

# Guía Técnica para la Selección e Integración de Frameworks Front-End con Django: Un Enfoque Basado en Métricas de Calidad del Software

## Introducción

En el desarrollo de aplicaciones web, la elección del framework frontend adecuado es una decisión clave que impacta directamente en la calidad del software. Dos de los aspectos más críticos en esta evaluación son la mantenibilidad y la usabilidad, ya que determinan la facilidad con la que un sistema puede ser modificado, ampliado y utilizado de manera eficiente a lo largo del tiempo.

La mantenibilidad, según la norma **ISO/IEC 25010:2011**, se refiere a la capacidad de un sistema para ser modificado con facilidad, garantizando su evolución sin comprometer la estabilidad del código. Factores como la modularidad, la complejidad del código, la facilidad para aplicar actualizaciones y la consistencia en la estructura son fundamentales para determinar qué tan sencillo será mantener un software en el tiempo.

Por otro lado, la usabilidad, definida en la **ISO 9241-11:2018**, mide la eficiencia, efectividad y satisfacción del usuario al interactuar con el sistema. En el contexto de frameworks frontend, esto implica evaluar qué tan rápido un desarrollador puede aprender y usar la herramienta, el tiempo requerido para implementar funcionalidades comunes y la optimización del rendimiento durante la ejecución de la aplicación.

Esta guía establece basado en estas métricas, aplicándolo a tres de los frameworks frontend más utilizados en la industria: React, Angular y Vue.js, en combinación con Django Rest Framework (DRF) como backend. A través de un análisis detallado, se compararán estos frameworks en términos de su facilidad de mantenimiento y experiencia de desarrollo, proporcionando una visión objetiva para la toma de decisiones tecnológicas.

Dirigida a desarrolladores, arquitectos de software y tomadores de decisiones, esta guía servirá como una referencia práctica para seleccionar el framework más adecuado según las necesidades del proyecto, priorizando estándares de calidad reconocidos a nivel internacional.

## Objetivo de la guía:

Proporcionar una guía para analizar la mantenibilidad y usabilidad de los principales frameworks frontend: React, Angular y Vue.js, en combinación con Django Rest Framework (DRF) como backend. Además, esta guía servirá como un recurso práctico para la implementación básica de cada uno de estos frameworks en conjunto con DRF, permitiendo a los desarrolladores comprender sus características, ventajas y consideraciones clave al integrarlos en sus proyectos.

## Alcance:

## Normativas utilizadas:

- ISO 9241-11:2018 (Usabilidad: eficiencia, efectividad y satisfacción).
- ISO/IEC 25010:2011 (Mantenibilidad: modificabilidad, analizabilidad, estabilidad y facilidad de prueba).

## Parámetros de Evaluación y Métricas

### Usabilidad (ISO 9241-11:2018)

La presente tabla establece métricas concretas para medir la usabilidad de diferentes frameworks, centrándose en aspectos esenciales como el tiempo de implementación, la cantidad de código necesario, la velocidad de aprendizaje, el uso de recursos del sistema y la facilidad de depuración. Estas métricas permiten comparar frameworks de manera objetiva, asegurando que la elección de una tecnología se base en criterios medibles y relevantes para el desempeño del equipo de desarrollo.

Usabilidad		
Eficiencia:	Efectividad:	Satisfacción:
Tiempo de implementación de funcionalidades básicas	Porcentaje de tareas completadas correctamente	Satisfacción del usuario
Cantidad de código necesario para implementar una funcionalidad (líneas de código)	Reutilización de componentes	Tasa de preferencia frente a otros Frameworks
Velocidad de aprendizaje inicial		
Uso de recursos del sistema durante la carga y renderizado		Percepción de productividad
Tiempo de depuración		

### Mantenibilidad (ISO/IEC 25010:2011)

La presente tabla establece métricas clave para medir la mantenibilidad de un framework, considerando aspectos como el tiempo necesario para modificar componentes, la cantidad de líneas de código alteradas, el impacto de los cambios en otros módulos, la consistencia en estilos y estructura, la facilidad para actualizar versiones y la complejidad del código.

Mantenibilidad
----------------

Analizabilidad	Estabilidad		
Soporte de arquitecturas			
Tiempo de aprendizaje del framework			
Documentación del framework			

# Definición del proyecto

## Introducción y Propósito del Proyecto

El proyecto **"Gestor de Tareas"** tiene como objetivo desarrollar una aplicación web o de escritorio básica que permita a un usuario administrar una lista de tareas personales. La funcionalidad principal se centrará en las operaciones **CRUD** (Crear, Leer, Actualizar, Eliminar) para las tareas, donde cada tarea estará definida por un título y una descripción.

El propósito es ofrecer una herramienta sencilla, intuitiva y eficiente para la organización básica de actividades pendientes, sin complejidades innecesarias.

## 2. Alcance del Proyecto

### Alcance:

- Creación de nuevas tareas con un título y una descripción.
- Visualización de la lista de todas las tareas existentes.
- Visualización de los detalles (título y descripción) de una tarea específica.
- Modificación del título y/o descripción de una tarea existente.
- Eliminación de tareas existentes.
- Persistencia de los datos (las tareas deben guardarse incluso si la aplicación se cierra y se vuelve a abrir).

### Fuera del Alcance:

- Autenticación de usuarios y múltiples cuentas de usuario.
- Asignación de fechas de vencimiento, prioridades o categorías a las tareas.
- Notificaciones o recordatorios.
- Funcionalidades de colaboración o compartición de tareas.
- Adjuntar archivos a las tareas.
- Interfaz de usuario avanzada o personalizable.
- Sincronización entre múltiples dispositivos.

## Objetivos del Proyecto

1. Desarrollar un sistema que permita la creación de tareas con título y descripción.
2. Implementar la funcionalidad para listar todas las tareas existentes.
3. Permitir la edición de los campos (título y descripción) de una tarea existente.
4. Facilitar la eliminación de tareas que ya no son necesarias.
5. Asegurar que la interfaz de usuario sea intuitiva y fácil de usar para las operaciones CRUD.

6. Garantizar que los datos de las tareas se almacenen de forma persistente.

## Requisitos

### Aplicación Web

- **Sistema Operativo (SO):**
  - Windows 10 o superior, macOS 10.15 (Catalina) o superior, Distribuciones Linux modernas (ej. Ubuntu 20.04+, Fedora 34+)
- **Navegador Web:**
  - Google Chrome, Mozilla Firefox, Microsoft Edge, Safari (últimas 2 versiones en macOS)
- **Memoria RAM:**
  - Mínimo: 4 GB (el navegador y el SO consumen la mayor parte; la aplicación en sí será ligera).
  - Recomendado: 8 GB para una experiencia fluida con otras aplicaciones abiertas.
- **Procesador (CPU):**
  - Cualquier procesador dual-core moderno (ej. Intel i3, AMD Ryzen 3 o equivalentes de los últimos 5-7 años). La aplicación no será intensiva en CPU.
- **Resolución de Pantalla:**
  - Mínimo: 1024x768 píxeles.
  - Recomendado: 1280x720 píxeles o superior para una mejor visualización.
- **Conexión a Internet:**
  - Requerida para acceder a la aplicación si está alojada en un servidor.
  - La velocidad no necesita ser alta, ya que la transferencia de datos (título y descripción) es mínima.
- **Espacio en Disco:**
  - No se requiere espacio adicional significativo en el cliente, más allá del caché del navegador.

Requisitos	
Funcionales	No Funcionales
<ul style="list-style-type: none"><li>• <b>RF001:</b> El sistema debe permitir al usuario introducir un título (obligatorio) y una descripción</li></ul>	<ul style="list-style-type: none"><li>• <b>RNF001 (Usabilidad):</b> La interfaz de usuario debe ser simple, clara y fácil de entender, permitiendo realizar las</li></ul>

(opcional) para crear una nueva tarea.	operaciones CRUD con un mínimo de pasos.
<ul style="list-style-type: none"> <li>• <b>RF002:</b> El sistema debe mostrar una lista de todas las tareas creadas. Para cada tarea en la lista, se mostrará al menos el título.</li> </ul>	<ul style="list-style-type: none"> <li>• <b>RNF002 (Rendimiento):</b> La aplicación debe responder rápidamente a las acciones del usuario (crear, listar, actualizar, eliminar) para una cantidad moderada de tareas (ej. &lt; 1000 tareas).</li> </ul>
<ul style="list-style-type: none"> <li>• <b>RF003:</b> El sistema debe permitir al usuario seleccionar una tarea de la lista para ver sus detalles completos (título y descripción).</li> </ul>	<ul style="list-style-type: none"> <li>• <b>RNF003 (Fiabilidad):</b> Los datos de las tareas no deben perderse ni corromperse debido a errores comunes de la aplicación.</li> </ul>
<ul style="list-style-type: none"> <li>• <b>RF004:</b> El sistema debe permitir al usuario modificar el título y/o la descripción de una tarea existente.</li> </ul>	<ul style="list-style-type: none"> <li>• <b>RNF004 (Mantenibilidad):</b> El código fuente deberá estar organizado y comentado de forma que facilite futuras modificaciones o correcciones.</li> </ul>
<ul style="list-style-type: none"> <li>• <b>RF005:</b> El sistema debe permitir al usuario eliminar una tarea existente.</li> </ul>	
<ul style="list-style-type: none"> <li>• <b>RF006:</b> El sistema debe solicitar confirmación al usuario antes de eliminar una tarea.</li> </ul>	
<ul style="list-style-type: none"> <li>• <b>RF007:</b> El sistema debe almacenar las tareas de forma que persistan entre sesiones de uso.</li> </ul>	

#### Actores del Sistema

- **Usuario:** Persona que interactúa con el sistema para gestionar sus tareas. Es el único actor en esta versión.

#### Historias de usuario

<b>No:</b>	<b>HU01</b>	<b>Nombre: Crear Tarea</b>	<b>Fecha:16/06/2025</b>
<b>Actor:</b>	Usuario		
<b>Descripción:</b>	El usuario desea añadir una nueva tarea a su lista.		
<b>Flujo:</b>	<ol style="list-style-type: none"> <li>1. El usuario selecciona la opción "Crear Tarea".</li> <li>2. El sistema presenta un formulario para ingresar el título y la descripción.</li> </ol>		

	3. El usuario ingresa el título (obligatorio) y la descripción (opcional). 4. El usuario confirma la creación. 5. El sistema guarda la nueva tarea y actualiza la lista de tareas.	
<b>Personal realizo:</b>	<b>quien</b> Axel Omar Salazar Guarneros	<b>Firma:</b>
<b>Condición de Éxito:</b>	La nueva tarea es visible en la lista de tareas.	
<b>Personal Reviso:</b>	<b>quien</b>	<b>Firma:</b>

<b>No:</b>	<b>HU02</b>	<b>Nombre: Listar Tareas</b>	16/06/2025
<b>Actor:</b>	Usuario		
<b>Descripción:</b>	El usuario desea ver todas las tareas que ha creado.		
<b>Flujo:</b>	1. El usuario accede a la vista principal de la aplicación. 2. El sistema muestra una lista de todas las tareas existentes, mostrando al menos el título de cada una.		
<b>Personal realizo:</b>	<b>quien</b> Axel Omar Salazar Guarneros	<b>Firma:</b>	
<b>Condición de Éxito:</b>	El usuario puede ver sus tareas.		
<b>Personal Reviso:</b>	<b>quien</b>	<b>Firma:</b>	

<b>No:</b>	<b>HU03</b>	<b>Nombre: Ver Detalle de Tarea</b> (Puede estar implícito en "Actualizar Tarea")	16/06/2025
<b>Actor:</b>	Usuario		
<b>Descripción:</b>	El usuario desea ver la información completa de una tarea específica.		
<b>Flujo:</b>	1. El usuario selecciona una tarea de la lista. 2. El sistema muestra el título y la descripción completa de la tarea seleccionada.		
<b>Personal realizo:</b>	<b>quien</b> Axel Omar Salazar Guarneros	<b>Firma:</b>	

<b>Condición de Éxito:</b>	El usuario ve los detalles de la tarea.	
<b>Personal      quien Reviso:</b>		<b>Firma:</b>

