



**Universidad  
Andrés Bello®**  
Conectar • Innovar • Liderar

# ICF233 Ingeniería de Software II

Segundo Semestre 2020

# Formato Presentación Avance

Nombre del proyecto

Integrantes

# Objetivo del Proyecto y Necesidad que aborda

¿Cuál es la necesidad que aborda el proyecto?/¿Qué problema resuelve el proyecto?

¿Por qué es importante?

¿Quiénes son los clientes y usuarios?

¿Qué otros stakeholders son importantes para el proyecto?

# Alcance del Proyecto

Mapa de Historias del Proyecto

Describir el alcance de lo que se desarrollará, limitando lo que no está considerado hacer en el proyecto

Cantidad de historias y puntaje total del Backlog

Velocidad del equipo en Hito 3 de ISW I y proyección de cantidad de sprints para abarcar el backlog

# Beneficios del Proyecto

¿Qué beneficios traerá el proyecto a los clientes y usuarios?

¿Cómo se diferencia lo entregado por el proyecto de otras soluciones?

# Próximos Pasos

¿Qué reciben los clientes y usuarios en cada uno de los tres sprints planificados para el curso?

¿Qué riesgos ven para el desarrollo del proyecto?

# Ejemplo Planificación Ágil y Burndown chart

Como cliente necesito recibir una notificación cuando mi compra sea despachada y venga en camino para recibirla (4 puntos)

Como cliente necesito saber el estado de mi compra para hacerle seguimiento (16 puntos)

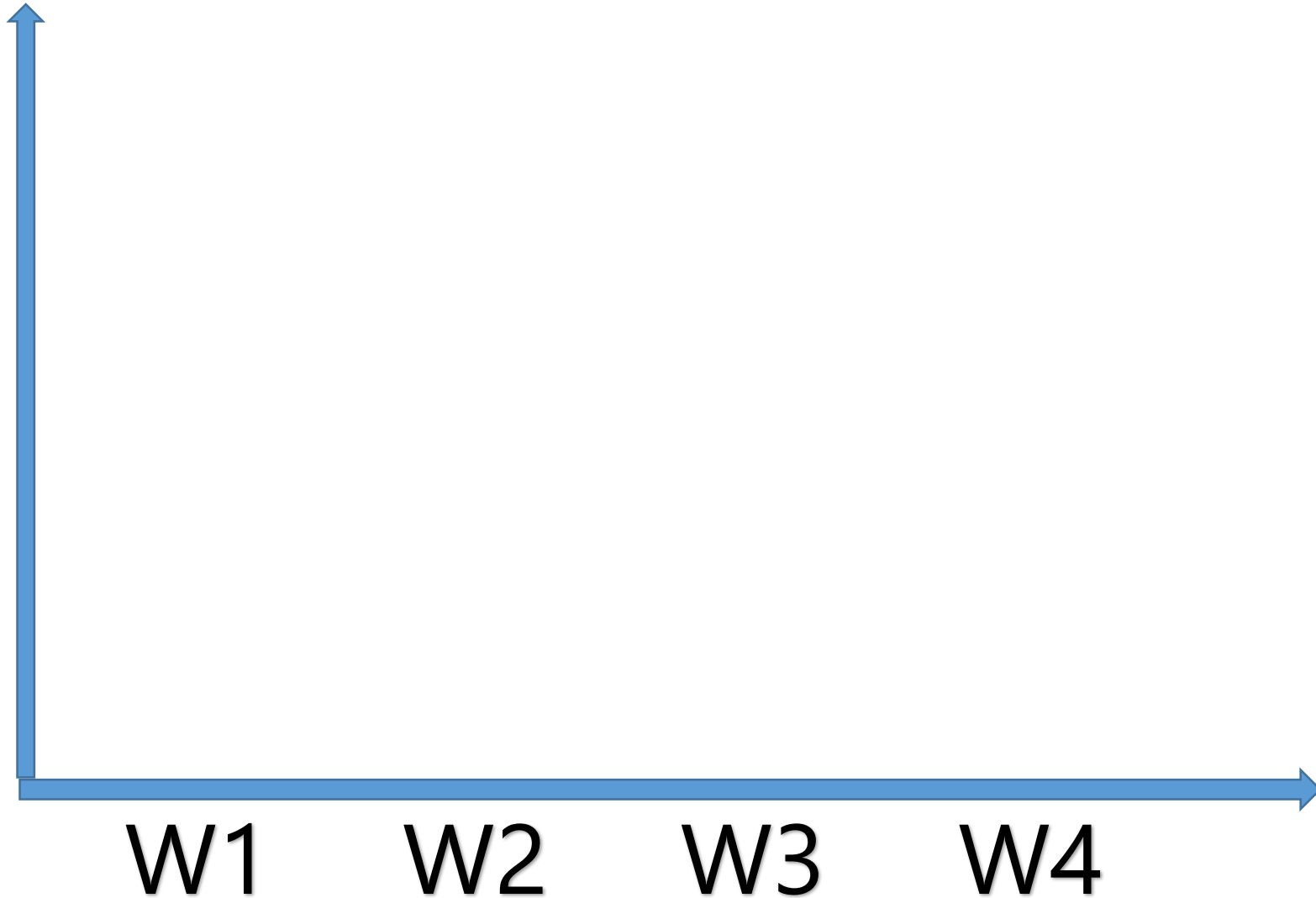
# Ejemplo Planificación Ágil y Burndown chart

