MINI PLATAFORMA
CREAR PROYECTO Y MODULOS

I. Crear el proyecto de Django.

```
$ django-admin.py startproject miniplataforma .
```

II. Crear ambos módulos (apps).

```
$ python manage.py startapp clases
$ python manage.py startapp discusion
```

III. Agregar apps a *settings.py*.

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # Uncomment the next line to enable the admin:
    # 'django.contrib.admin',
    # Uncomment the next line to enable admin documentation:
    # 'django.contrib.admindocs',
    'clases',
    'discusion',
)
```

IV. Agregar *sqlite3* como base de datos.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': 'miniplataforma.db',
        'USER': '',
        'PASSWORD': '',
        'HOST': '',
        'PORT': '',
    }
}
```

V. Levantar el *servidor* de desarrollo.

```
$ python manage.py runserver
```

APP CLASES

I. Crear archivo *clases.html* dentro de la carpeta *templates* en la app de *clases*.

```
<!doctype html>
<html lang="es">
<head>
     <meta charset="UTF-8">
     <title>Mini Plataforma</title>
</head>
<body>
     <header>
          <h1>Mini Plataforma</h1>
     </header>
</body>
</html>
```

II. Crear el enlace de la ruta en *urls.py*.

```
urlpatterns = patterns('',
    url(r'^$', 'clases.views.home', name='home'),
)
```

III. Crear la *view* (*home*) en el módulo de *clases*.

```
from django.shortcuts import render_to_response

def home(request):
     return render_to_response('clases.html')
```

IV. Probar que todo esté correctamente *enlazado*.
V. Crear la estructura principal de las clases.

```
<section id="wrapper">
          <section id="libro">
               <article id="clases">
               </article>
               <article id="contenido-clase"></article>
          </section>
</section>
```

VI. Crear el *modelo* de clases en *models.py*.

```
from django.db import models

class Clases(models.Model):
     nombre = models.CharField(max_length=255)
     descripcion = models.TextField()
     url = models.CharField(max_length=100)
     thumb = models.CharField(max_length=100)

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

VII. *Sincronizar* la base de datos para que Django cree las *tablas*.

```
$ python manage.py syncdb
```

VIII. Abrir el *shell* para crear los *datos* de las clases.

A. Abrir el shell.

```
$ python manage.py shell
```

B. Crear los datos.

```
 >>> from clases.models import Clases

c1 = Clases(nombre="Sublime Text 2", descripcion="Clase de Sublime Text 2
donde aprenderemos consejos y trucos de este gran editor",
url="_gs_EIPkMVo", thumb="sublime2.png")

c2 = Clases(nombre="Motores Render de los Browsers",
descripcion="Descripcion de cuales son los motores render que usan
actualmente los navegadores y su importancia", url="hfGVnq7to0w",
thumb="motores.png")

c3 = Clases(nombre="Mide y organiza tus proyectos web como la gente
atractiva", descripcion="Aprende a organizar y optimizar tus proyectos de
forma correcta", url="hC_blYTGYhY", thumb="organiza.png")

c4 = Clases(nombre="La importancia del Responsive Design
@LeonidasEsteban", descripcion="Charla de @LeonidasEsteban sobre la
importancia del Responsive Design en la web moderna", url="YnphMCRsdXM",
thumb="responsive.png")

c1.save()
c2.save()
c3.save()
```

IX. *Importar* el modelo *Clases* en la *vista* de clases.
X. Traer *todas* las *clases* y pasarlas a la *vista*.

```
from clases.models import Clases
from django.shortcuts import render_to_response

def home(request):
     clases = Clases.objects.all()
     return render_to_response('clases.html', {'clases': clases})
```

XI. *Agregar* el *for* en *clases.html* para mostrar las clases.

```
<article id="clases">
     <ul>
          {% for clase in clases %}
          <li>
               <a data-id="{{ clase.id }}">
                    <figure>
                         <img src="/static/img/{{ clase.thumb }}">
                              <figcaption>{{ clase.descripcion }}</figcaption>
                    </figure>
               </a>
          </li>
          {% endfor %}
     </ul>
</article>
```

XII. *Agregar* las imágenes a la carpeta *static/img/*.
XIII. *Agregar* los archivos *css* a la carpeta *static/css* e incluirlos en *clases.html*.

```
<link rel="stylesheet" href="/static/css/normalize.css">
<link rel="stylesheet" href="/static/css/clases.css">
<link rel="stylesheet" href="/static/css/discusion.css">
```

XIV. Reiniciar servidor para cargar los archivos y las imágenes.
XV. Crear un nuevo *article* para la *sección* donde se va a cargar el *video* de la clase.

```
<article id="contenido-clase"></article>
```

XVI. Copiar los archivos *js* a la carpeta *static/js* e incluirlos en *clases.html*.

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/
jquery.min.js"></script>
<script src="/static/js/prefixfree.min.js"></script>
<script src="/static/js/main.js"></script>
```

XVII. *Agregar* la *url* a *urls.py* para el llamado *AJAX* de *cargar* el *contenido* de la *clase*.

```
url(r'^cargar-contenido-clase/(?P<id>\d+)$',
'clases.views.cargar_clase'),
```

XVIII. *Definir* en la vista de *clases* (*views.py*) la *función* que *carga* el *contenido* de la clase.

A. Importar *json*, *HttResponse* y *Http404*

```
import json

from django.http import HttpResponse, Http404
from django.shortcuts import render_to_response

def cargar_clase(request, id):
     if request.is_ajax():
          clase = Clases.objects.get(pk=id)
          return HttpResponse(
               json.dumps({'nombre': clase.nombre, 'descripcion':
clase.descripcion, 'url': clase.url }),
               content_type="application/json; charset=uft8"
               )
     else:
          raise Http404
```