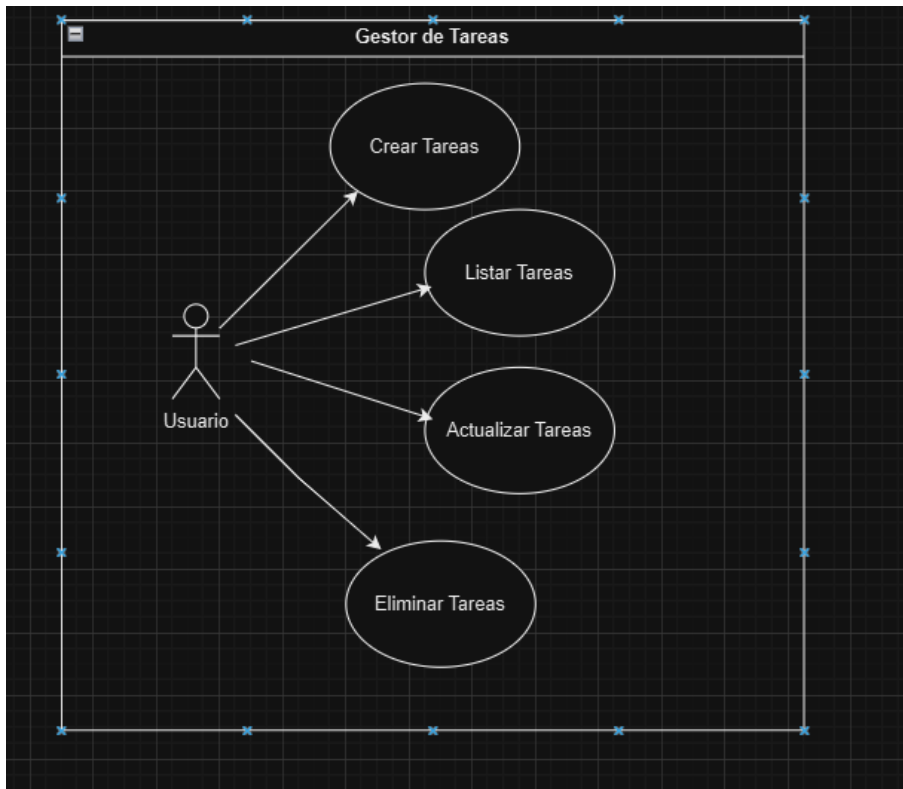
No:	HU04	Nombre: Actualizar Tarea	16/06/2025
Actor:	Usuario		
Descripción:	El usuario desea modificar la información de una tarea existente.		
Flujo:	<div>1. El usuario selecciona la tarea que desea modificar (o la opción "Editar" asociada a una tarea).</div> <div>2. El sistema presenta un formulario con los datos actuales de la tarea (título y descripción).</div> <div>3. El usuario modifica el título y/o la descripción.</div> <div>4. El usuario confirma los cambios.</div> <div>5. El sistema actualiza la tarea con la nueva información y refresca la vista correspondiente.</div>		
Personal realizo:	quien	Axel Omar Salazar Guarneros	Firma:
Condición de Éxito:	La tarea se actualiza con la nueva información.		
Personal Reviso:	quien		Firma:

No:	HU05	Nombre: Eliminar Tarea	16/06/2025
Actor:	Usuario		
Descripción:	El usuario desea quitar una tarea de su lista.		
Flujo:	<div>1. El usuario selecciona la tarea que desea eliminar (o la opción "Eliminar" asociada a una tarea).</div> <div>2. El sistema solicita confirmación para la eliminación.</div> <div>3. El usuario confirma la eliminación.</div> <div>4. El sistema elimina la tarea de forma permanente y actualiza la lista de tareas.</div>		
Personal realizo:	quien Axel Omar Salazar Guarneros	Firma:	
Condición de Éxito:	La tarea ya no existe en el sistema.		



Personal Reviso:	quien		Firma:
---------------------	-------	--	--------

## Casos de Uso



## Diagrama de Clases



## Implementación Práctica con React.js, Angular y Vue.js

### Configuración del Entorno

## Django Rest Framework como backend.

### Paso 1: Crear un entorno virtual

```
(reactvenv) PS C:\Users\q-ani\Desktop\Proyectos para tesis\react\django_react_api> python -m venv reactvenv
```

### Paso 2: Instalar Django y Django Rest Framework

Con el entorno virtual activado, se instala Django y Django Rest Framework y Corseheaders.

```
(reactvenv) PS C:\Users\q-ani\Desktop\Proyectos para tesis\react\django_react_api> pip install django
Requirement already satisfied: django in c:\users\q-ani\desktop\proyectos para tesis\react\reactvenv\lib\site-packages (5.1.1)
Requirement already satisfied: asgiref<4,>=3.8.1 in c:\users\q-ani\desktop\proyectos para tesis\react\reactvenv\lib\site-packages (from django) (3.8.1)
Requirement already satisfied: sqlparse>=0.3.1 in c:\users\q-ani\desktop\proyectos para tesis\react\reactvenv\lib\site-packages (from django) (0.5.1)
Requirement already satisfied: tzdata in c:\users\q-ani\desktop\proyectos para tesis\react\reactvenv\lib\site-packages (from django) (2024.1)
```

```
(reactvenv) PS C:\Users\q-ani\Desktop\Proyectos para tesis\react\django_react_api> pip install djangorestframework
Requirement already satisfied: djangorestframework in c:\users\q-ani\desktop\proyectos para tesis\react\reactvenv\lib\site-packages (3.15.2)
Requirement already satisfied: django>=4.2 in c:\users\q-ani\desktop\proyectos para tesis\react\reactvenv\lib\site-packages (from djangorestframework) (5.1.1)
Requirement already satisfied: asgiref<4,>=3.8.1 in c:\users\q-ani\desktop\proyectos para tesis\react\reactvenv\lib\site-packages (from django>=4.2->djangorestframework) (3.8.1)
Requirement already satisfied: sqlparse>=0.3.1 in c:\users\q-ani\desktop\proyectos para tesis\react\reactvenv\lib\site-packages (from django>=4.2->djangorestframework) (0.5.1)
Requirement already satisfied: tzdata in c:\users\q-ani\desktop\proyectos para tesis\react\reactvenv\lib\site-packages (from django>=4.2->djangorestframework) (2024.1)
```

```
(reactvenv) PS C:\Users\q-ani\Desktop\Proyectos para tesis\react\django_react_api> pip install django-cors-headers
Requirement already satisfied: django-cors-headers in c:\users\q-ani\desktop\proyectos para tesis\react\reactvenv\lib\site-packages (4.4.0)
Requirement already satisfied: django>=3.6 in c:\users\q-ani\desktop\proyectos para tesis\react\reactvenv\lib\site-packages (from django-cors-headers) (3.8.1)
Requirement already satisfied: django>=3.2 in c:\users\q-ani\desktop\proyectos para tesis\react\reactvenv\lib\site-packages (from django-cors-headers) (5.1.1)
Requirement already satisfied: sqlparse>=0.3.1 in c:\users\q-ani\desktop\proyectos para tesis\react\reactvenv\lib\site-packages (from django>=3.2->django-cors-headers) (0.5.1)
Requirement already satisfied: tzdata in c:\users\q-ani\desktop\proyectos para tesis\react\reactvenv\lib\site-packages (from django>=3.2->django-cors-headers) (2024.1)
```

### Paso 3: Crear un proyecto Django

```
(reactvenv) PS C:\Users\q-ani\Desktop\Proyectos para tesis\react\django_react_api> django-admin startproject django_react_api
```

### Paso 4: Crear una aplicación Django

Dentro de tu proyecto, crea una nueva aplicación.

```
(reactvenv) PS C:\Users\q-ani\Desktop\Proyectos para tesis\react\django_react_api> python manage.py start app tasks
```

### Paso 5: Configurar Django Rest Framework con corseheaders

Abre el archivo `settings.py` en la carpeta `django\_react\_api` y agrega `rest\_framework` y tu aplicación (`tasks`) a la lista de `INSTALLED\_APPS`.

```

  25 # SECURITY WARNING: don't run with debug turned on in production!
  26 DEBUG = True
  27
  28 ALLOWED_HOSTS = []
  29
  30 # Application definition
  31
  32 INSTALLED_APPS = [
  33     'django.contrib.admin',
  34     'django.contrib.auth',
  35     'django.contrib.contenttypes',
  36     'django.contrib.sessions',
  37     'django.contrib.messages',
  38     'django.contrib.staticfiles',
  39     'corsheaders',
  40     'rest_framework',
  41     'tasks',
  42 ]
  43
  44
```

Y en el apartado de middleware agregaremos los componentes de Corsheaders

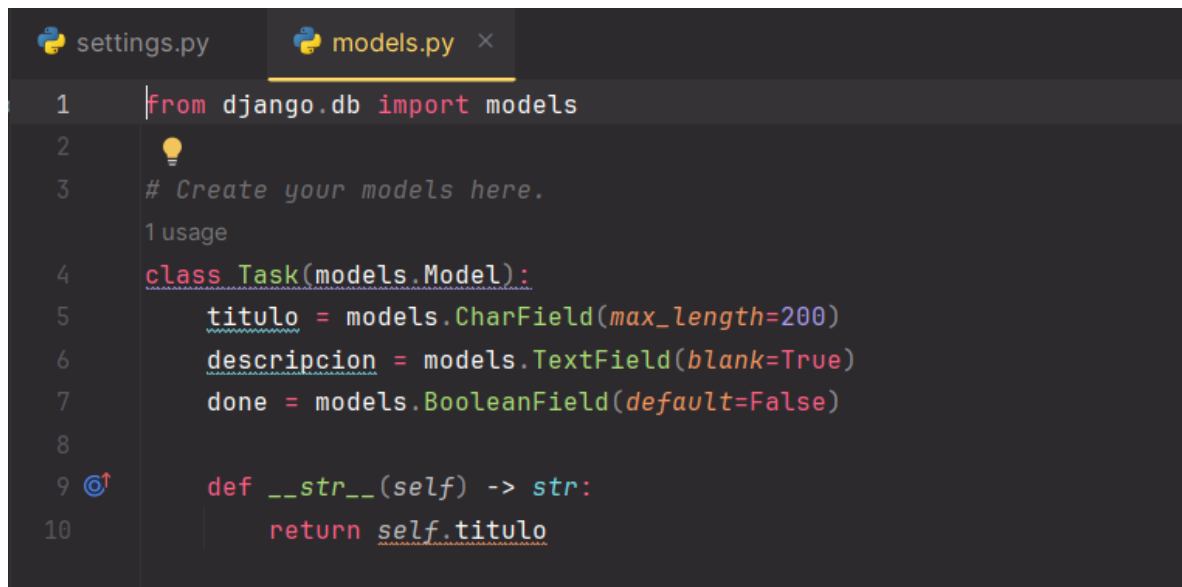
```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',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

también agregamos los `cors_allowed_origins` según los puertos que vamos a ocupar para el front-end

```
#cors auterization
CORS_ALLOWED_ORIGINS = [
    |
]
```

#### Paso 6: Crear un modelo

En la aplicación `tasks`, define un modelo en el archivo `models.py`.



```

1  from django.db import models
2
3  # Create your models here.
4  class Task(models.Model):
5      titulo = models.CharField(max_length=200)
6      descripcion = models.TextField(blank=True)
7      done = models.BooleanField(default=False)
8
9      def __str__(self) -> str:
10         return self.titulo

```

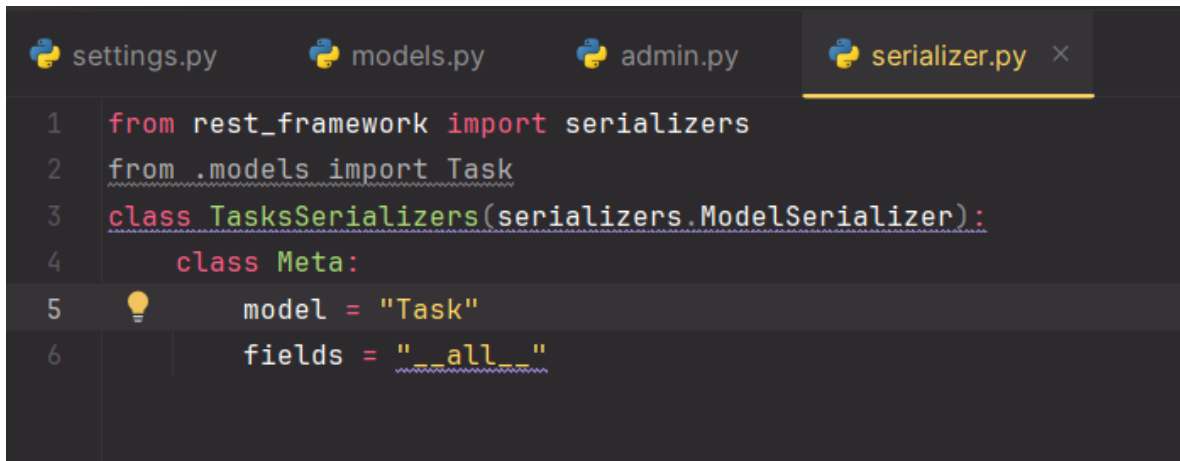
#### Paso 7: Crear y aplicar migraciones

Genera y aplica las migraciones para crear la tabla en la base de datos.