Como cliente necesito recibir una notificación cuando mi compra sea despachada y venga en camino para recibirla (4 puntos)

Como cliente necesito saber el estado de mi compra para hacerle seguimiento (16 puntos)

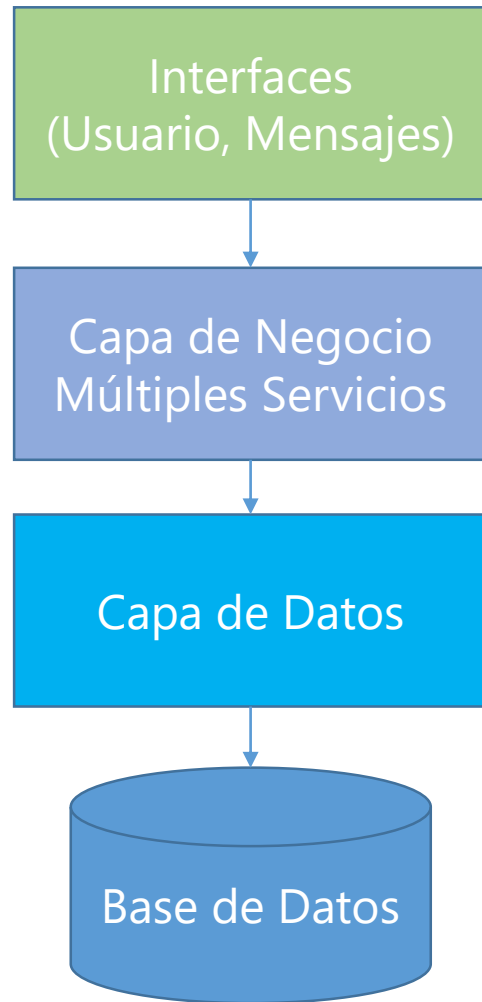


# Ejemplo Planificación Ágil y Burndown chart

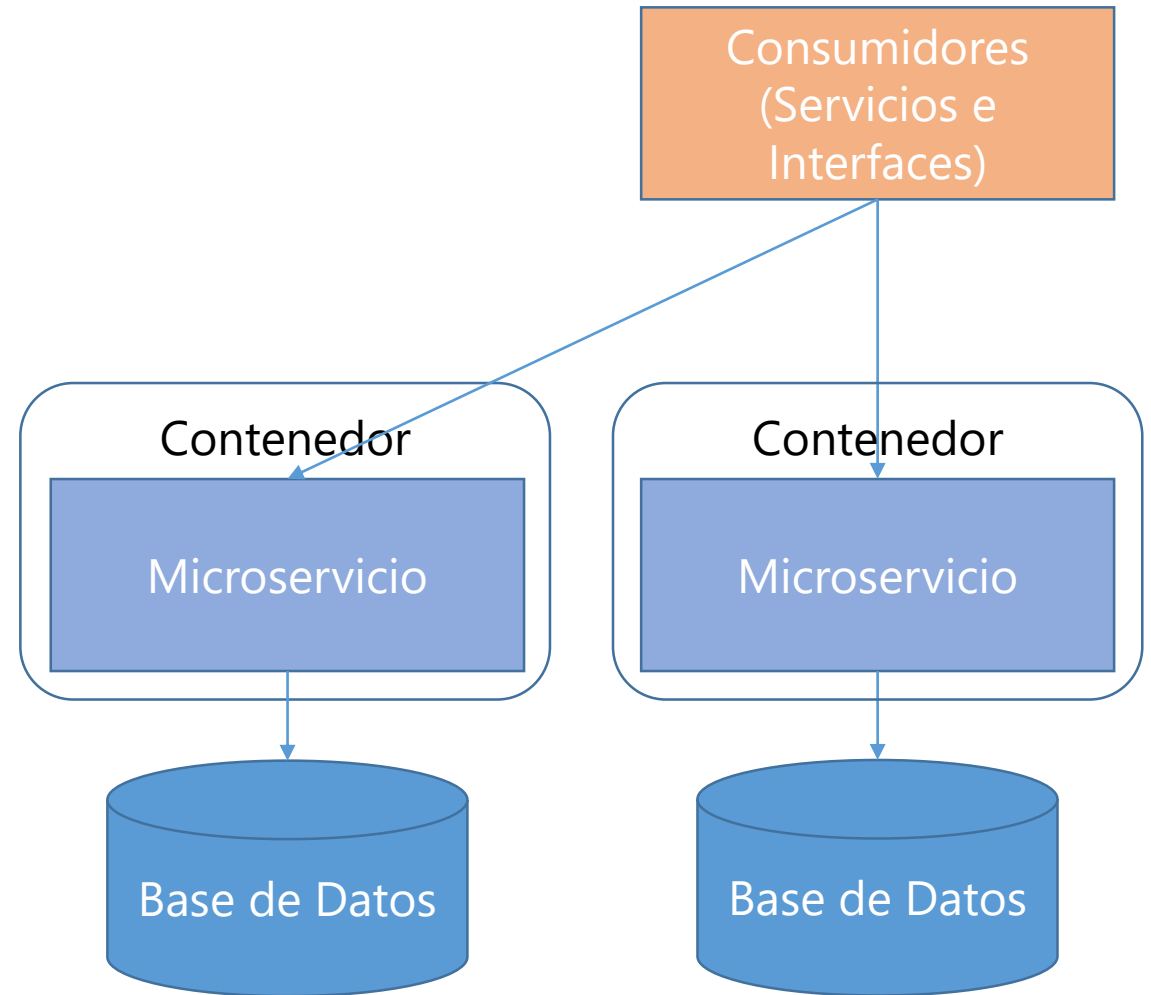


# Desarrollo de Servicios REST

# Arquitecturas Básicas

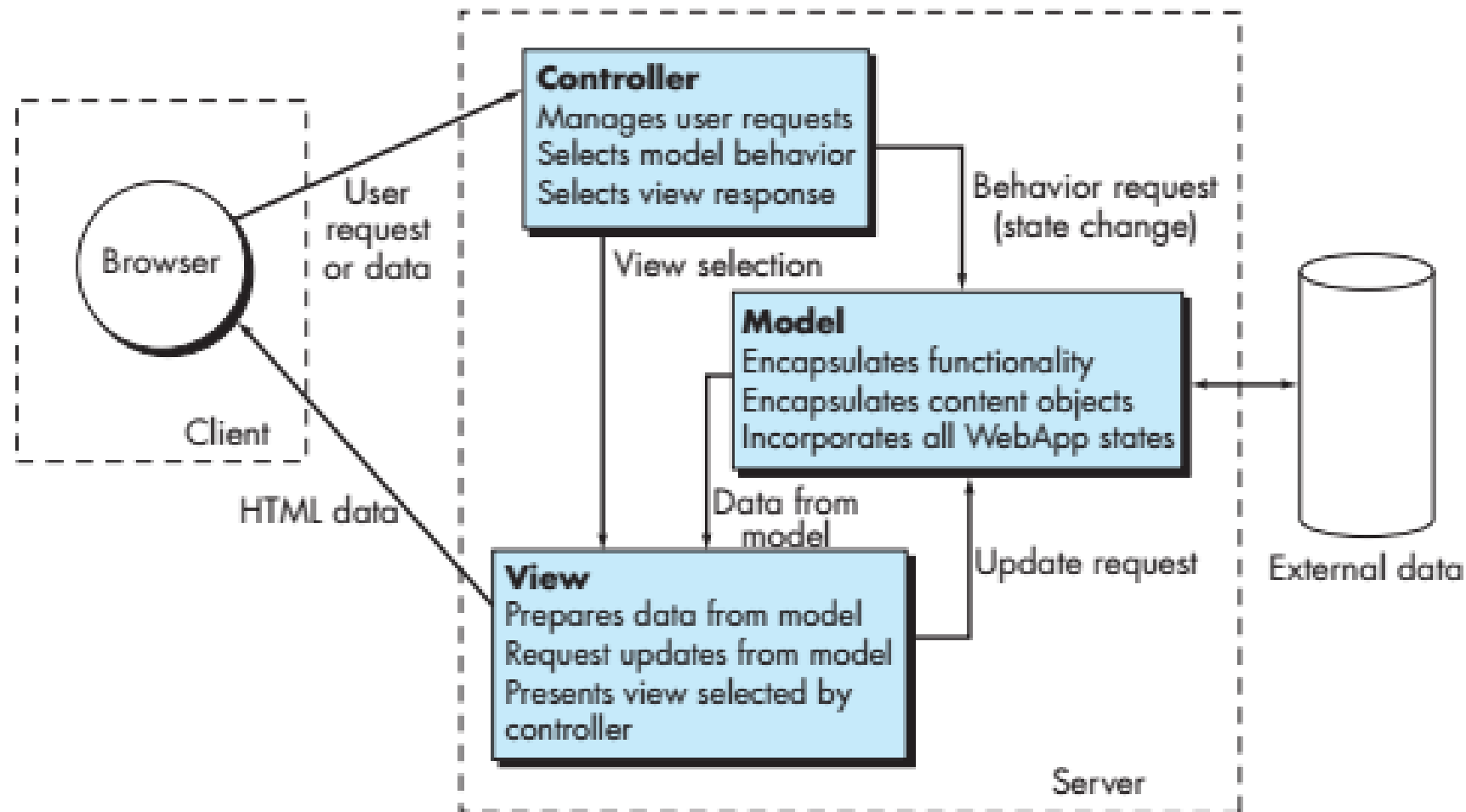


**Multicapa**



**Microservicios**

# Patrón Modelo – Vista – Controlador



# Creación de aplicación y modelo asociado al servicio

# Crear entorno virtual, activarlo e Instalar dependencias

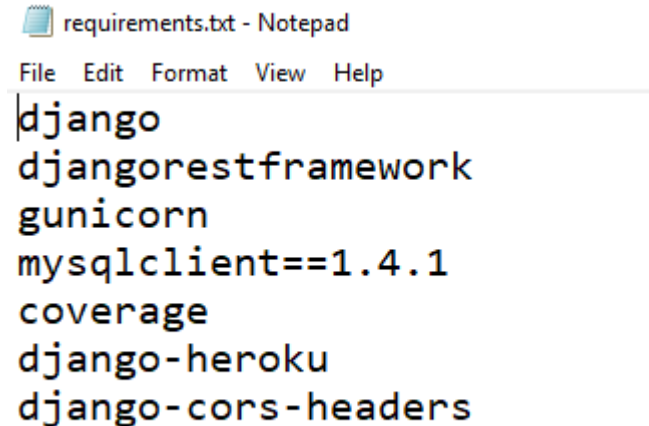
**pip3 install virtualenv**

Creamos un directorio para el proyecto e ingresamos en él. Ej. **servicios**

**virtualenv venv**

**venv/Scripts/activate.bat**

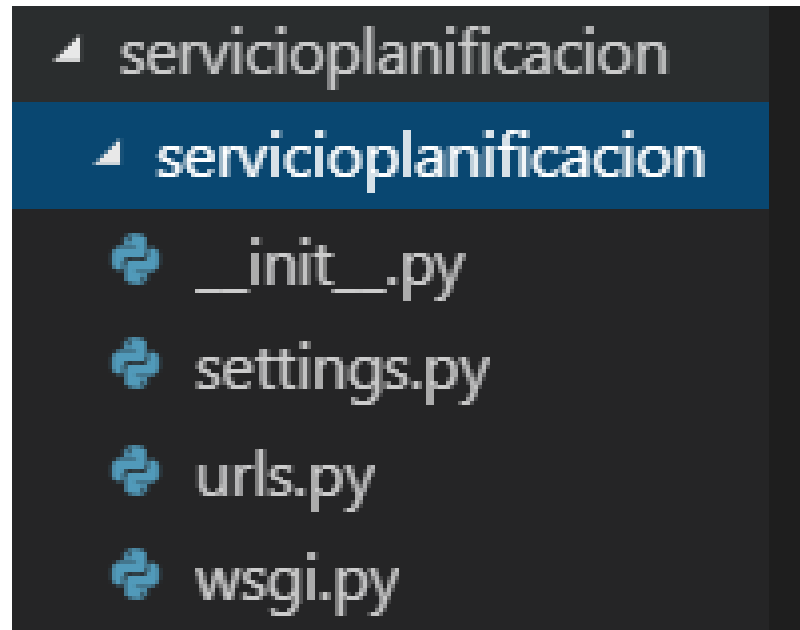
**pip3 install -r requirements.txt**



```
requirements.txt - Notepad
File Edit Format View Help
django
django-rest-framework
unicorn
mysqlclient==1.4.1
coverage
django-heroku
django-cors-headers
```

