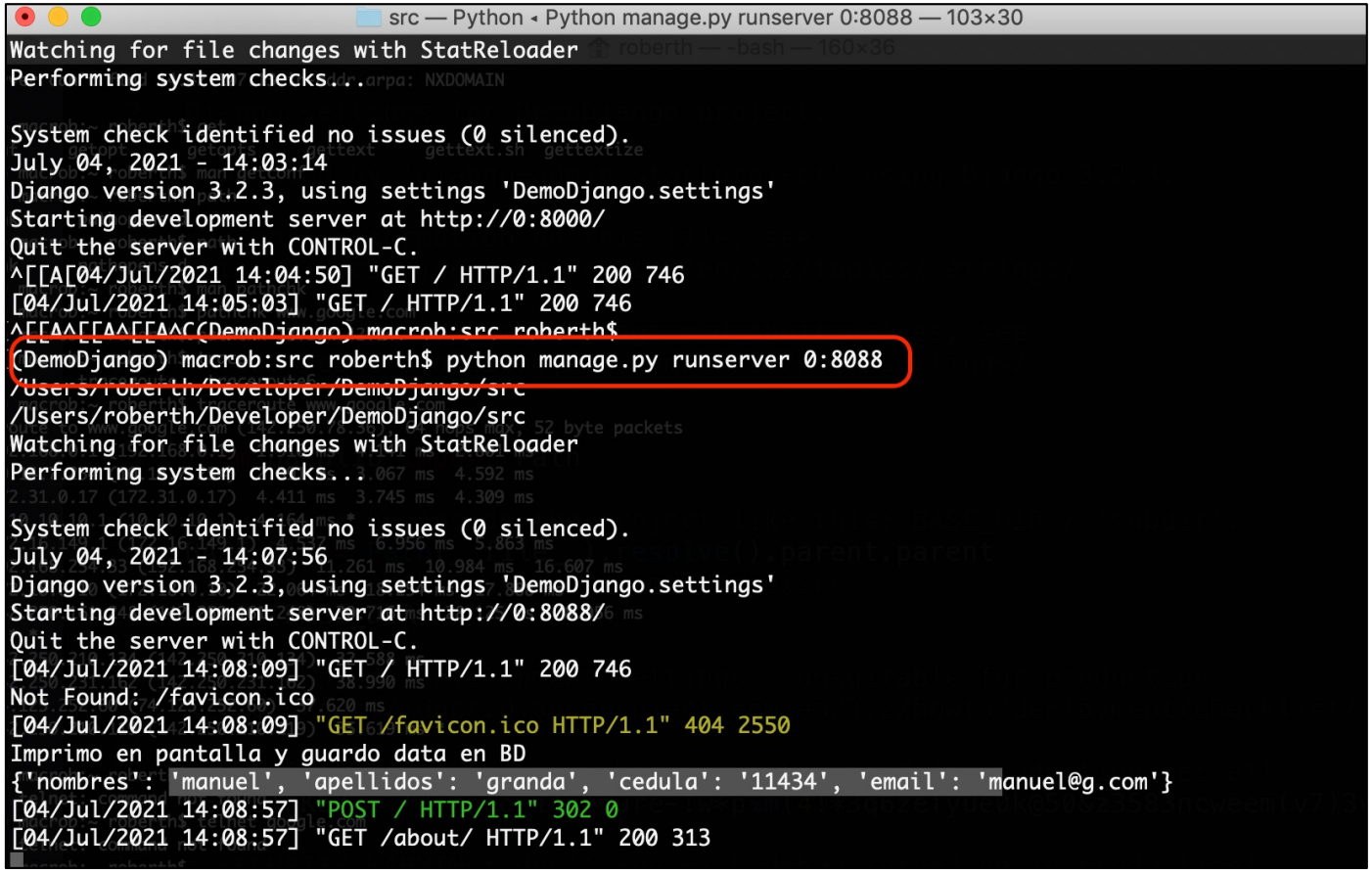
Modificar el **settings.py**

```
For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.2/ref/settings/
"""

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent
#/Users/roberth/Developer/DemoDjango/src
print(BASE_DIR)

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-!w*p%m(4!*3q6zefyue0k@50&z3583ncweem(v7)3v1&^rgat*'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

#Para permitir acceder desde host externos '*'
ALLOWED_HOSTS = ['*']


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'appUno',
    'CajaAhorro',
]