```
(reactvenv) PS C:\Users\q-ani\Desktop\Proyectos para tesis\react\django_react_api> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, tasks
Running migrations:
  No migrations to apply.
```

#### Paso 8: Crear un serializador

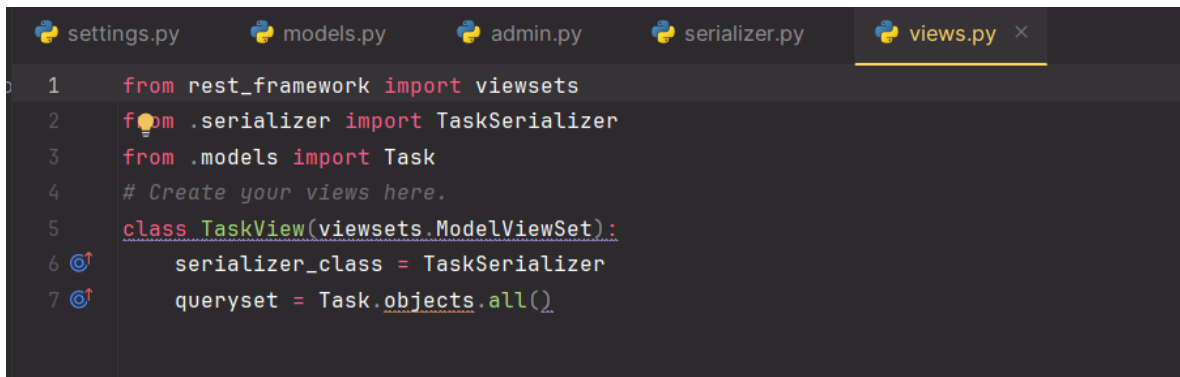
Creas un archivo `serializers.py` en la carpeta `tasks` para definir cómo se serializarán los objetos del modelo.



```
1 from rest_framework import serializers
2 from .models import Task
3 class TaskSerializers(serializers.ModelSerializer):
4     class Meta:
5         model = "Task"
6         fields = "__all__"
```

#### Paso 9: Crear una vista

En el archivo `views.py` de `tasks`, crea una vista para manejar las solicitudes HTTP.



```
1 from rest_framework import viewsets
2 from .serializer import TaskSerializers
3 from .models import Task
4 # Create your views here.
5 class TaskView(viewsets.ModelViewSet):
6     serializer_class = TaskSerializers
7     queryset = Task.objects.all()
```

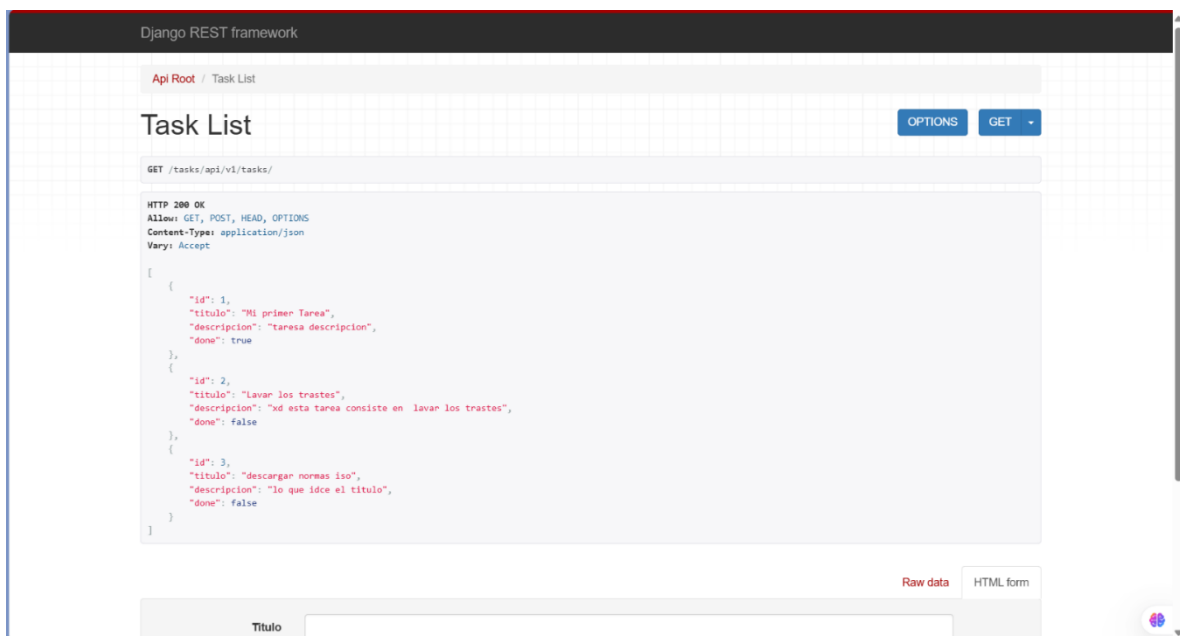
#### Paso 10: Configurar las URLs

En la carpeta `tasks`, crea un archivo `urls.py` y define las rutas para tu API.

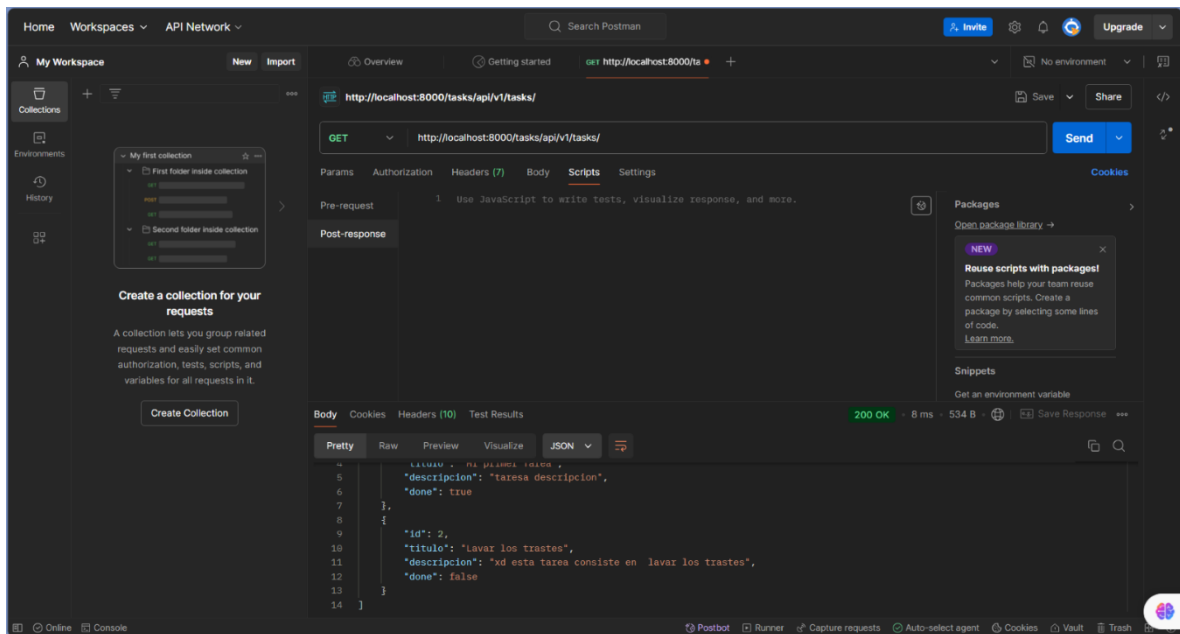
```
settings.py  models.py  admin.py  serializer.py  views.py  urls.py x
1  from django.urls import path, include
2  from rest_framework import routers
3  from .views import TaskView
4
5  router = routers.DefaultRouter()
6  router.register( prefix: r'tasks', TaskView, basename: 'tasks')
7
8  urlpatterns = [
9      path("api/v1", include(router.urls))
10 ]
```

**Paso 11:** Ejecutar el servidor de desarrollo

Finalmente, ejecuta el servidor de desarrollo.



**Paso 12:** Probar la API



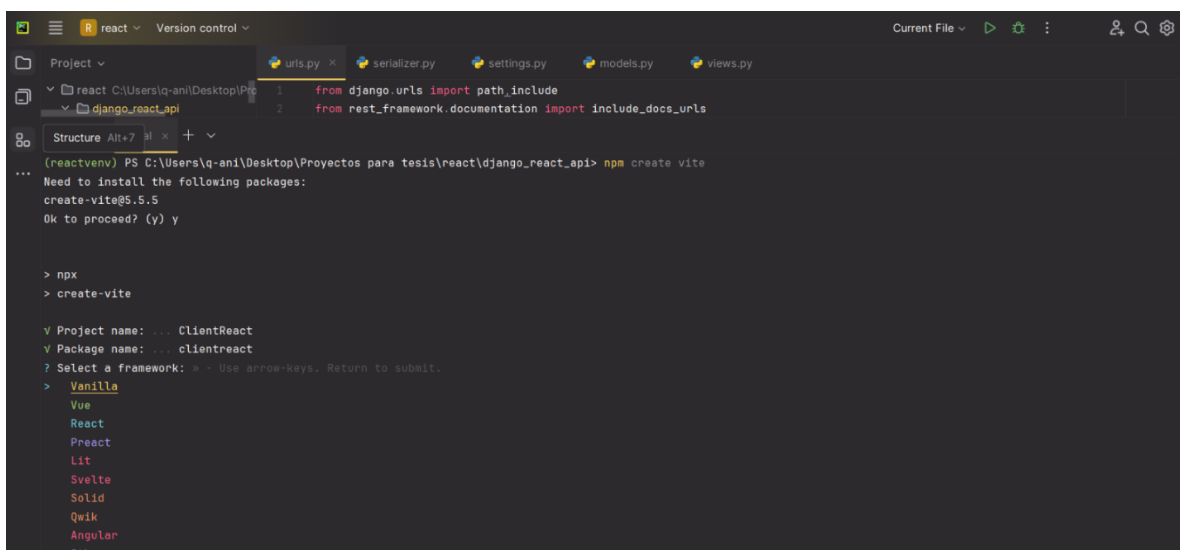
## Desarrollo de un Caso de Uso Común

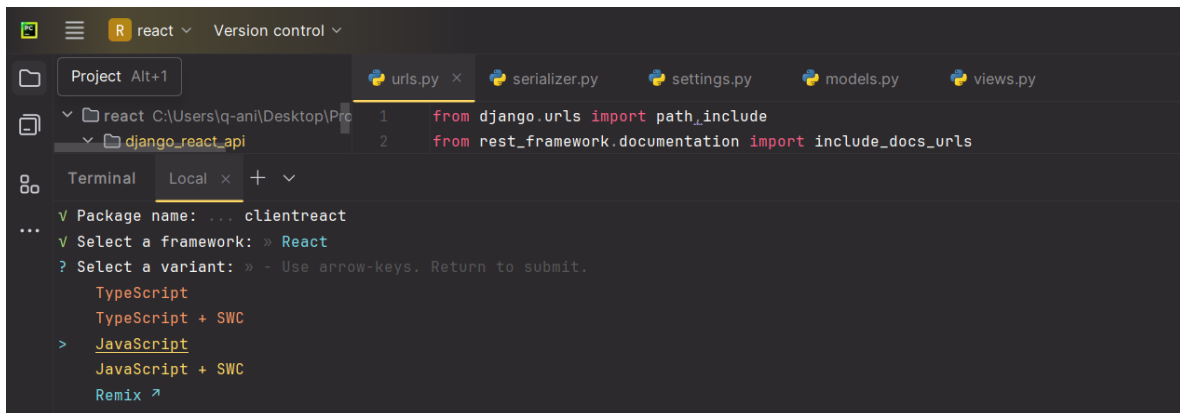
Creación de una aplicación CRUD para gestión de tareas con paginación, formularios y validaciones.

Implementación en cada framework (React.js, Angular, Vue.js) con consumo de API en DRF.

### React

#### Paso 1: Instalación de cliente React con Vitejs

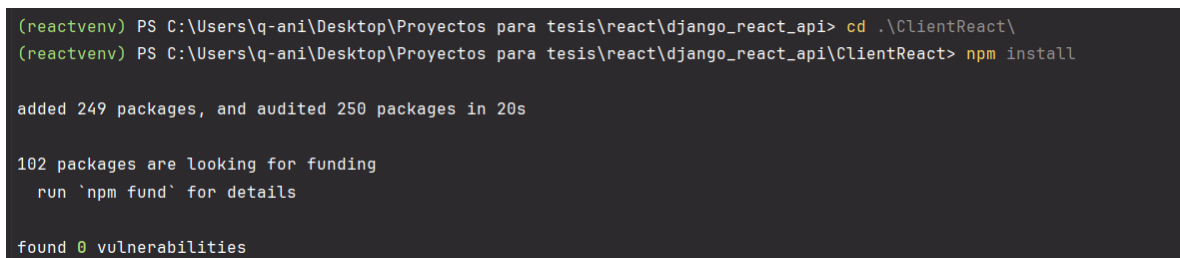




The screenshot shows the Visual Studio Code interface. The top bar indicates the 'react' extension is active. The Explorer sidebar on the left shows a project structure with a 'django\_react\_api' folder. The Editor pane displays two Python files: 'urls.py' and 'serializer.py'. The 'urls.py' file contains two lines of code: `from django.urls import path_include` and `from rest_framework.documentation import include_docs_urls`. The Terminal pane at the bottom shows the output of a command, displaying package names and framework selection options.