# Crear aplicación

```
django-admin startproject servicioplanificacion
```



# Editor settings.py

```
ALLOWED_HOSTS = []

# Application definition

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

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```





# Crear Aplicación

```
(venv) D:\Clases\UNAB\ICF232-201910\servicios\servicioplanificacion>python manage.py startapp api
```

```
<DIR>      .
<DIR>      ..
<DIR>      api
          568 manage.py
<DIR>      servicioplanificacion
```

# Editar models.py y settings.py tal como en la aplicación Web

```
from django.db import models

class Curso(models.Model):
    sigla=models.CharField(max_length=6)
    nombre=models.CharField(max_length=60)
    creditos=models.IntegerField()

    def __str__(self):
        return "{}".format(self.nombre)
```



ORM  
Object Relational Mapper

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'api'
]
```



# Implementamos un test editando test.py

```
from django.test import TestCase

from .models import Curso

class CursoTestCase(TestCase):
    """Esta clase define la testsuite para el Curso."""

    def setUp(self):
        """Definición de variables generales."""
        self.sigla = "ICF121"
        self.nombre="Introducción a la Ingeniería Civil Informática"
        self.creditos=6
        self.curso = Curso(sigla=self.sigla,nombre=self.nombre,creditos=self.creditos)

    def test_creacion_de_curso(self):
        """Test de creación de un curso"""
        old_count = Curso.objects.count()
        self.curso.save()
        new_count = Curso.objects.count()
        self.assertNotEqual(old_count, new_count)
```

# Creamos y Ejecutamos las migraciones

```
(venv) D:\Clases\UNAB\ICF232-201910\servicios\servicioplanificacion>python manage.py makemigrations
Migrations for 'api':
  api\migrations\0001_initial.py
    - Create model Curso
```

```
(venv) D:\Clases\UNAB\ICF232-201910\servicios\servicioplanificacion>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, api, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying api.0001_initial... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying sessions.0001_initial... OK
```