```

ALLOWED_HOST = [‘ * ’] → Permite acceso a otros equipos

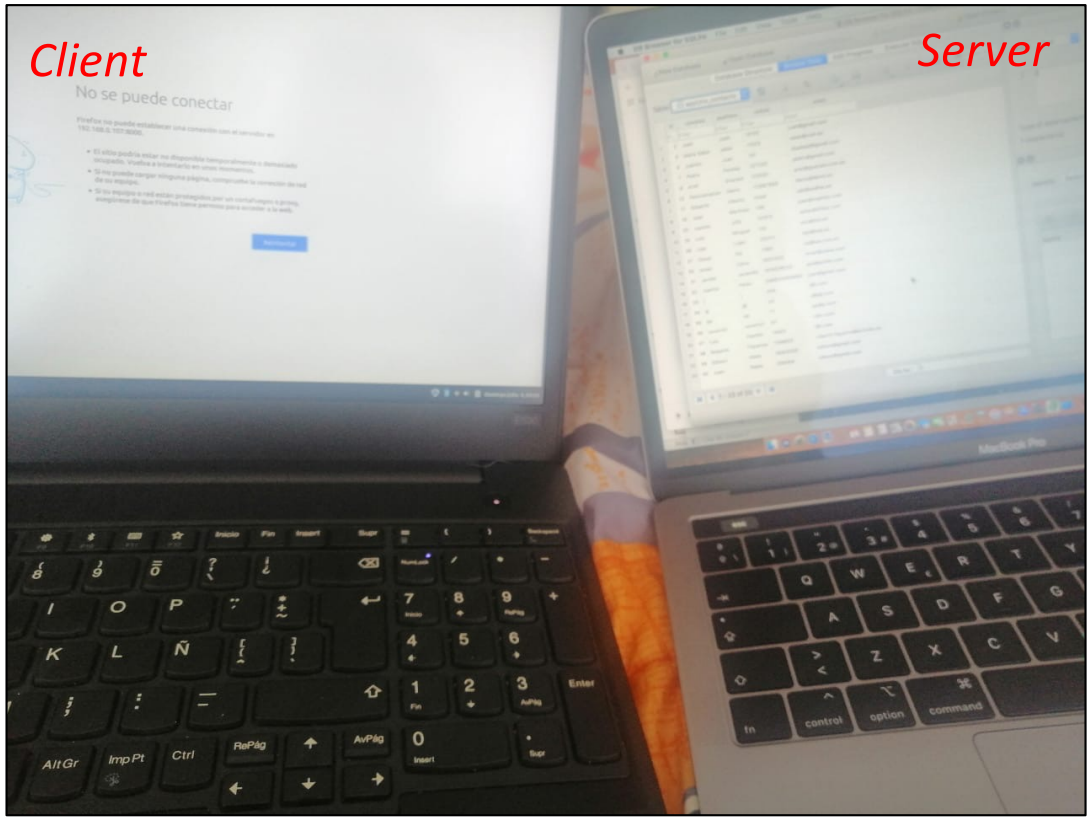
Reiniciar el Servidor Web WSGI **0:[puerto]**