```
Project Alt+1
react C:\Users\q-ani\Desktop\Proyectos para tesis\react\django_react_api
  django_react_api
    urls.py
    serializer.py
    settings.py
    models.py
    views.py

Terminal Local x + v
...
V Package name: ... clientreact
V Select a framework: » React
? Select a variant: » - Use arrow-keys. Return to submit.
  TypeScript
  TypeScript + SWC
> JavaScript
  JavaScript + SWC
  Remix ↗
```



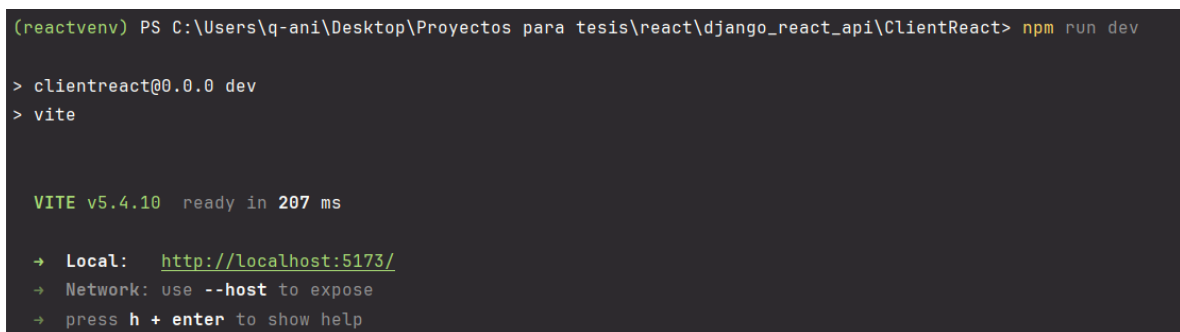
The screenshot shows a PowerShell terminal window with the following output:

```
(reactvenv) PS C:\Users\q-ani\Desktop\Proyectos para tesis\react\django_react_api> cd .\ClientReact\
(reactvenv) PS C:\Users\q-ani\Desktop\Proyectos para tesis\react\django_react_api\ClientReact> npm install

added 249 packages, and audited 250 packages in 20s

102 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```



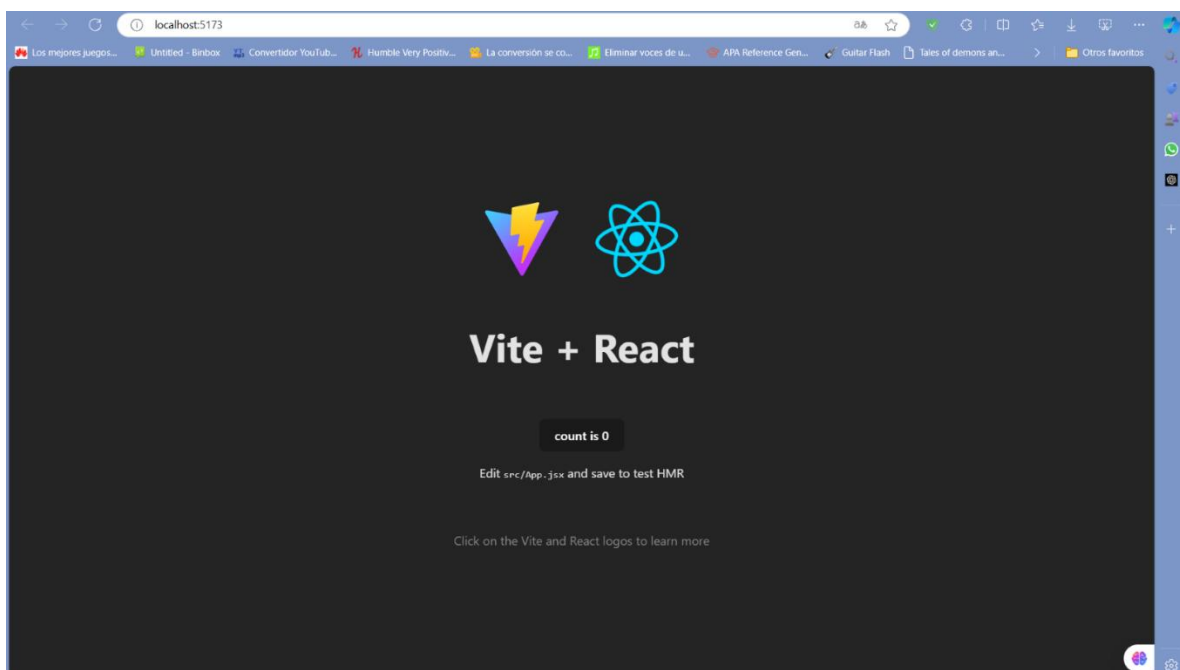
The screenshot shows a PowerShell terminal window with the following output:

```
(reactvenv) PS C:\Users\q-ani\Desktop\Proyectos para tesis\react\django_react_api\ClientReact> npm run dev

> clientreact@0.0.0 dev
> vite

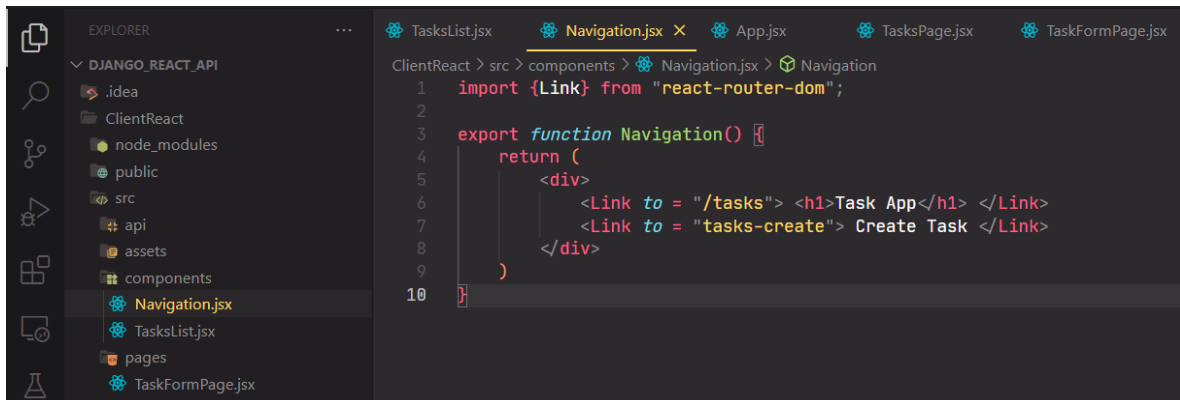
VITE v5.4.10 ready in 207 ms

→ Local: http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help
```



**Paso 2:** Agregamos 2 archivos a una nueva carpeta llamada components dentro de la carpeta src

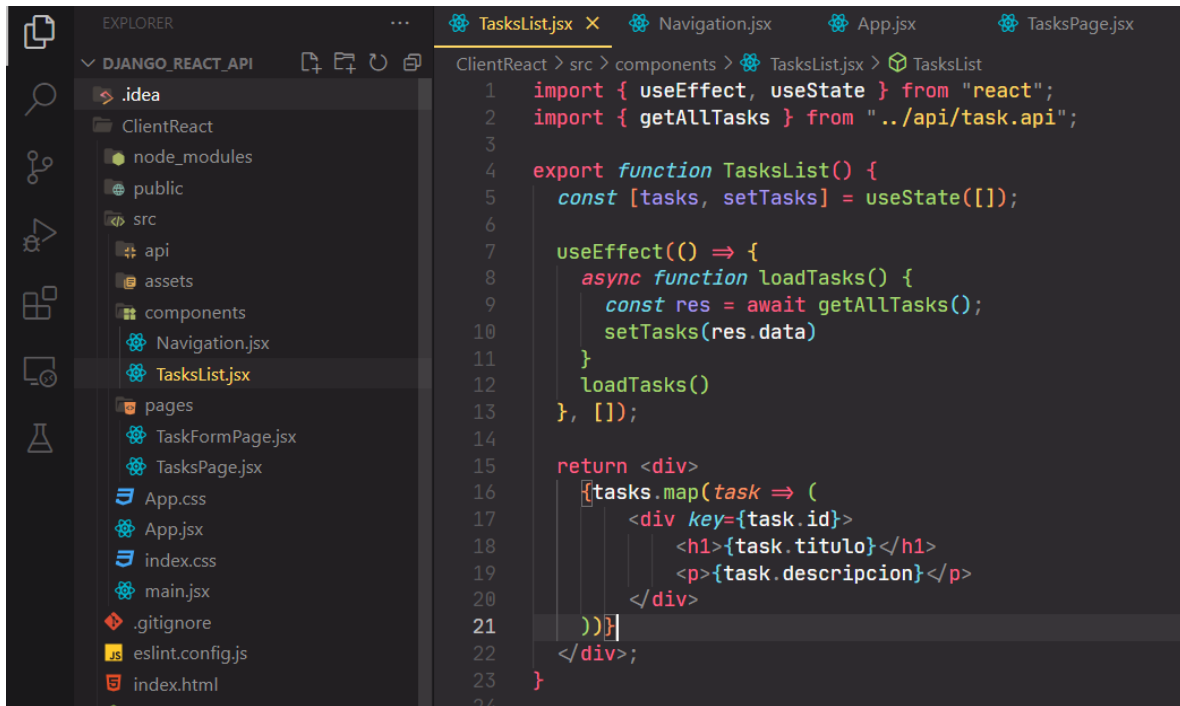
### Navigation



The screenshot shows the VS Code interface with the Explorer panel on the left displaying the project structure. The file 'Navigation.jsx' is selected in the 'components' folder. The main editor shows the code for 'Navigation.jsx' with the following content:

```
1 import {Link} from "react-router-dom";
2
3 export function Navigation() {
4   return (
5     <div>
6       <Link to = "/tasks"> <h1>Task App</h1> </Link>
7       <Link to = "tasks-create"> Create Task </Link>
8     </div>
9   )
10 }
```

### TaskList



The screenshot shows the VS Code interface with the Explorer panel on the left displaying the project structure. The file 'TasksList.jsx' is selected in the 'components' folder. The main editor shows the code for 'TasksList.jsx' with the following content:

```
1 import { useEffect, useState } from "react";
2 import { getAllTasks } from "../api/task.api";
3
4 export function TasksList() {
5   const [tasks, setTasks] = useState([]);
6
7   useEffect(() => {
8     async function loadTasks() {
9       const res = await getAllTasks();
10      setTasks(res.data)
11    }
12    loadTasks()
13  }, []);
14
15   return <div>
16     {tasks.map(task => (
17       <div key={task.id}>
18         <h1>{task.titulo}</h1>
19         <p>{task.descripcion}</p>
20       </div>
21     ))}
22   </div>;
23 }
```