# Ejecutamos los tests

```
(venv) D:\Clases\UNAB\ICF232-201910\servicios\servicioplanificacion>python manage.py test -v 2
Creating test database for alias 'default' ('file:memorydb_default?mode=memory&cache=shared')...
Operations to perform:
  Synchronize unmigrated apps: messages, rest_framework, staticfiles
  Apply all migrations: admin, api, auth, contenttypes, sessions
Synchronizing apps without migrations:
  Creating tables...
    Running deferred SQL...
Running migrations:
test_creacion_de_curso (api.testsCursoTestCase)
Test de creación de un curso ... ok

-----
Ran 1 test in 0.008s

OK
Destroying test database for alias 'default' ('file:memorydb_default?mode=memory&cache=shared')...
```

# Django REST

<https://www.django-rest-framework.org/community/tutorials-and-resources/>

<https://www.django-rest-framework.org/api-guide/generic-views/>

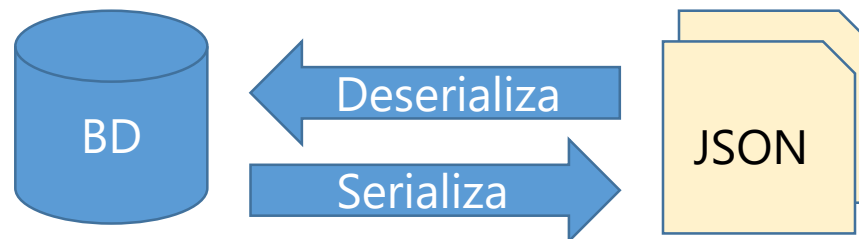
<https://www.django-rest-framework.org/api-guide/testing/#apiclient>

# Creamos un serializador en serializers.py

```
from rest_framework import serializers
from .models import Curso

class CursoSerializer(serializers.ModelSerializer):
    """Serializador para mapear un curso a formato JSON."""

    class Meta:
        """Meta clase para el mapeo de atributos."""
        model = Curso
        fields = ('id', 'sigla', 'nombre', 'creditos')
        read_only_fields = ()
```



# Extendemos el Test

```
from django.test import TestCase

from .models import Curso
from rest_framework.test import APIClient
from rest_framework import status
from django.urls import reverse
```

```
class ViewTestCase(TestCase):
    """Esta clase define la testsuite para la API REST."""
    def setUp(self):
        """Definición de variables generales"""
        self.client = APIClient()
        curso_data = {'sigla': 'ICF121', 'nombre': 'Introducción a la Ingeniería Civil Informática', 'creditos': 6}
        self.response_setup = self.client.post(
            reverse('create'),
            curso_data,
            format="json")

    def test_api_creacion_de_cursos(self):
        """Test creación de curso a través de la API."""
        self.assertEqual(self.response_setup.status_code, status.HTTP_201_CREATED)
```



# Verificamos con el test

```
(venv) D:\Clases\UNAB\ICF232-201910\servicios\servicioplanificacion>python manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.E
=====
ERROR: test_api_creacion_de_cursos (api.tests.ViewTestCase)
Test creación de curso a través de la API.
-----
Traceback (most recent call last):
  File "D:\Clases\UNAB\ICF232-201910\servicios\servicioplanificacion\api\tests.py", line 32, in setUp
    reverse('create'),
  File "D:\Clases\UNAB\ICF232-201910\servicios\venv\lib\site-packages\django\urls\base.py", line 90, in reverse
    return iri_to_uri(resolver._reverse_with_prefix(view, prefix, *args, **kwargs))
  File "D:\Clases\UNAB\ICF232-201910\servicios\venv\lib\site-packages\django\urls\resolvers.py", line 668, in _reverse_with_prefix
    raise NoReverseMatch(msg)
django.urls.exceptions.NoReverseMatch: Reverse for 'create' not found. 'create' is not a valid view function or pattern name.
-----
Ran 2 tests in 0.006s

FAILED (errors=1)
Destroying test database for alias 'default'...
```

# Implementamos la vista

```
from django.shortcuts import render

from rest_framework import generics
from .serializers import CursoSerializer
from .models import Curso

class CreateView(generics.ListCreateAPIView):
    """Vista que representa el comportamiento de la API REST."""
    #El queryset contiene la colección con todos los cursos
    queryset = Curso.objects.all()
    serializer_class = CursoSerializer


    def perform_create(self, serializer):
        """Almacena los datos recibidos mediante POST como un Curso."""
        serializer.save()
```

# Implementamos los URLs de la API, agregando urls.py

```
from django.conf.urls import url, include
from rest_framework.urlpatterns import format_suffix_patterns
from .views import CreateView


urlpatterns = {
    url(r'^curso/$', CreateView.as_view(), name="create")
}
urlpatterns = format_suffix_patterns(urlpatterns)
```

# Agregamos los URLs de la API, en urls.py principal



```
from django.contrib import admin
from django.urls import path, re_path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    re_path(r'^', include('api.urls'))
]
```



# Verificamos funcionamiento correcto con los tests

```
(venv) D:\Clases\UNAB\ICF232-201910\servicios\servicioplanificacion>python manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
..
-----
Ran 2 tests in 0.020s

OK
Destroying test database for alias 'default'...
```

# Ejecutamos el servidor

## Verificamos el funcionamiento de los servicios

Create

## Create

OPTIONS GET

Vista que representa el comportamiento de la API REST.