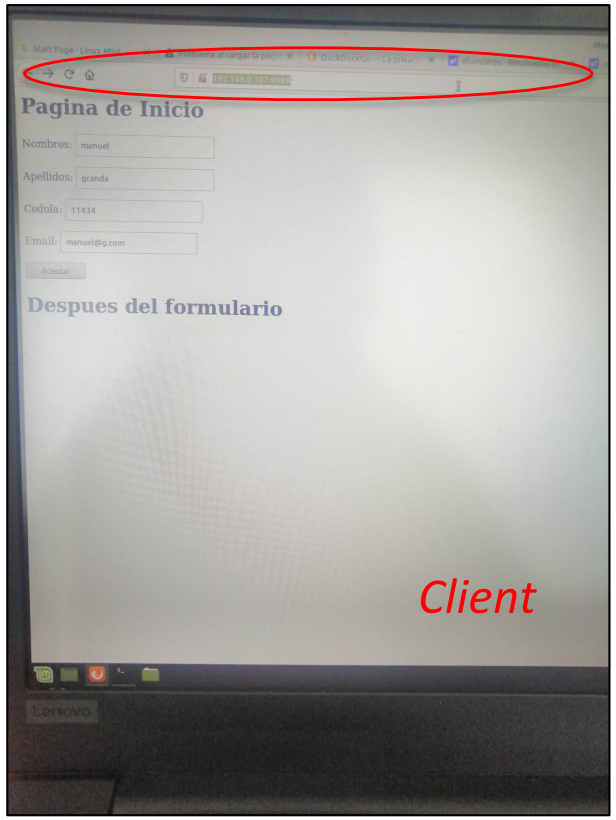
*python manage.py runserver **0:8088***



Invocar en el **host remoto** o equipo cliente

http://192.168.0.111:8087/

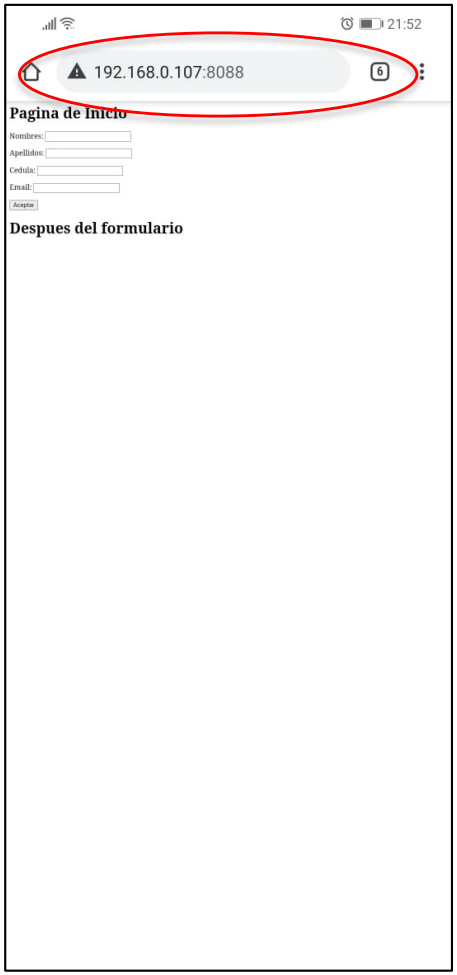


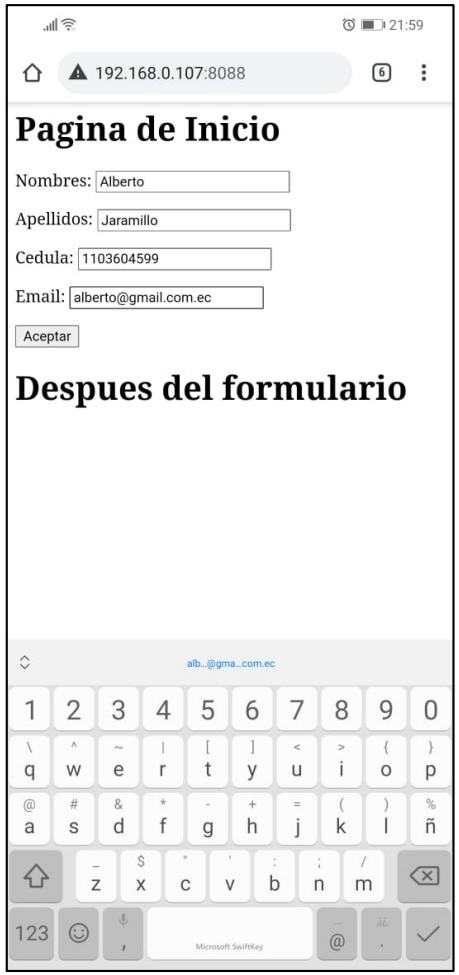


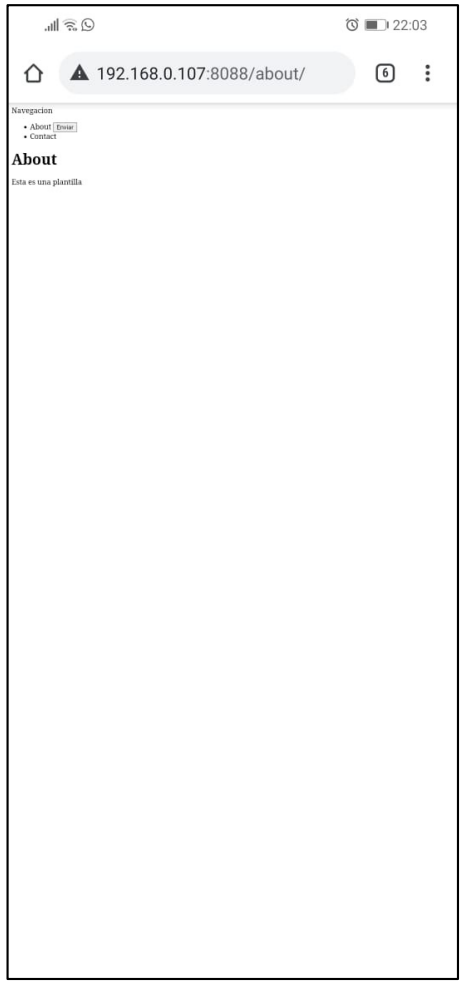
Usar en la LAN UNL → **python manage.py runserver 0:8081**
http://10.20.137.253:8081/

Ejecutar en el **host remoto** o celular

Colocar en el móvil → http://192.168.0.107:8088/





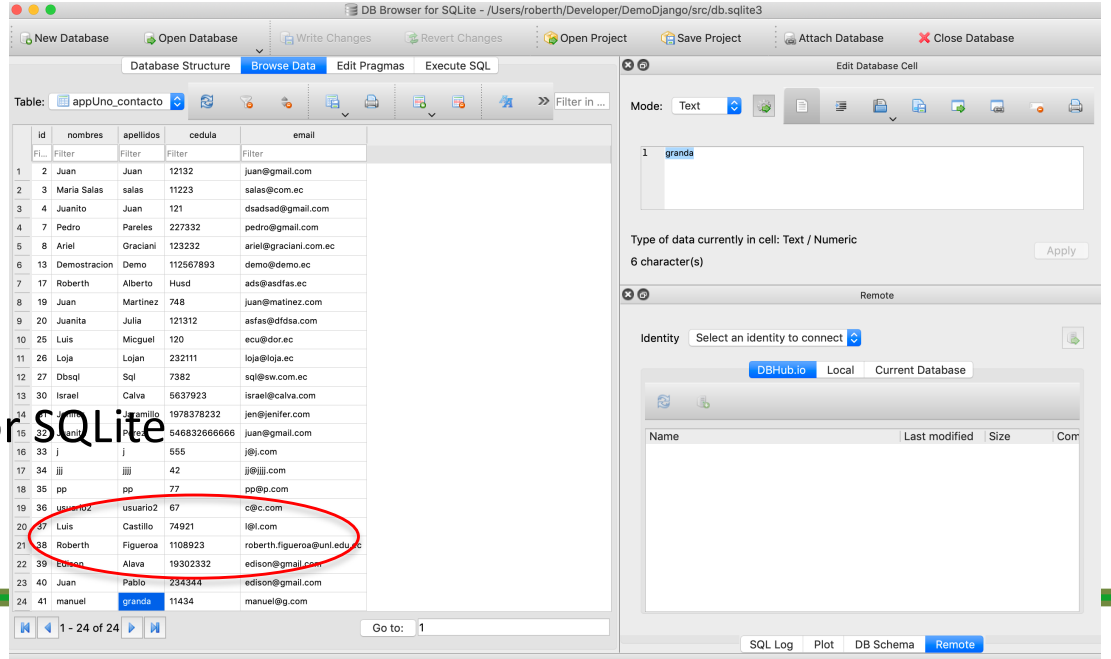


Ver en equipo servidor

```
Performing system checks...

System check identified no issues (0 silenced).
July 04, 2021 - 14:07:56
Django version 3.2.3, using settings 'DemoDjango.settings'
Starting development server at http://0:8088/
Quit the server with CONTROL-C.
[04/Jul/2021 14:08:09] "GET / HTTP/1.1" 200 746
Not Found: /favicon.ico
[04/Jul/2021 14:08:09] "GET /favicon.ico HTTP/1.1" 404 2550
Imprimo en pantalla y guardo data en BD
{'nombres': 'manuel', 'apellidos': 'granda', 'cedula': '11434', 'email': 'manuel@g.com'}
[04/Jul/2021 14:08:57] "POST / HTTP/1.1" 302 0
[04/Jul/2021 14:08:57] "GET /about/ HTTP/1.1" 200 313
```