**Paso 3:** Agregamos 2 archivos a una nueva carpeta llamada “pages” dentro de la carpeta “src”

### TaskList



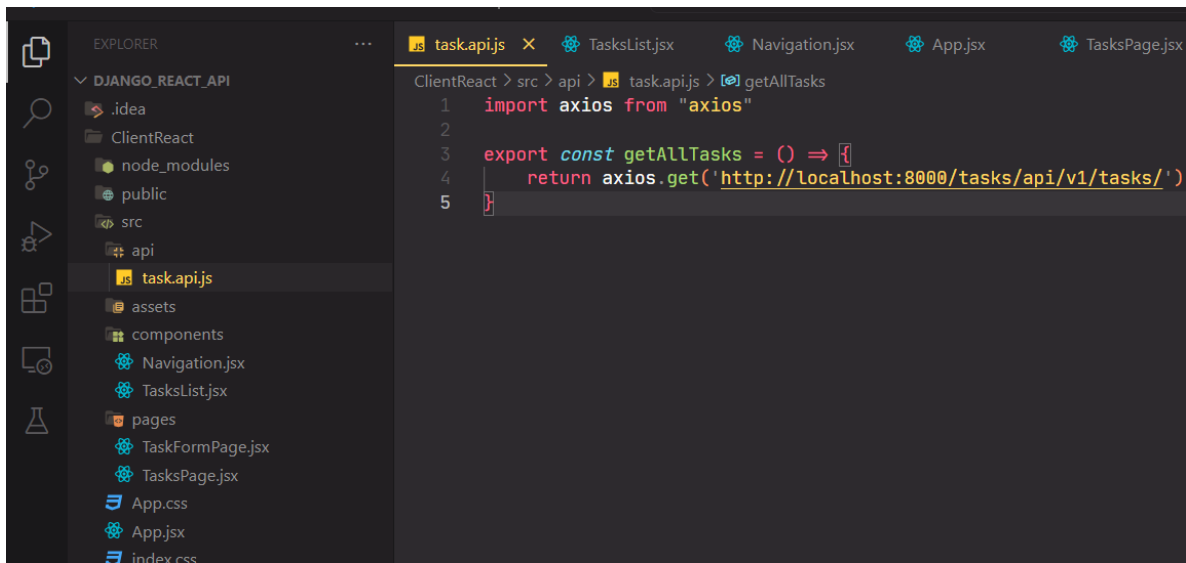
```
1 import { TasksList } from \"../components/TasksList\";
2
3 export function TasksPage() {
4   return <TasksList/>;
5 }
6
7
```

## TaskFormPage

```
1
2 export function TaskFormPage() {
3   return (
4     <div>TaskFormPage</div>
5   )
6 }
7
8
```

**Paso 4:** Agregamos 1 archivo a una nueva carpeta llamada “api” dentro de la carpeta “src”

## Taskapi



### Paso 5 redireccionamiento

En “TaskFormPage”

Importamos el hook usenavigate de React-router-dom

```
import { useNavigate } from "react-router-dom";
```

Lo definimos para poder usarlo

```
const navigate = useNavigate()
```

Navegamos al crear la tarea al listado de tareas

```
const onSubmit = handleSubmit((data) => {  
  createTask(data)  
  navigate('/tasks')  
})
```

### Paso 6 eliminación de tareas

Agregamos una nueva ruta para obtener la tarea seleccionada en app.jsx

```
<Routes>  
  <Route path="/tasks" element={<TasksPage />} />  
  <Route path="/tasks-create" element={<TaskFormPage />} />  
  <Route path="/tasks/:id" element={<TaskFormPage />} />  
</Routes>
```

Agregamos la navegación a una tarea en taskcard

```
TaskCard.jsx M X App.jsx M TasksList.jsx TaskFormPage.jsx M JS
Horus > Fronts-Django > ClientReact > src > components > TaskCard > TaskCard
1 | import { useNavigate } from "react-router-dom";
2
3 export function TaskCard({ task }) {
4 |   const navigate = useNavigate()
5 |   return(
6 |     <div
7 |       onClick={() => navigate(`/tasks/${task.id}`)}>
8 |       <h1>{task.titulo}</h1>
9 |       <p>{task.descripcion}</p>
10 |     </div>
11 |   );
12 }
```

Agregamos el hook useParams Para traer el id de la tarea

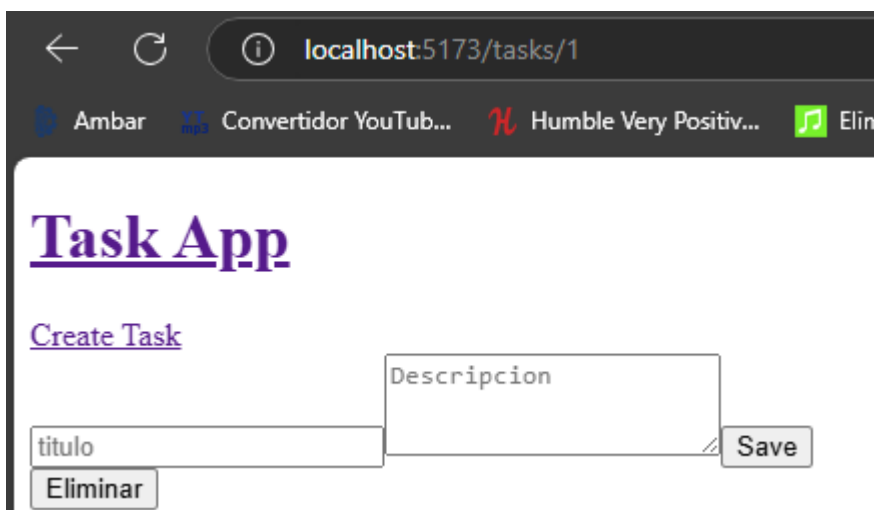
```
import { useNavigate, useParams } from "react-router-dom";
```

Lo definimos

```
const navigate = useNavigate()
const params = useParams()
```

Agregamos una condicional para el botón eliminar si existe el parámetro id entonces lo renderiza

```
{params.id && <button>Eliminar</button>}
```

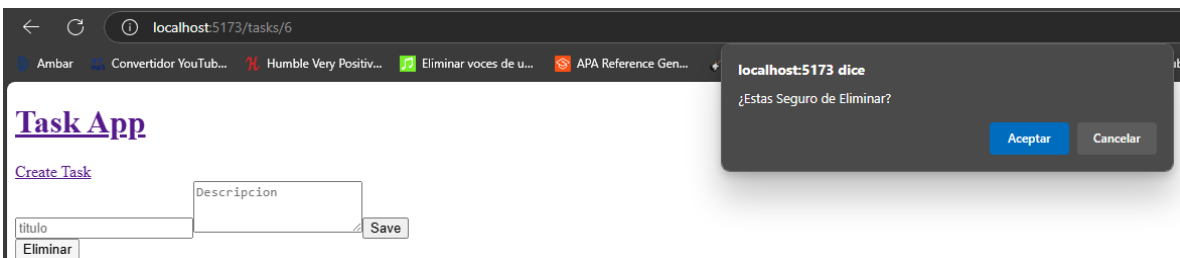
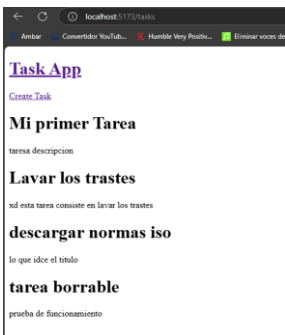


Agregamos al api la función de eliminar

```
export const deleteTask = (id) => tasksApi.delete(`/${id}/`);
```

Y al botón eliminar le agregamos la función deletetask

```
{params.id && <button onClick={async() => {  
  const confirm = window.confirm('¿Estas Seguro de Eliminar?')  
  if (confirm) {  
    await deleteTask(params.id)  
    navigate('/tasks')  
  }  
}}>Eliminar</button>}
```



## Paso 7 UpdateTask

Agregamos la función a la api de actualizar tarea y obtener tarea individual

```
export const updateTask = (id, task) => tasksApi.put(`/${id}/`, task);
```

```
export const getTask = (id) => tasksApi.get(`/${id}/`);
```

Se importa desde taskformpage e importamos el hook useeffect de react

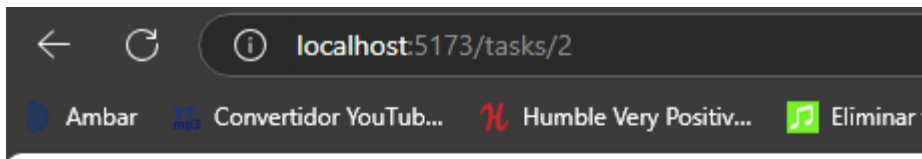
```
TaskCard.jsx M App.jsx M TasksList.jsx TaskFormPage.jsx 1, M JS task.api.js M TasksPa
Horus > Fronts-Django > ClientReact > src > pages > TaskFormPage.jsx > ...
1 import { useForm } from "react-hook-form";
2 import { useEffect } from "react";
3 import { createTask, deleteTask, updateTask, getTask } from '../api/task.api'
```

Agregamos setValue de useform()

```
export function TaskFormPage() {
  const {
    register,
    handleSubmit,
    formState: { errors },
    setValue,
  } = useForm();
```

Se define useeffect

```
useEffect(() => {
  async function loadTask() {
    if (params.id) {
      const {
        data: { titulo, descripcion },
      } = await getTask(params.id);
      setValue("titulo", titulo);
      setValue("descripcion", descripcion);
    }
  }
  loadTask();
}, []);
```



# Task App

## Create Task

xd esta tarea consiste en lavar los trastes

Save

Eliminar

Definimos updatetask

```
const onSubmit = handleSubmit(async (data) => {  
  if (params.id) {  
    await updateTask(params.id, data)  
  } else {  
    createTask(data)  
  }  
  navigate('/tasks')  
})
```

← ↻ ⓘ localhost:5173/tasks/2

Ambar Convertidor YouTub... Humble Very Positiv... Eliminar voces de

# Task App

## Create Task

compre lechuga :')

Save

Eliminar



## Paso 8 Agregando estilos (opcional) tailwind y toast

En el archivo app.jsx agregamos toaster de React-hot-toast

```
TaskCard.jsx M App.jsx M X TasksList.jsx TaskFormPage.jsx M JS task.ap
Horus > Fronts-Django > ClientReact > src > App.jsx > App
1 import { BrowserRouter, Routes, Route } from "react-router-dom";
2 import { TasksPage } from "../pages/TasksPage";
3 import { TaskFormPage } from "../pages/TaskFormPage";
4 import { Navigation } from "../components/Navigation";
5 import { Toaster } from "react-hot-toast";
6
7 function App() {
8   return (
9     <BrowserRouter>
10      <Navigation/>
11      <Routes>
12        <Route path="/tasks" element={<TasksPage />} />
13        <Route path="/tasks-create" element={<TaskFormPage />} />
14        <Route path="/tasks/:id" element={<TaskFormPage />} />
15      </Routes>
16      <Toaster/>
17    </BrowserRouter>
18  );
19 }
20
21 export default App;
```

Se importa en taskformpage

```
import { toast } from "react-hot-toast";
```

Se implementa después de crear, actualizar y eliminar tareas

```

const onSubmit = handleSubmit(async (data) => {
  if (params.id) {
    await updateTask(params.id, data)
    createTask(data)
    toast.success('Tarea Actualizada', {
      position: 'bottom-right',
      style: {
        background: '#101010',
        color: '#fff',
      }
    })
  } else {
    createTask(data)
    toast.success('Tarea Creada', {
      position: 'bottom-right',
      style: {
        background: '#101010',
        color: '#fff',
      }
    })
  }
  navigate('/tasks')
})

```

```

{params.id && <button onClick={async () => {
  const confirm = window.confirm('¿Estas Seguro de Eliminar?')
  if (confirm) {
    await deleteTask(params.id)
    createTask(data)
    toast.success('Tarea Eliminada', {
      position: 'bottom-right',
      style: {
        background: '#101010',
        color: '#fff',
      }
    })
  }
  navigate('/tasks')
}}>Eliminar</button>}

```



Instalación de tailwind



```

PS C:\Users\Axe1089\Desktop\Proyectos\Horus\Fronts-Django\ClientReact> npm install tailwindcss @tailwindcss/vite

added 12 packages, and audited 277 packages in 7s

106 packages are looking for funding
  run `npm fund` for details

6 vulnerabilities (1 low, 4 moderate, 1 high)

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
PS C:\Users\Axe1089\Desktop\Proyectos\Horus\Fronts-Django\ClientReact>

