

# Django / Vue 3

dj






# Qué vamos a usar

<b>Python</b>	<b>3.12.4</b>
<b>Django</b>	<b>5.1.5</b>
<b>Vue 3</b>	<b>3.5.13</b>
<b>Bulma CSS</b>	<b>1.0.2</b>
<b>Nada más</b>	<b>0.0.0</b>

# Bulma CSS

 **BULMA**

[Docs](#) [Expo](#) [Love](#) [Sponsor](#) [GitHub](#) [Twitter](#) [...](#)

[→ Start Here](#)

[Overview](#)

[Installation](#)


[Alternative Versions](#)

[Syntax](#)

[Modularity](#)

[Responsiveness](#)

[Migrating to v1](#)

 **Features**


[CSS Variables](#)

[Themes](#)

[Dark Mode](#)

[Color Palettes](#)

[Skeletons](#)


 **Sass Tools**

[Mixins](#)

[Responsive mixins](#)

[Form Control mixins](#)

[Extends](#)

 **Customization**

[Concepts](#)

## Overview of Bulma


You only need **1 CSS file** to use Bulma

Bulma is a **CSS library**. This means it provides CSS classes to help you style your HTML code.

To use Bulma, you can either:

- use one of the pre-compiled `.css` files
- or install the `.scss` files so you can customize Bulma to your needs

### # Code requirements



Design and your inbox.




# Creamos un proyecto



```
$ mkdir Django-Vue && cd Django-Vue
$ pyenv virtualenv 3.12.4 dj-vue
$ pyenv activate dj-vue
$ echo "django==5.1.5" >> requirements.txt
$ pip install --upgrade pip
$ pip install -r requirements.txt
$ django-admin startproject main
$ mv main demo
$ cd demo
$ python3 manage.py check
```

# La app commons



```
$ python3 manage.py startapp commons
$ # Add Bulma.css
$ mkdir -p commons/static/commons/css
$ cd commons/static/commons/css
$ wget https://cdn.jsdelivr.net/npm/bulma@1.0.2/css/bulma.min.css -o bulma.css
$ cd ../../../../..
# # Add Vue.js
$ mkdir -p commons/static/commons/js
$ cd commons/static/commons/js
$ wget https://unpkg.com/vue@3/dist/vue.global.js -o vue3.js
$ cd ../../../../..
# # Add templates
$ mkdir -p commons/templates/commons/
$ touch commons/templates/commons/base.html
$ touch commons/templates/commons/homepage.html
```



# Disclaimer

Hay **varias formas** de usar Vue enDjango:

## **Con Node:**


Muy adecuado si tu infraestructura de Backend está basada en Javascript, o si estas desarrollando una aplicación Single-Page Application. Toda la potencia de Vue. Requiere instalar npm y toda la complejidad que implica

## **Usando directamente el código de Vue:**

Más sencillo, solo tenemos que incluir el fichero en nuestros contenidos estáticos.

Aquí **sólo veremos la segunda forma**

# Hola Django, Vue style



```
{% extends "commons/base.html" %}

{% block content %}
<div id="app"> {{ message }} </div>
{% endblock content %}

{% block javascript %}
<script charset="utf-8">
var app = Vue.createApp({
  'data': function() {
    return { 'message': 'Hola, Vue' };
  },
});
app.mount('#app')
</script>
{% endblock javascript %}
```



# No funciona





# Por qué no funciona

Vue utiliza por defecto el mismo sistema para presentar los datos que Django, las dobles llaves:

```
{{ message }}
```

El sistema de plantillas de Django (que no sabe nada de Vue) intenta presentar una variable `message`, pero **no existe** en el contexto, así que no imprime nada. Para cuando Vue se ejecuta en el cliente, la marca `{{ variable }}` ha sido **eliminada**, así que Vue se inicializa, pero no hace nada más.

Para evitarlo, *podríamos* escapar en el sistema de plantillas de Django las llaves, para que le lleguen a vue:

```
{% verbatim %}{{ variable }}{% endverbatim %}
```

# Fuerte rollo





# Usando delimiters

Para usar Vue.js desde Django con comodidad, hay un parámetro en la configuración, **delimiters** que permite usar como delimitadores los que tú especifiques, yo uso `[[ y ]]`:

```
var app = Vue.createApp({
  'delimiters': ['[[', ']]'],
  'data': function() {
    return {
      'message': 'Hello Django, from Vue',
    }
  },
});
app.mount('#app');
```

# Observese que...

- Al montar Vue especificamos el elemento en el que puede actuar, en este caso, `#app`. Vue no interactuará ni afectará nada que no esté dentro de ese elemento.
- Ya podemos usar los delimitadores `[[ y ]]`
- La página es reactiva



# Truco extra

En la carga de la página, se pueden ver, las marcas de plantilla hasta que Vue haya terminado de cargarse e inicializarse. Queda Feo.