APP DISCUSION

I. Crear *discusiones.html* dentro de *templates* del módulo *discusion*.

```
<article id="preguntas">
     <div id="crear-pregunta">
          {% csrf_token %}
          <input type="text" placeholder="Escribe tu pregunta">
          <button>Enviar Pregunta</button>
     </div>
</article>
<article id="respuestas"></article>
```

II. Crear el *modelo* de *discusiones* en *models.py* de la app discusion.

```
from django.db import models

class Preguntas(models.Model):
     titulo = models.CharField(max_length=255)

     def __unicode__(self):
          return self.titulo

class Respuestas(models.Model):
     titulo = models.CharField(max_length=255)
     pregunta = models.ForeignKey(Preguntas)

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

III. Sincronizar la base de datos.

```
$ python manage.py syncdb
```

IV. Crear el *tag* que se va a usar para *incluir* el módulo de *discusion* en el de *clases*.
   A. Crear una *carpeta* llamada *templatetags* dentro de *discusion*.
   B. Crear dentro, un *archivo vacío* llamado *__init__.py* para que se pueda *importar*.

V. Crear el archivo *discusiones_tags.py* donde se guardaran los *tags*.
   A. Lo primero es *importar template* desde *django* por que necesitamos registrar nuestro tag dentro de la librería de tags de Django.

```
from django import template
```

   B. Instanciar el objeto *Library* de template que es donde registramos nuestros tags y filters.

```
register = template.Library()
```

   C. Para *registrar* el *tag* se puede hacer de *2* formas.
      1. Crear la función y luego registrar el tag.

```
from discusion.models import Preguntas

def show_discusiones(context):
     preguntas = Preguntas.objects.all().order_by('-id')

     return {'preguntas': preguntas}

register.inclusion_tag('discusiones.html')(show_discusiones)
```

2. Usar un decorator.

```
from discusion.models import Preguntas

@register.inclusion_tag('discusiones.html', takes_context=True)
def show_discusiones(context):
     preguntas = Preguntas.objects.all().order_by('-id')

     return {'preguntas': preguntas}
```

VI. Agregar el *tag* a *clases.html*.

A. Cargar los *tags* en el *head* de clases.html.

```
<head>
     {% load discusiones_tags %}
     ....
<head>
```

B. Crear un nuevo *section* donde incluimos el *tag* para cargar el módulo discusion.

```
<section id="side">
     {% show_discusiones %}
</section>
```

VII. Agregar *discusiones.js* en *clases.html*.

```
<head>
     .....
     <script src="/static/js/discusion.js"></script>
</head>
```

VIII. Agregar *guardar-pregunta/* a *urls.py*.

```
url(r'^guardar-pregunta/$', 'discusion.views.guardar_pregunta'),
```

IX. Crear *guardar_pregunta* en *views.py* del módulo discusion.

A. Debe generar un error de *csrf*.

```
import json

from django.http import HttpResponse, Http404
from discusion.models import Preguntas, Respuestas

def guardar_pregunta(request):
     if request.is_ajax():

          if request.POST['pregunta']:
               pregunta = Preguntas(titulo=request.POST['pregunta'])
               pregunta.save()

          preguntas = Preguntas.objects.all().order_by('-id')

          data = list()
          for pregunta in preguntas:
               data.append({ 'id': pregunta.pk, 'titulo':
pregunta.titulo })

          return HttpResponse(
               json.dumps({ 'preguntas': data }),
               content_type="application/json; charset=uft8"
               )
     else:
          raise Http404
```

B. Una solución rápida es agregar un *RequestContext* en *views.py* de *clases*.

```
from django.template import RequestContext

def home(request):
     clases = Clases.objects.all()
     return render_to_response('clases.html', {'clases': clases},
context_instance=RequestContext(request))
```

X. Agregar el *for* de *preguntas* a *discusiones.html*.

```
<article id="preguntas">
     .....
     <ul>
          {% for pregunta in preguntas %}
          <li><a data-id="{{ pregunta.id }}">{{ pregunta.titulo }}</a></li>
          {% endfor %}
     </ul>
</article>
```

XI. Crear un nuevo *section* para *cargar* las *respuestas* de las preguntas en *discusiones.html*.

```
<article id="preguntas">
     .....
</article>
<article id="respuestas"></article>
```

XII. Agregar en *urls.py* la url para *cargar* las *respuestas*.

```
url(r'^cargar-respuestas/(?P<id>\d+)$',
'discusion.views.cargar_respuestas'),
```

XIII. Definir la *función* para *cargar* las *respuestas* en *views.py* de discusion.

A. No olvidar importar Respuestas.

```
def cargar_respuestas(request, id):
     if request.is_ajax():
          respuestas =
Respuestas.objects.filter(pregunta__id=id).order_by('-id')

          data = list()
          for respuesta in respuestas:
               data.append(respuesta.titulo)

          return HttpResponse(
               json.dumps({'respuestas': data, 'pregunta': id}),
               content_type="application/json; charset=uft8"
               )
     else:
          raise Http404
```

XIV. Agregar la *url* de *guardar respuestas* en *urls.py*.

```
url(r'^guardar-respuesta/$', 'discusion.views.guardar_respuesta'),
```

XV. Definir la *función* de *guardar respuesta* en *views.py* de discusion.

```
def guardar_respuesta(request):
     if request.is_ajax():

          if request.POST['respuesta']:
               respuesta =
Respuestas(titulo=request.POST['respuesta'],
pregunta_id=request.POST['pregunta'])
               respuesta.save()

          return cargar_respuestas(request, request.POST['pregunta'])
```