```

Se agrega al archivo vite

```

TaskCard.jsx M App.jsx M TasksList.jsx TaskFormPage.jsx M vite.config.js M X
Horus > Fronts-Django > ClientReact > vite.config.js > ...
1 import { defineConfig } from 'vite'
2 import react from '@vitejs/plugin-react'
3 import tailwindcss from '@tailwindcss/vite'
4
5 export default defineConfig({
6   plugins: [
7     react(),
8     tailwindcss(),
9   ],
10 })

```

Importamos en el archivo css

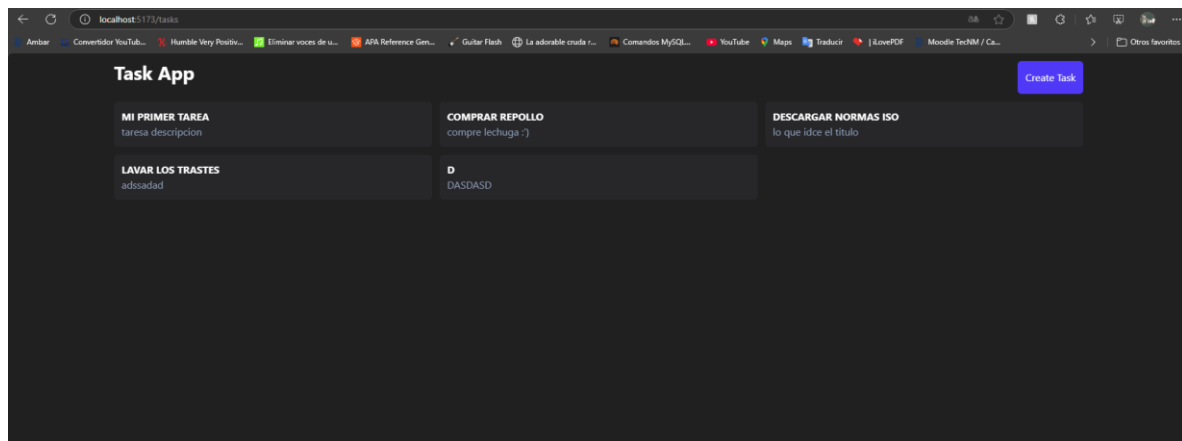
```

TaskCard.jsx M App.jsx M TasksList.jsx TaskFormPage.jsx M vite.config.js M App.css M X
Horus > Fronts-Django > ClientReact > src > App.css
1 @import "tailwindcss";

```

Ahora ya puedes ocupar tailwind

tasklist



taskformpage

Task App

Create Task

titulo

Descripcion

Save

Taskformpage modo edición

Task App

Create Task

Lavar los trastes

adssadad

Save

Eliminar

# Angular

**Paso 1:** Instalación y configuración de Angular en el entorno de desarrollo usando la herramienta ViteJs

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

● PS C:\Users\Axel089\Desktop\Proyectos\angular\django_angular_api> npm create vite

> npx
> create-vite

√ Project name: ... ClientAngular
√ Package name: ... clientangular
√ Select a framework: » Angular
√ Select a variant: » Angular ↗
Need to install the following packages:
@angular/cli@18.2.12
Ok to proceed? (y) y

Would you like to share pseudonymous usage data about this project with the Angular Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details and how to change this setting, see https://angular.dev/cli/analytics.

no
Global setting: disabled
Local setting: No local workspace configuration file.
Effective status: disabled
? Which stylesheet format would you like to use? CSS [ https://developer.mozilla.org/docs/Web/CSS
? Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? no
CREATE ClientAngular/angular.json (2715 bytes)
CREATE ClientAngular/package.json (1086 bytes)
CREATE ClientAngular/README.md (1102 bytes)
CREATE ClientAngular/tsconfig.json (1045 bytes)
CREATE ClientAngular/.editorconfig (331 bytes)
CREATE ClientAngular/.gitignore (629 bytes)
CREATE ClientAngular/tsconfig.app.json (439 bytes)
CREATE ClientAngular/tsconfig.spec.json (449 bytes)
CREATE ClientAngular/.vscode/extensions.json (134 bytes)
CREATE ClientAngular/.vscode/launch.json (490 bytes)
CREATE ClientAngular/.vscode/tasks.json (980 bytes)
CREATE ClientAngular/src/main.ts (256 bytes)
CREATE ClientAngular/src/index.html (312 bytes)
CREATE ClientAngular/src/styles.css (81 bytes)
CREATE ClientAngular/src/app/app.component.html (20239 bytes)
CREATE ClientAngular/src/app/app.component.spec.ts (966 bytes)
CREATE ClientAngular/src/app/app.component.ts (322 bytes)
CREATE ClientAngular/src/app/app.component.css (0 bytes)
CREATE ClientAngular/src/app/app.config.ts (318 bytes)
CREATE ClientAngular/src/app/app.routes.ts (80 bytes)
CREATE ClientAngular/public/favicon.ico (15086 bytes)
✓ Packages installed successfully.
Author identity unknown

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'Axel089@PcMasterRace.(none)')
● PS C:\Users\Axel089\Desktop\Proyectos\angular\django_angular_api>
```

Crearemos un modulo para las tareas

```
PS C:\Users\q-ani\Desktop\Proyectos\Fronts-Django\ClientAngular> ng generate module tasks --routing
CREATE src/app/tasks/tasks-routing.module.ts (258 bytes)
CREATE src/app/tasks/tasks.module.ts (290 bytes)
```

**Paso 2:** Generamos 2 componentes para tareas uno llamado tasks-list y task-form

```
● PS C:\Users\q-ani\Desktop\Proyectos\Fronts-Django\ClientAngular> ng generate component tasks/task-list
CREATE src/app/tasks/task-list/task-list.component.html (25 bytes)
CREATE src/app/tasks/task-list/task-list.component.spec.ts (630 bytes)
CREATE src/app/tasks/task-list/task-list.component.ts (257 bytes)
CREATE src/app/tasks/task-list/task-list.component.css (0 bytes)
● PS C:\Users\q-ani\Desktop\Proyectos\Fronts-Django\ClientAngular> ng generate component tasks/task-form
CREATE src/app/tasks/task-form/task-form.component.html (25 bytes)
CREATE src/app/tasks/task-form/task-form.component.spec.ts (630 bytes)
CREATE src/app/tasks/task-form/task-form.component.ts (257 bytes)
CREATE src/app/tasks/task-form/task-form.component.css (0 bytes)
○ PS C:\Users\q-ani\Desktop\Proyectos\Fronts-Django\ClientAngular>
```

**Paso 3:** genera un servicio para tareas

```
tasks.service.ts X
Fronts-Django > ClientAngular > src > app > services > tasks.service.ts > TasksService > constructor
1 import { Injectable } from '@angular/core';
2 import { HttpClient } from '@angular/common/http';
3 import { Observable } from 'rxjs';
4 import { Task } from '../task.model';
5
6 @Injectable({
7   providedIn: 'root'
8 })
9 export class TasksService {
10   private apiUrl = 'http://127.0.0.1:8000/tasks/api/v1/tasks/'; // API de Django
11
12   constructor(private http: HttpClient) {}
13
14   getTasks(): Observable<Task[]> {
15     return this.http.get<Task[]>(this.apiUrl);
16   }
17
18   createTask(task: Task): Observable<Task> {
19     return this.http.post<Task>(this.apiUrl, task);
20   }
21
22   updateTask(id: number, task: Task): Observable<Task> {
23     return this.http.put<Task>(`${this.apiUrl}${id}/`, task);
24   }
25
26   deleteTask(id: number): Observable<any> {
27     return this.http.delete(`${this.apiUrl}${id}/`);
28   }
29 }
30
```

Definimos la interfaz para la tarea

```
task.model.ts X
Fronts-Django > ClientAngular > src > app > task.model
1 export interface Task {
2   id: number
3   titulo: string;
4   descripcion: string;
5 }
```

## Configuramos el enrutamiento

```
task.model.ts U tasks.module.ts U task.service.ts U tasks-routing.module.ts U X
Fronts-Django > ClientAngular > src > app > tasks > tasks-routing.module.ts > TasksRoutingModule
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3 import { TaskListComponent } from '../task-list/task-list.component';
4 import { TaskFormComponent } from '../task-form/task-form.component';
5
6 const routes: Routes = [
7   { path: '', component: TaskListComponent }, // Ruta raíz del módulo tasks
8   { path: 'new', component: TaskFormComponent }, // Para crear una nueva tarea
9   { path: 'edit/:id', component: TaskFormComponent }, // Para editar una tarea existente
10 ];
11
12 @NgModule({
13   imports: [RouterModule.forChild(routes)],
14   exports: [RouterModule]
15 })
16 export class TasksRoutingModule {}
```

```
task-list.component.html U styles.css M task-form.component.css M task-list.component.ts U app.routes.ts M X
Fronts-Django > ClientAngular > src > app > app.routes.ts > ...
1
2 import { Routes } from '@angular/router';
3
4 export const routes: Routes = [
5
6   { path: 'tasks', loadChildren: () => import('../tasks/tasks.module').then(m => m.TasksModule) },
7   { path: '', redirectTo: '/tasks', pathMatch: 'full' }, // Redirige la raíz a /tasks
8
9 ];
10
```

## Implementación de task-list

```
task.model.ts U tasks.module.ts U task.service.ts U tasks-routing.module.ts U app.routes.ts M task-list.component.ts U X
Fronts-Django > ClientAngular > src > app > tasks > task-list > task-list.component.ts > TaskListComponent > deleteTask
1 import { Component, OnInit } from '@angular/core';
2 import { Task } from '../task.model';
3 import { TaskService } from '../task.service';
4 import { Router } from '@angular/router'; // Para navegar
5
6 @Component({
7   selector: 'app-task-list',
8   templateUrl: './task-list.component.html',
9   styleUrls: ['./task-list.component.css']
10 })
11 export class TaskListComponent implements OnInit {
12   tasks: Task[] = [];
13
14   constructor(private taskService: TaskService, private router: Router) { }
15
16   ngOnInit(): void {
17     this.loadTasks();
18   }
19
20   loadTasks(): void {
21     this.taskService.getTasks().subscribe(
22       (data) => this.tasks = data,
23       (error) => console.error('Error al cargar tareas', error)
24     );
25   }
26
27   deleteTask(id: number | undefined): void {
28     if (id === undefined) return; // Seguridad
29     if (confirm('Estas seguro de que quieres eliminar esta tarea?')) {
30       this.taskService.deleteTask(id).subscribe(
31         () => {
32           this.tasks = this.tasks.filter(task => task.id !== id); // Actualiza la lista localmente
33         },
34         (error) => console.error('Error eliminado tarea', error)
35       );
36     }
37   }
38
39   editTask(id: number | undefined): void {
40     if (id === undefined) return;
41     this.router.navigate(['/tasks/edit', id]);
42   }
43
44   createTask(): void {
45     this.router.navigate(['/tasks/new']);
46   }
47 }
```