Ejecutar desde el otro Host y verificar en el servidor los nuevos datos almacenados

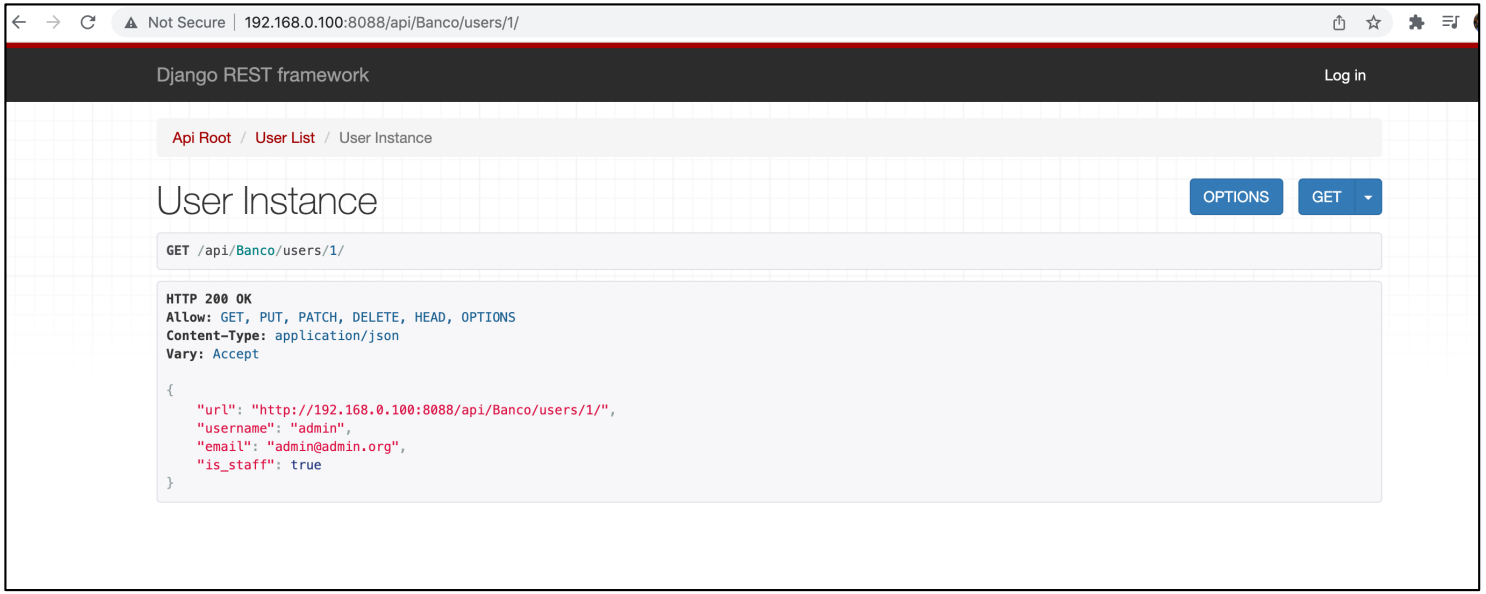
Ver la DB en Browser for SQLite



CREANDO UNA API REST

Django versión 3.2.3

Creando una API REST



Envío de datos en formato Json por Http

Usando Django Rest Framework

https://www.youtube.com/watch?v=F5mRW0jo-U4&t=400s

https://www.youtube.com/watch?v=TmsD8QExZ84

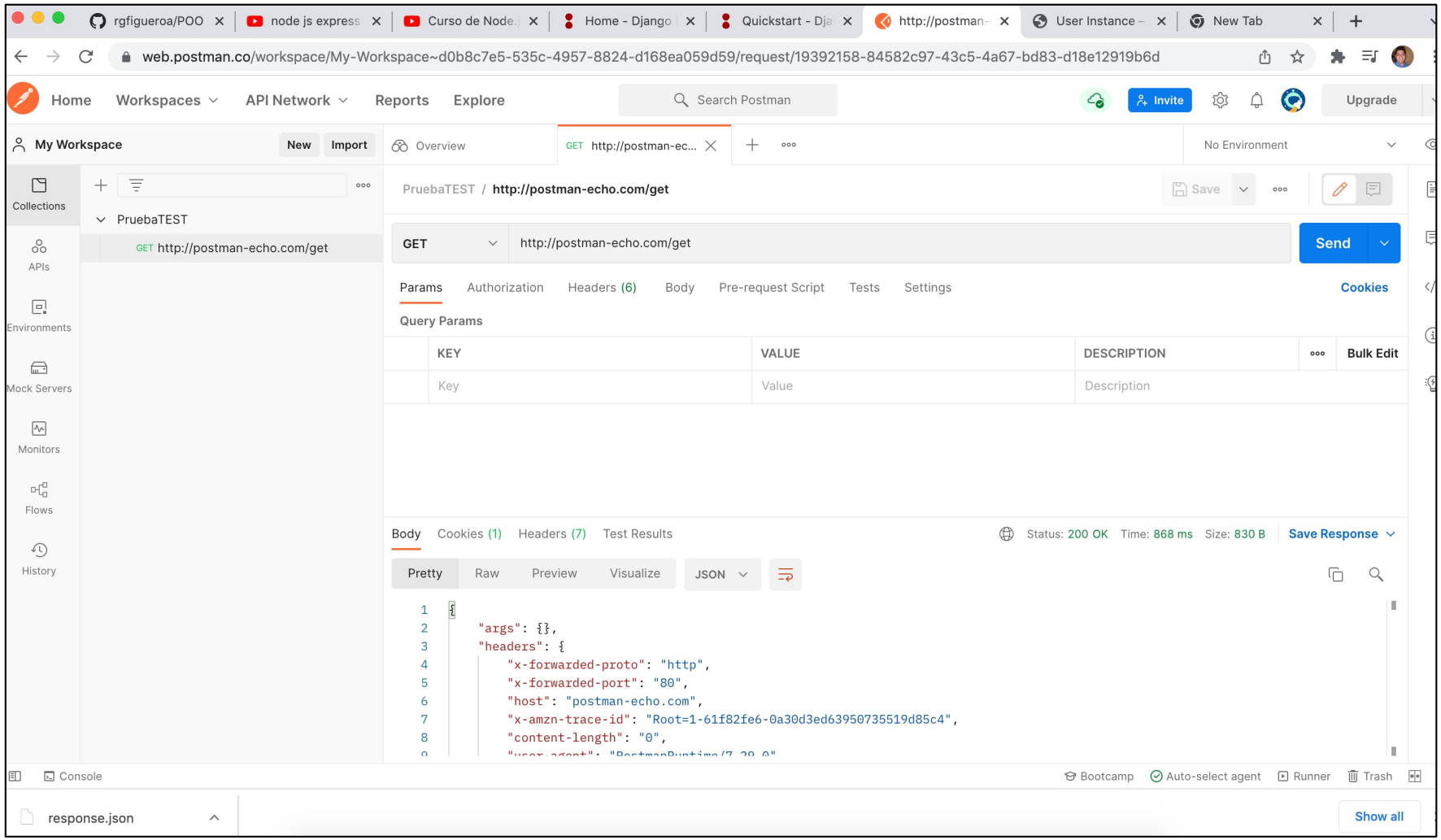
https://www.django-rest-framework.org/tutorial/quickstart/

https://docs.djangoproject.com/en/4.0/topics/db/queries/

Ejemplo de Django con Vue: → **https://djackets.codewithstein.com/**

https://www.youtube.com/watch?v=Yg5zkd9nm6w

Usando Postman o navegador web



Contacto

Ing. Roberth Figueroa Díaz M.Sc.
Carrera Computación
roberth.figueroa@unl.edu.ec
www.unl.edu.ec

Gracias por su atención

REDES UNL OFICIALES

Universidad Nacional De Loja - UNL @UnlOficial UnlOficial UnlOficial