GET /curso/

HTTP 200 OK  
Allow: GET, POST, HEAD, OPTIONS  
Content-Type: application/json  
Vary: Accept  
[]

Raw data HTML form

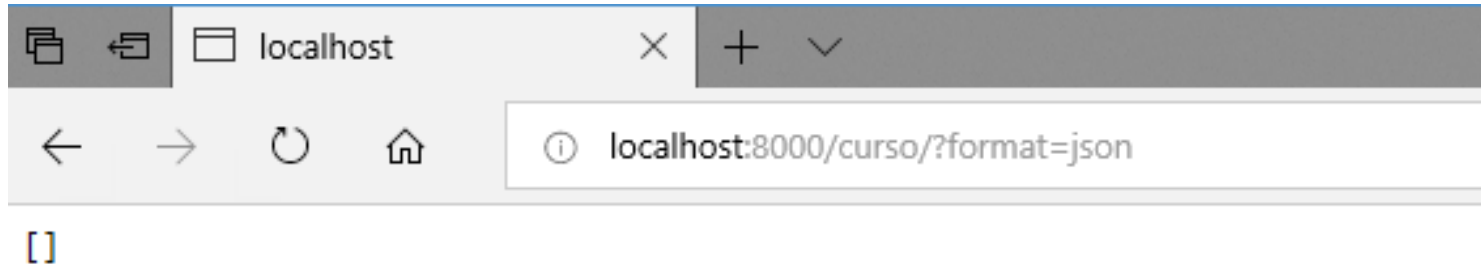
Sigla

Nombre

Creditos

POST

# Verificamos el funcionamiento de los servicios



# Agregamos un curso

Create

OPTIONS GET

Vista que representa el comportamiento de la API REST.

POST /curso/

HTTP 201 Created  
Allow: GET, POST, HEAD, OPTIONS  
Content-Type: application/json  
Vary: Accept

```
{
  "id": 1,
  "sigla": "ICF121",
  "nombre": "Introducción a la Ingeniería Civil Informática",
  "creditos": 6
}
```

Raw data HTML form

Sigla ICF121

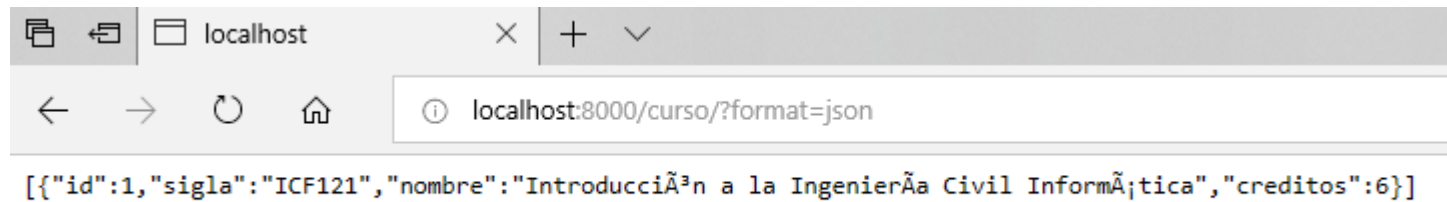
Nombre Introducción a la Ingeniería Civil Informática

Creditos 6

POST



# Verificamos que el curso se agregó



# En la base de datos

DB Browser for SQLite - D:\Clases\UNAB\ICF232-201910\servicios\servicioplanificacion\db.sqlite3

Archivo Editar Ver Herramientas Ayuda

Nueva base de datos Abrir base de datos Guardar cambios Deshacer cambios Abrir proyecto Guardar proyecto Anexar base de datos

Estructura Hoja de datos Editar pragmas Ejecutar SQL

Tabla: api\_curso

	id	sigla	creditos	nombre
	Filtro	Filtro	Filtro	Filtro
1	1	ICF121	6	Introducción a...

Editar celda

Modo: Texto

Importar Exportar Borrar

Introducción a la Ingeniería Civil Informática

Tipo de datos actualmente en la celda: Texto / Numérico

Apl

# Cambio de Base de Datos a Mysql

# Creación de base de datos y asignación de privilegios

```
create user "icf"@"%" identified by "Secret.123";
```

```
grant all privileges on *.* to "icf"@"%" identified by "Secret.123";
```

```
create database planificacion character set utf8 collate utf8_bin;
```

```
grant all on planificacion.* to "icf"@"%" identified by "Secret.123";
```


# Configuración de Conexión en settings.py

```
DATABASES = {  
#     'default': {  
#         'ENGINE': 'django.db.backends.sqlite3',  
#         'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),  
#     }  
    'default': {  
        'ENGINE': 'django.db.backends.mysql',  
        'NAME': 'planificacion',  
        'USER': 'icf',  
        'PASSWORD': 'Secret.123',  
        'DEFAULT-CHARACTER-SET': 'utf8',  
        'HOST': '127.0.0.1',  
        'PORT': '3306',  
        'TEST': {  
            'NAME': 'planificacion_test'  
        }  
    }  
}
```

