Ejercicio 1

Cread un proyecto llamado proyecto en Django. Dentro del proyecto crear una aplicacion llamada aplicación

1. Usamos el siguiente comando **django-admin.py startproject *nombre_proyecto*** cambiando nombre_proyecto por el que nos indican.

2. Para crear la aplicación nos movemos a la carpeta del proyecto recién creado (cd nombre_proyecto) y ejecutamos el siguiente comando **python manage.py startapp *nombre_app*** cambiando nombre_app por el nombre que nos indican.

3. En el fichero settings.py, buscamos `INSTALLED_APPS` y añadimos 'nombre_app',

Ejercicio 2

Cread el modelo de datos (fichero models.py) que pueda ser accedido desde el interfaz de administracion. En el fichero memoria.txt incluid el nombre de usuario y la password del administrador. En la interfaz de administracion se deben poder ver las tablas relacionadas con pacientes, medicos y recetas

1. Vamos al fichero models.py que esta en la carpeta de la aplicación y ponemos tantas clases como tablas tenga la base de datos que queramos crear un ejemplo seria el siguiente.

```
class User(models.Model):
      userName = models.CharField(max_length=20,unique=True)
      password = models.CharField(max_length=20)
      mouseUser = models.ForeignKey(Game, related_name='gamemouseUsers',
null=True)
      cat1 = models.IntegerField(default=57, null=False)


 def __unicode__(self):
  return self.userName
```

Como se puede observar los Field son los campos de la tabla, en este ejemplo están los mas característicos. En caso de necesitar un campo que solo acepte un rango de numeros ir a la practica 3 y copiar la clase IntegerRangeField y el ejemplo de algun field esta en la clase move. Los id no se ponen ya que los añade automáticamente django.

2. Después vamos a admin.py y añadimos:

```
from aplicacion.models import *

admin.site.register(nombre_class)
```

Donde ponemos tantos admin.site.register como tablas tenemos, donde nombre_class lo cambiamos por el nombre de las clases antes creadas(la del IntegerRangeField de ponerla antes no se pone acá).

3. Abrimos una terminal en la carpeta donde esta la base de datos y ejecutamos **python manage.py makemigrations** y **python manage.py migrate**

4. En la terminal ejecutamos **python manage.py createsuperuser** y una vez creado ponemos levantar el servidor con **python manage.py runserver** y vamos al navegador web y ponemos localhost:8000/admin podemos hacer login con la cuenta recien creada y podemos ver que hay secciones con cada tabla.

Ejercicio 3

Escribid un script (llamado poblar.py) que inserte los datos siguientes en la base de datos usando la API ofrecida por Django.

1. Creamos en el mismo nivel de la base de datos un archivo llamado poblar.py, y copiar lo de 5.7 del pdf de django, solo hay que cambiar las funciones para que funcione con nuestros modelos creados anteriormente y añadir estos import:

   import os

```
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'proyecto.settings')
import django
django.setup()
from aplicacion.models import *
```

2. Se puede borrar el codigo del populate que nos dan e ir creando el nuestro según lo que pide el enunciado

Ejercicio 4

Cread un fichero python similar a test query (practica3) que imprima por pantalla todos los m´edicos que han recetado al paciente con el identificador de paciente asociado a paciente1. La consulta debe ser realizada en su totalidad por la base. No es admisible sacar de la base de datos todas las recetas y seleccionar -en la vista- las que satisfacen las restricciones

1. Copiar el de la practica 3 haciendo cambios en los import que son los mismos que en el ejercicio anterior, y cambiarlo para que hago lo que pide el ejercicio

```
      u = User.objects.filter(id=id)
if u.exists():
    u10 = u[0]
    print("El usuario con id = %d ya existe"%id)
else:
    u10=User(id=id,password=password,userName=username)
    u10.save()
```

Esto es lo que se basa el ejercicio, el filter es como un Select en sql, u siempre es una array y con exists() comprobamos que tenga alguna cosa, lo del else es como crear un nuevo elemento en la DB, siempre hay que hacer un save()

Ejercicio 5

En la misma aplicacion crear un webservice que dado un identificador de medico devuelva un listado con todos los pacientes (su nombre) que lo consuman.

1. Crear api.py en aplicación y copiar de Api.py de la p3 en server pero solo hasta el def de add_user (este incluido), en meta poner la clase y los campos de donde se va a buscar en la base de datos, cambiar de nombre, borrar las funciones de validar movimientos y realizar los que nos piden.

Ejercicio 6

En el mismo proyecto, cread un fichero python (llamadlo test server.py) que invoque este webservice con el identificador de m´edico asociado a medico1.

1. En proyecto/urls.py añadir **url(r'^aplicacion/', include("aplicacion.urls")),** donde aplicación es el nombre de nuestra aplicación,despues copiar el test_server de proyecto, en la ip de arriba cambiar el nombre de aplicación por el de nuestra aplicación y lo de medico por lo de meta del webservice.

2. En aplicación crear urls.py y copiar:

```
            from django.conf.urls import patterns, url, include
from django.contrib import admin
admin.autodiscover()
from tastypie.api import Api # modulo webservices
from api import UserResource # clase en api.py con informacion sobre url
import views
user_api=Api(api_name='v1')       # v1 es una etiqueta para identificar el
recurso
user_api.register(UserResource()) # registrar  api.UserResource
urlpatterns = patterns('',
        url(r'^api/',include(user_api.urls)),# agnadir los urls contenidas en
api.Userresource.prepend_urls
                       )
```