```
task-list.component.html U X
Fronts-Django > ClientAngular > src > app > tasks > task-list > task-list.component.html > div.task-list-container
1  <!-- LISTA DE TAREAS -->
2  <div class="task-list-container">
3    <div class="task-list-header">
4      <h2>Mis Tareas</h2> <!-- Cambiado para diferenciar del titulo del form -->
5      <button class="button button-primary" (click)="createTask()">
6        <i class="fas fa-layer-group"></i> Nueva Tarea <!-- Icono diferente -->
7      </button>
8    </div>
9
10   <!-- Si no hay tareas, el ng-template se mostrará directamente.
11       Si hay tareas, se renderizará el ul. -->
12   <ng-container *ngIf="tasks.length > 0; else noTasksBlock">
13     <ul class="task-list">
14       <li class="task-item" *ngFor="let task of tasks">
15         <h3>{{ task.titulo }}</h3>
16         <p>{{ task.descripcion || 'Esta tarea no tiene descripción.' }}</p>
17         <div class="task-item-actions">
18           <button class="button-edit" (click)="editTask(task.id)" title="Editar Tarea">
19             <i class="fas fa-pencil-alt"></i> <span class="action-text">Editar</span>
20           </button>
21           <button class="button-delete" (click)="deleteTask(task.id)" title="Eliminar Tarea">
22             <i class="fas fa-trash-alt"></i> <span class="action-text">Eliminar</span>
23           </button>
24         </div>
25       </li>
26     </ul>
27   </ng-container>
28
29   <ng-template #noTasksBlock>
30     <div class="no-tasks-message">
31       <i class="fas fa-clipboard-list"></i>
32       Aún no tienes tareas pendientes. <br>¡Es un buen momento para añadir una!
33     </div>
34   </ng-template>
35 </div>
```

## Implementación de task-form

task-form.component.ts M X

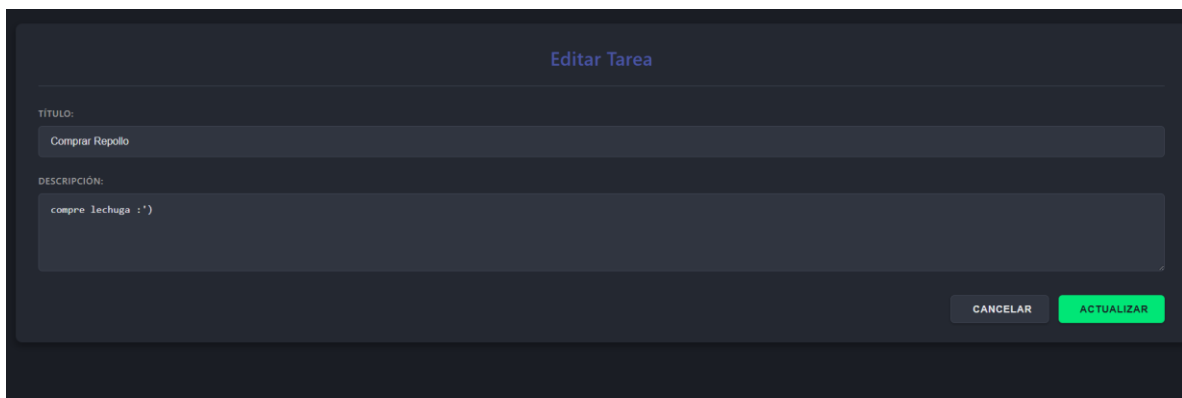
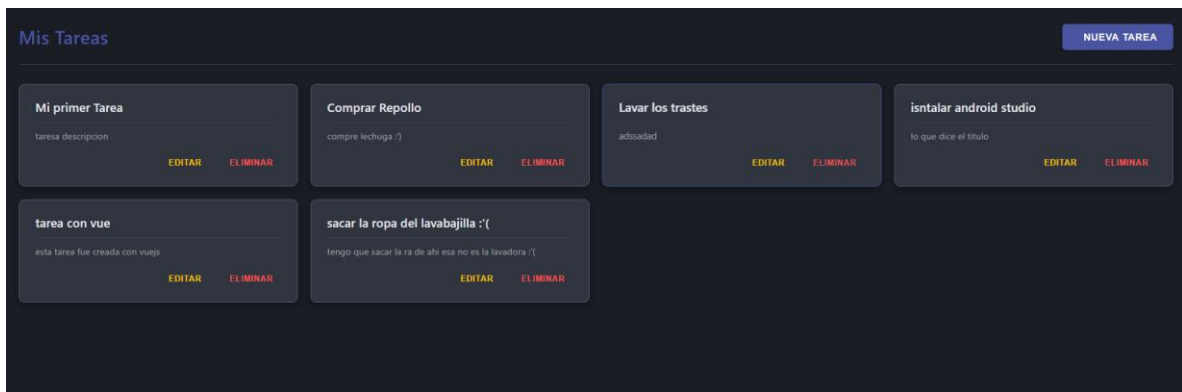
Fronts-Django > ClientAngular > src > app > tasks > task-form > task-form.component.ts TaskFormComponent

```
1 import { Component, OnInit } from '@angular/core';
2 import { FormBuilder, FormGroup, ReactiveFormsModule, Validators } from '@angular/forms';
3 import { ActivatedRoute, Router } from '@angular/router';
4 import { TaskService } from '../task.service';
5 import { Task } from '../task.model';
6 import { CommonModule } from '@angular/common';
7
8 @Component({
9   selector: 'app-task-form',
10   templateUrl: './task-form.component.html',
11   styleUrls: ['./task-form.component.css'],
12   standalone: true,
13   imports: [
14     CommonModule,
15     ReactiveFormsModule,
16   ]
17 })
18 export class TaskFormComponent implements OnInit {
19   taskForm: FormGroup;
20   isEditMode = false;
21   taskId: number | null = null;
22
23   constructor(
24     private fb: FormBuilder,
25     private taskService: TaskService,
26     private router: Router,
27     private route: ActivatedRoute
28   ) {
29     this.taskForm = this.fb.group({
30       titulo: ['', Validators.required],
31       description: ['']
32     });
33   }
34
35   ngOnInit(): void {
36     this.route.paramMap.subscribe(params => {
37       const id = params.get('id');
38       if (id) {
39         this.isEditMode = true;
40         this.taskId = +id;
41         this.taskService.getTask(this.taskId).subscribe(task => {
42           this.taskForm.patchValue(task);
43         });
44       }
45     });
46   }
47
48   onSubmit(): void {
49     if (this.taskForm.invalid) {
50       return;
51     }
52
53     const taskData: Task = this.taskForm.value;
54
55     if (this.isEditMode && this.taskId !== null) {
56       this.taskService.updateTask(this.taskId, taskData).subscribe(
57         () => this.router.navigate(['/tasks']),
58         (error) => console.error('Error actualizando tarea', error)
59       );
60     } else {
61       this.taskService.createTask(taskData).subscribe(
62         () => this.router.navigate(['/tasks']),
63         (error) => console.error('Error creando tarea', error)
64       );
65     }
66   }
67
68   cancel(): void {
69     this.router.navigate(['/tasks']);
70   }
71 }
```



```
task-form.component.ts M task-form.component.html M X
Fronts-Django > ClientAngular > src > app > tasks > task-form > task-form.component.html > div.task-form-container
1  e! FORMULARIO (sin cambios en estructura, pero se beneficia de las clases de error con icono) ->
2  <div class="task-form-container">
3    <h2>{{ isEditMode ? 'Editar Tarea' : 'Crear Nueva Tarea' }}</h2>
4    <form [formGroup]="taskForm" (ngSubmit)="onSubmit()">
5      <div class="form-group">
6        <label for="titulo">Titulo:</label>
7        <input id="titulo" type="text" formControlName="titulo" placeholder="Ej: Diseño de la nueva App">
8        <div *ngIf="taskForm.get('titulo')?.invalid && taskForm.get('titulo')?.touched" class="form-field-error">
9          <i class="fas fa-exclamation-circle"></i> Título es Requerido.
10        </div>
11      </div>
12
13      <div class="form-group">
14        <label for="descripcion">Descripción:</label>
15        <textarea id="descripcion" formControlName="descripcion" placeholder="Detallar los requerimientos, mockups, etc..."></textarea>
16      </div>
17
18      <div class="form-button-group">
19        <button type="button" class="button button-secondary" (click)="cancel()">
20          <i class="fas fa-times"></i> Cancelar
21        </button>
22        <button type="submit" class="button button-success" [disabled]="taskForm.invalid">
23          <i class="fas {{ isEditMode ? 'fa-save' : 'fa-plus-circle' }}"></i>
24          {{ isEditMode ? 'Actualizar' : 'Crear' }}
25        </button>
26      </div>
27    </form>
28  </div>
```

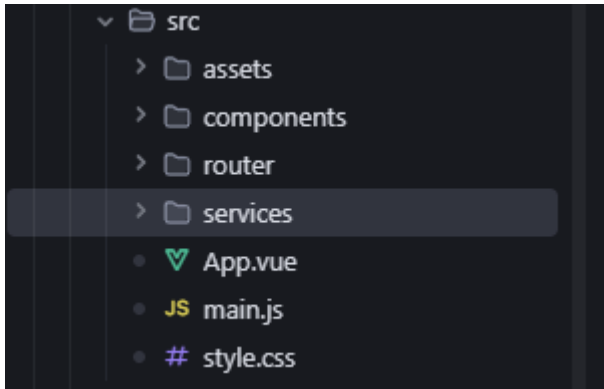
## Y agregamos estilos



## VueJs

**Paso 1:** Instalación de VueJs

**Paso 2:** Creamos una carpeta llamada (components,router,services) en la ruta src



**Paso 3:** Definimos las Apis para consumir con el sistema

En la carpeta que acabamos services que acabamos de crear añadimos un archivo llamado **TaskApi.js** donde se centraran todas las Apis para la aplicación de tareas

```
1  import axios from 'axios';
2
3  const TasksApi = axios.create({ // Crear una instancia de Axios
4    baseURL: 'http://localhost:8000/tasks/api/v1/tasks/', // URL base de la API
5    timeout: 5000, // Tiempo máximo de espera
6  });
7
8  // Obtener todas las tareas
9  export const getAllTasks = () => TasksApi.get('/');
10
11 // Crear nueva tarea
12 export const createTask = (task) => TasksApi.post('/', task);
13
14 // Obtener un tarea por ID
15 export const getTaskById = (id) => TasksApi.get(`/${id}/`);
16
17 // Actualizar un tarea
18 export const updateTask = (id, task) => TasksApi.put(`/${id}/`, task);
19
20 // Eliminar un tarea
21 export const deleteTask = (id) => TasksApi.delete(`/${id}/`);
```

**Paso 4:** Dentro de la carpeta components creamos un archivo llamado ListTasks.vue

que va a ser nuestro componente principal donde se visualizaran todas las tareas

dividimos el componente en una estructura básica

```

1  <template>
2    <div class="task-manager">
3      <h1>Lista de Tareas</h1>
4    </div>
5  </template>
6
7  <script setup>
8    // Lógica del componente irá aquí
9  </script>
10
11 <style scoped>
12 /* Estilos irán aquí */
13 </style>

```

Paso 5: Añadimos el estado dinámico de los datos y su carga inicial

```

7  <script setup>
8  import { ref, onMounted } from 'vue';
9  import { getAllTasks } from '../services/TasksApi';
10
11  const tareas = ref([]);
12  const loading = ref(true);
13
14  onMounted(async () => {
15    try {
16      const response = await getAllTasks();
17      tareas.value = response.data;
18    } catch (error) {
19      console.error('Error al obtener tareas:', error);
20      alert('Error al cargar las tareas');
21    } finally {
22      loading.value = false;
23    }
24  });
25 </script>

```

Ahora podemos empezar a modificar el apartado de listado de tareas en una estructura básica de divs donde colocamos una condicional según el estado en el que se encuentre si esta cargando o si ya se cargaron las tareas

```
<template>
  <div class="task-manager">
    <h1>Lista de Tareas</h1>

    <div v-if="loading" class="loading">Cargando tareas...</div>

    <div v-else>
      <ul v-if="tareas.length" class="task-list">
        <li v-for="tarea in tareas" :key="tarea.id" class="task-item">
          <div class="task-content">
            <h3>{{ tarea.titulo }}</h3>
            <p>{{ tarea.descripcion }}</p>
          </div>
        </li>
      </ul>
      <p v-else class="no-tasks">No hay tareas disponibles</p>
    </div>
  </div>
</template>
```

Paso 6: agregamos los botones de acción

```
<div v-else>
  <ul v-if="tareas.length" class="task-list">
    <li v-for="tarea in tareas" :key="tarea.id" class="task-item">
      <div class="task-content">
        <h3>{{ tarea.titulo }}</h3>
        <p>{{ tarea.descripcion }}</p>
      </div>
      <div class="task-actions">
        <button @click="editTask(tarea)" class="btn btn-edit">Editar</button>
        <button @click="confirmDelete(tarea.id)" class="btn btn-delete">Eliminar</button>
      </div>
    </li>
  </ul>
  <p v-else class="no-tasks">No hay tareas disponibles</p>
</div>
```

Ahora agregamos la lógica para editar y eliminar tareas

```
// Abrir formulario para editar
const editTask = (task) => {
  formData.value = { ...task };
  isEditing.value = true;
  showForm.value = true;
};
```

```
// Confirmar eliminación
const confirmDelete = (id) => {
  taskToDelete.value = id;
  showConfirmModal.value = true;
};
```

Paso 7: ahora creamos un formulario para la creación y edición de tareas