Ahora debemos volver a ejecutar las migraciones  
python manage.py migrate --run-syncdb

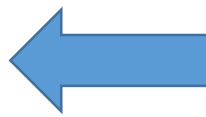
# Integración a tablas no creadas por django

<https://docs.djangoproject.com/en/2.2/howto/legacy-databases/>



```
class Person(models.Model):  
    id = models.IntegerField(primary_key=True)  
    first_name = models.CharField(max_length=70)  
    class Meta:  
        managed = False  
        db_table = 'CENSUS_PERSONS'
```

```
$ python manage.py inspectdb > models.py
```



inspectdb permite  
hacer ingeniería  
inversa del esquema  
de la Base de Datos

# Ejecución de SQL directo

<https://docs.djangoproject.com/en/2.2/topics/db/sql/>



```
>>> for p in Person.objects.raw('SELECT * FROM myapp_person'):
...     print(p)
John Smith
Jane Jones
```

Un raw query permite definir el SQL a usar pero retornar objetos del modelo

```
from django.db import connection

def my_custom_sql(self):
    with connection.cursor() as cursor:
        cursor.execute("UPDATE bar SET foo = 1 WHERE baz = %s", [self.baz])
        cursor.execute("SELECT foo FROM bar WHERE baz = %s", [self.baz])
        row = cursor.fetchone()

    return row
```

También es posible usar SQL y retornar datos simples no relacionados a un modelo

# Completando la TestSuite



# Completamos los test

```
def test_api_obtener_curso(self):
    """Test de obtención de curso a través de la API."""
    curso = Curso.objects.get()
    response = self.client.get(
        reverse('details',
            kwargs={'pk': curso.id}), format="json")

    self.assertEqual(response.status_code, status.HTTP_200_OK)
    self.assertContains(response, curso)

def test_api_actualizar_curso(self):
    """Test de actualización de curso a través de la API."""
    curso = Curso.objects.get()
    actualizacion_curso = {'nombre': 'Introduccion a la Ingenieria Civil Informatica'}
    res = self.client.patch(
        reverse('details', kwargs={'pk': curso.id}),
        actualizacion_curso, format='json'
    )
    self.assertEqual(res.status_code, status.HTTP_200_OK)

def test_api_borrar_curso(self):
    """Test borrado de curso a través de la API."""
    curso = Curso.objects.get()
    response = self.client.delete(
        reverse('details', kwargs={'pk': curso.id}),
        format='json',
        follow=True)

    self.assertEqual(response.status_code, status.HTTP_204_NO_CONTENT)
```

# Incorporamos la vista de detalle y su url

```
class DetailsView(generics.RetrieveUpdateDestroyAPIView):  
    """Vista que maneja las peticiones GET, PUT y DELETE."""  
  
    queryset = Curso.objects.all()  
    serializer_class = CursoSerializer
```

```
from django.conf.urls import url, include  
from rest_framework.urlpatterns import format_suffix_patterns  
from .views import CreateView  
from .views import DetailsView  
  
urlpatterns = {  
    url(r'^curso/$', CreateView.as_view(), name="create"),  
    url(r'^curso/(?P<pk>[0-9]+)/$', DetailsView.as_view(), name="details")  
}  
urlpatterns = format_suffix_patterns(urlpatterns)
```

# Ejecutamos los test

```
(venv) D:\Clases\UNAB\ICF232-201910\servicios\servicioplanificacion>python manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.....
-----
Ran 5 tests in 0.053s

OK
Destroying test database for alias 'default'...
```

# Detenemos y reiniciamos el servidor

## http://localhost:8000/curso/1/

The screenshot shows a web browser window with the address bar at `localhost:8000/curso/1/`. The page title is "Details - Django REST framework". The main content area is titled "Details" and includes a "Create" link and a "Details" breadcrumb. Below the title, there are three buttons: "DELETE", "OPTIONS", and "GET". A description states: "Vista que maneja las peticiones GET, PUT y DELETE." Below this, a "GET /curso/1/" request is shown with its response. The response is an HTTP 200 OK with headers: `Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS`, `Content-Type: application/json`, and `Vary: Accept`. The response body is a JSON object: `{ "id": 1, "sigla": "ICF121", "nombre": "Introducción a la Ingeniería Civil Informática", "creditos": 6 }`. At the bottom, there is a "Raw data" tab and an "HTML form" tab. The "HTML form" tab is active, showing a form with three fields: "Sigla" (ICF121), "Nombre" (Introducción a la Ingeniería Civil Informática), and "Creditos" (6). A "PUT" button is located at the bottom right of the form.

Details

DELETE OPTIONS GET

Vista que maneja las peticiones GET, PUT y DELETE.