Para resolverlo, Vue busca todos los elementos con la directiva v-cloak, y cuando ya esté completamente operativo quita la directiva de todos ellos.

En resumen, añadir la directiva v-cloak a los elementos que queremos ocultar inicialmente, e incluir en la hoja de estilos:

```
[v-cloak] { display:none; }
```

# Un contador con Vue

- Podemos añadir métodos al objeto vue.
- Invocamos a los métodos usando la directiva `v-click` o `@click`.
- La variable **this** se usa para acceder a los datos definidos en la instancia de Vue (Los que se definieron con la función data)



# Ejercicio

- Implementar un segundo botón que nos permita decrementar el contador

# La directiva v-html



# La directiva v-bind

Podemos vincular un dato de nuestro modelo con un atributo; para ello usamos la directiva v-bind.

Lo más habitual es para cambiar la clase, aunque se puede modificar cualquier atributo

Se puede abreviar de la siguiente manera:

$$\text{v-bind:href} \cong \text{:href}$$

# Doble vinculación con v-model

- Podemos realizar una vinculación de doble sentido
- Los cambios en el modelo se reflejarán en la vista, y los cambios en la vista se reproducirán en el modelo
- para ello usamos la directiva **v-model**



# La directivas v-if y v-show

- Las dos permite controlar si un elemento aparece o no en la página
- La diferencia es que con v-if el elemento realmente estará o no en el árbol DOM, dependiendo del predicado, pero con v-show el elementos siempre estará, y será visible o no en función del predicado

# La directivas v-if y v-show

- Dentro de la condición podemos incluir expresiones en javascript (solo una línea)
- Si queremos que la directiva afecte a varias etiquetas, podemos agruparlas `<template v-if="...">` y `</template>`, estas desaparecen en el resultado final
- Existe la directiva v-else



# La directiva v-for

- Podemos usar la directiva v-for para recorrer una lista de datos
- La sintaxis usa la forma v-for="item in items"

# La directiva v-for

- Podemos obtener el índice con la sintaxis (item, index) in items
- También se puede iterar por las propiedades de un objeto
- Ojo con los métodos que no modifican un array, sino que devuelven uno nuevo
- Cuidado: Cambios por el índice no son detectados por Vue, usar siempre shift, unshift, pop, etc.



# La directiva v-on

- Podemos vincular acciones con métodos usando la directiva v-on:evento
- Tenemos que definir nuestros métodos en la entrada methods de Vue.

# La directiva v-on

- El contenido de v-on es javascript, pero lo normal es que sea una llamada a un método
- Existen modificadores del evento, en la forma v-on.event.modifier, por ejemplo v-on:keyup.enter solo se activa si la tecla pulsada es enter
  - Podemos abreviar v.bind:evento=... como @evento=...



# Pasar datos de Django a Vue

La primera idea puede ser la de usar el propio sistema de plantillas de Django, por ejemplo, haciendo:

```
{# DON'T DO THIS #}
```

```
<script>
```

```
    const username = "{{ username }}";
```

```
</script>
```



# Pasar datos de Django a Vue

Es mejor **evitar siempre esta opción**. Django por defecto escapa los valores en las plantillas, así que si username fuera, por ejemplo "Adam <3", la salida en la plantilla sería:

```
<script>  
  const username = "Adam &lt;3";  
</script>
```



# Pasar datos de Django a Vue

Otro peligro relacionado con esto sería la posibilidad de **inyección de código**:

```
{# DON'T DO THIS #}
```

```
<script>
```

```
    const greeting = `Hi {{ username }}`;
```

```
</script>
```

Un visitante malicioso podría incluir una comilla invertida para terminar el literal, y añadir su propio código



# Pasar datos de Django a Vue

Hay dos técnicas:

- Usar atributos en marcas Html para datos sencillos
- Usar ``json_script`` para datos complejos