GET /curso/1/

HTTP 200 OK  
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS  
Content-Type: application/json  
Vary: Accept

```
{
  "id": 1,
  "sigla": "ICF121",
  "nombre": "Introducción a la Ingeniería Civil Informática",
  "creditos": 6
}
```

Raw data HTML form

Sigla ICF121

Nombre Introducción a la Ingeniería Civil Informática


Creditos 6

PUT

# Seguridad en Servicios REST

# Incorporando autenticación basada en tokens

## Editamos settings.py y urls.py




```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'rest_framework',  
    'api',  
    'rest_framework.authtoken'  
]  
  
REST_FRAMEWORK = {  
    'DEFAULT_AUTHENTICATION_CLASSES': (  
        'rest_framework.authentication.BasicAuthentication',  
        'rest_framework.authentication.TokenAuthentication',  
    )  
}
```

```
from rest_framework.authtoken import views  
urlpatterns += [  
    path(r'api-token-auth/', views.obtain_auth_token)  
]
```

# Ejecutamos migraciones

```
(venv) D:\Clases\UNAB\ICF232-201910\servicios\servicioplanificacion>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, api, auth, authtoken, contenttypes, sessions
Running migrations:
  Applying authtoken.0001_initial... OK
  Applying authtoken.0002_auto_20160226_1747... OK
```

# Habilitamos la verificación de permisos en settings.py



```
REST_FRAMEWORK = {  
    'DEFAULT_AUTHENTICATION_CLASSES': (  
        'rest_framework.authentication.BasicAuthentication',  
        'rest_framework.authentication.TokenAuthentication',  
    ),  
    'DEFAULT_PERMISSION_CLASSES': (  
        'rest_framework.permissions.IsAuthenticated',  
    )  
}
```

<https://www.django-rest-framework.org/api-guide/permissions/>



# Verificamos con los test, fallan

```
(venv) D:\Clases\UNAB\ICF232-201910\servicios\servicioplanificacion>python manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
```

```
-----
Ran 5 tests in 0.036s
```

```
FAILED (failures=1, errors=3)
```

```
Destroying test database for alias 'default'...
```

# Agregamos un url de login en views.py

```
from django.contrib.auth import authenticate
from django.views.decorators.csrf import csrf_exempt
from rest_framework.authtoken.models import Token
from rest_framework.decorators import api_view, permission_classes
from rest_framework.permissions import AllowAny
from rest_framework.status import (
    HTTP_400_BAD_REQUEST,
    HTTP_404_NOT_FOUND,
    HTTP_200_OK
)
from rest_framework.response import Response
```

[https://www.django-rest-framework.org/api-guide/views/#api\\_view](https://www.django-rest-framework.org/api-guide/views/#api_view)

<https://docs.djangoproject.com/en/2.2/ref/csrf/>

<https://docs.djangoproject.com/en/2.2/topics/class-based-views/intro/#id1>

# Agregamos un url de login en views.py

```
@csrf_exempt
@api_view(["POST"])
@permission_classes((AllowAny,))
def login(request):
    username = request.data.get("username")
    password = request.data.get("password")
    if username is None or password is None:
        return Response({'error': 'Please provide both username and password'},
                        status=HTTP_400_BAD_REQUEST)
    user = authenticate(username=username, password=password)
    if not user:
        return Response({'error': 'Invalid Credentials'},
                        status=HTTP_404_NOT_FOUND)
    token, _ = Token.objects.get_or_create(user=user)
    return Response({'token': token.key},
                    status=HTTP_200_OK)
```

[https://www.django-rest-framework.org/api-guide/views/#api\\_view](https://www.django-rest-framework.org/api-guide/views/#api_view)

<https://docs.djangoproject.com/en/2.2/ref/csrf/>

<https://docs.djangoproject.com/en/2.2/topics/class-based-views/intro/#id1>

# Agregamos el url

```
from django.conf.urls import url, include
from rest_framework.urlpatterns import format_suffix_patterns
from .views import CreateView
from .views import DetailsView
from .views import login

urlpatterns = {
    url(r'^curso/$', CreateView.as_view(), name="create"),
    url(r'^curso/(?P<pk>[0-9]+)/$', DetailsView.as_view(), name="details"),
    url(r'^api/login', login)
}
urlpatterns = format_suffix_patterns(urlpatterns)
```

# Creamos un usuario para las API

```
(venv) D:\Clases\UNAB\ICF232-201910\servicios\servicioplanificacion>python manage.py createsuperuser  
Username (leave blank to use 'pablo'): servidor  
Email address: api@servicio.com  
Password:  
Password (again):  
Superuser created successfully.
```

Secret.123

# Verificamos la api de login usando curl

```
C:\Users\pablo>curl -X POST -F "username=servidor" -F "password=Secret.123" http://localhost:8000/api/login  
{"token":"dc13f0aeb2e35770ae17f70ba2419d2948559884"}
```

```
C:\Users\pablo>curl -X GET http://localhost:8000/curso/1/  
{"detail":"Authentication credentials were not provided."}
```

```
C:\Users\pablo>curl -X GET --header "Authorization: Token dc13f0aeb2e35770ae17f70ba2419d2948559884" http://localhost:8000/curso/1/  
{"id":1,"sigla":"ICF121","nombre":"Introducci|n a la Ingenier|a Civil Inform|tica","creditos":6}
```

# Preguntas