```
<!-- Modal para crear/editar -->
<div v-if="showForm" class="modal-overlay">
  <div class="modal-content">
    <h2>{{ isEditing ? 'Editar Tarea' : 'Crear Tarea' }}</h2>
    <form @submit.prevent="submitForm" class="task-form">
      <div class="form-group">
        <label for="title">Título:</label>
        <input
          id="title"
          v-model="formData.titulo"
          placeholder="Ingrese el título"
          required
          class="form-input"
        >
      </div>
      <div class="form-group">
        <label for="description">Descripción:</label>
        <textarea
          id="description"
          v-model="formData.descripcion"
          placeholder="Ingrese la descripción"
          required
          class="form-textarea"
        ></textarea>
      </div>
      <div class="form-actions">
        <button type="submit" class="btn btn-submit" :disabled="submitting">
          {{ isEditing ? 'Actualizar' : 'Crear' }}
        </button>
        <button type="button" @click="closeForm" class="btn btn-cancel">Cancelar</button>
      </div>
    </form>
  </div>
</div>
```

Y agregamos la lógica para el formulario

Agregamos al estado reactivo-dinámico

```
const showForm = ref(false);  
const isEditing = ref(false);  
const submitting = ref(false);
```

Lógica para abrir el formulario en modo creación

```
// Abrir formulario para crear  
const openCreateForm = () => {  
  formData.value = { titulo: '', descripcion: '' };  
  isEditing.value = false;  
  showForm.value = true;  
};
```

Lógica para abrir el formulario en modo edición

```
// Abrir formulario para editar  
const editTask = (task) => {  
  formData.value = { ...task };  
  isEditing.value = true;  
  showForm.value = true;  
};
```

Lógica para refrescar las tareas

```
// Refrescar lista de tareas  
const refreshTasks = async () => {  
  try {  
    const response = await getAllTasks();  
    tareas.value = response.data;  
  } catch (error) {  
    console.error('Error al refrescar tareas:', error);  
  }  
};
```

Lógica para envío del formulario crear o actualizar

```
// Enviar formulario (crear o actualizar)
const submitForm = async () => {
  submitting.value = true;
  try {
    if (isEditing.value) {
      await apiUpdateTask(formData.value.id, formData.value);
    } else {
      await apiCreateTask(formData.value);
    }
    await refreshTasks();
    closeForm();
  } catch (error) {
    console.error('Error al guardar tarea:', error);
    alert('Error al guardar la tarea');
  } finally {
    submitting.value = false;
  }
};
```

### Paso 8: Ahora implementamos la lógica para la eliminación de las tareas

Agregamos en el template el modal de confirmación que se mostrara cuando presionemos el botón de eliminar

```
<!-- Modal de confirmación de eliminacion -->
<div v-if="showConfirmModal" class="modal-overlay">
  <div class="modal-content confirm-modal">
    <p>¿Estás seguro de que deseas eliminar esta tarea?</p>
    <div class="confirm-actions">
      <button @click="deleteTask" class="btn btn-confirm">Sí, eliminar</button>
      <button @click="cancelDelete" class="btn btn-cancel">Cancelar</button>
    </div>
  </div>
</div>
```

Ahora agregamos al estado reactivo

```
const showConfirmModal = ref(false);
const taskToDelete = ref(null);
```

Y agregamos la lógica para la eliminación de las tareas

```
// Eliminar tarea
const deleteTask = async () => {
  try {
    await apiDeleteTask(taskToDelete.value);
    await refreshTasks();
    showConfirmModal.value = false;
  } catch (error) {
    console.error('Error al eliminar tarea:', error);
    alert('Error al eliminar la tarea');
  }
};
```

Y también agregamos la lógica si ya no se quiere eliminar la tarea

```
// Cancelar eliminación
const cancelDelete = () => {
  showConfirmModal.value = false;
  taskToDelete.value = null;
};
```

#### Paso 9: Agregar estilos (opcional)

#### Paso 10: Dentro de la carpeta router creamos un archivo index.js

donde importamos (CreateRouter, y CreateWebHistory) y creamos las rutas para nuestra aplicación y creamos las rutas que estarán disponibles en el sistema

```
1 import { createRouter, createWebHistory } from 'vue-router';
2 import ListTasks from '../components/ListTasks.vue';
3
4 const routes = [
5   {
6     path: '/tasks',
7     name: 'Tasks',
8     component: ListTasks,
9   },
10  // Redirección para manejar la ruta base
11  {
12    path: '/',
13    redirect: '/tasks'
14  }
15 ];
16
17 const router = createRouter({
18   history: createWebHistory(),
19   routes,
20 });
21
22 export default router;
```



Agregamos una barra de navegación para acceder al listado de tareas en el archivo app.vue

```
1  <template>
2    <div>
3      <nav>
4        <router-link to="/tasks">Ver tareas</router-link>
5      </nav>
6      <router-view></router-view> <!-- Aquí se renderiza el componente -->
7    </div>
8  </template>
9
10 <script setup>
11 // No necesitas lógica adicional aquí
12 </script>
13
14 <style>
15 /* Estilos opcionales */
16 nav {
17   margin-bottom: 20px;
18 }
19 router-link {
20   margin-right: 10px;
21 }
22 </style>
```

Y lo agregamos al archivo main.js que se encuentra en la raíz del de la carpeta src

```
1  import { createApp } from 'vue';
2  import App from './App.vue';
3  import router from './router'; // Importa el router
4
5  const app = createApp(App);
6
7  app.use(router); // Usa el router
8  app.mount('#app');
```

Y vemos cómo funciona

[Ver tareas](#)

## Lista de Tareas

[Crear Nueva Tarea](#)

### Mi primer Tarea

tarea descripcion

Editar

Eliminar

### Comprar Repollo

compre lechuga :)

Editar

Eliminar

### Lavar los trastes

adssadad

Editar

Eliminar

### isntalar android studio

lo que dice el titulo

Editar

Eliminar

### tarea con vue

esta tarea fue creada con vuejs

Editar

Eliminar

## Lista de Tareas

[Crear Nueva Tarea](#)

### Mi primer Tarea

tarea descripcion

Editar

Eliminar

### Comprar Repollo

compre lechuga :

Editar

Eliminar

### Lavar los trastes

adssadad

Editar

Eliminar

### isntalar android

lo que dice el titulo

Editar

Eliminar

### tarea con vue

esta tarea fue creada con vuejs

Editar

Eliminar

### Crear Tarea

Título:

Ingrese el título

Descripción:

Ingrese la descripción

Crear

Cancelar

# Lista de Tareas

Crear Nueva Tarea

<b>Mi primer Tarea</b>	Editar	Eliminar
tarea descripción		
<b>Comprar Repuestos</b>	Editar	Eliminar
compre lechuga :)		
<b>Lavar los traseros</b>	Editar	Eliminar
adssadad		
<b>isntalar androide</b>	Editar	Eliminar
lo que dice el titulo		
<b>tarea con vue</b>	Editar	Eliminar
esta tarea fue creada con vuejs		

## Editar Tarea

Titulo:

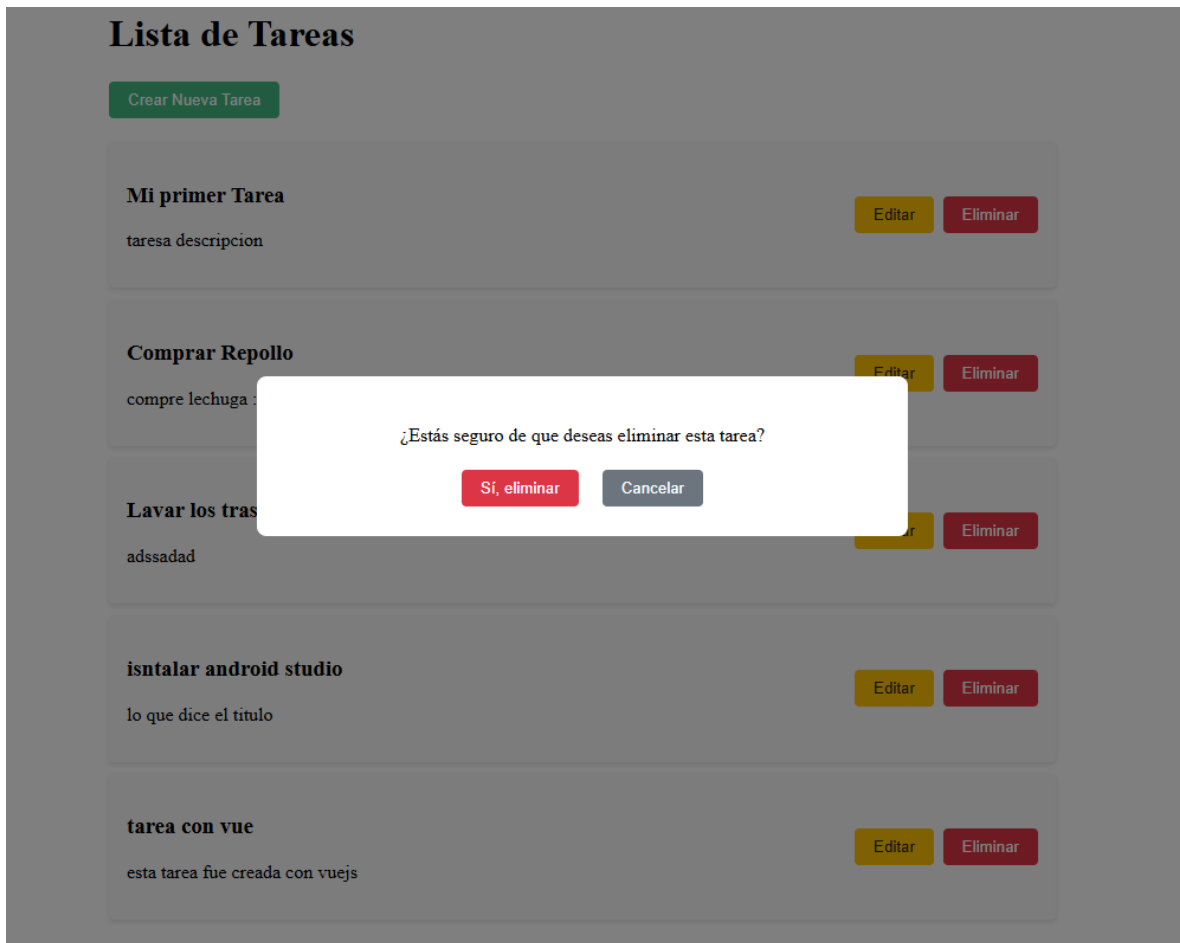
tarea con vue

Descripción:

esta tarea fue creada con vuejs

Actualizar

Cancelar



## Usabilidad y Mantenibilidad en Cada Framework

### Resultados

El presente estudio se fundamenta en un análisis comparativo de los frameworks React, Angular y Vue.js, aplicando un enfoque basado en métricas de calidad del software según las normas ISO 9241-11:2018 (usabilidad) e ISO/IEC 25010:2011 (mantenibilidad). El análisis se apoya en métricas reconocidas y datos provenientes de investigaciones recientes, encuestas de uso, benchmarks y documentación técnica, con el fin de orientar la selección del framework más adecuado para su integración con Django Rest Framework (DRF).

### Usabilidad

- **Cantidad de código necesario para implementar una funcionalidad (líneas de código)**

Según Moraguez, E. R. (2025). En términos generales, Vue.js permite implementar funcionalidades con menos líneas de código en comparación con React y Angular, gracias a su sintaxis sencilla y su enfoque progresivo, que favorece un código más conciso y estructurado. Su diseño flexible y el uso de

directivas facilitan la creación de funcionalidades sin necesidad de escribir tanto código como en otros frameworks. React, aunque eficiente, suele requerir más líneas al integrarse con bibliotecas externas para la gestión de estados y efectos. Por su parte, Angular, al estar basado en TypeScript y contar con una estructura más completa, tiende a demandar una mayor cantidad de código para definir componentes y servicios.

- **Velocidad de aprendizaje inicial**

Según Moraguez, E. R. (2025). **Angular, React y Vue** son frameworks populares con enfoques distintos. Angular presenta una curva de aprendizaje más pronunciada debido a su estructura compleja y al uso de TypeScript, aunque es robusto y cuenta con numerosas herramientas integradas. Por su parte, React ofrece un enfoque minimalista basado en componentes, lo que lo hace más accesible al principio; sin embargo, en proyectos avanzados, puede requerir la incorporación de múltiples paquetes externos, aumentando así la complejidad. **Vue**, en cambio, se destaca por su simplicidad y claridad, lo que facilita su adopción tanto para principiantes como para desarrolladores con experiencia. Su naturaleza progresiva permite aprenderlo rápidamente y adaptarlo según las necesidades del proyecto.

- **Uso de recursos del sistema durante la carga y renderizado**

Según los datos presentados en el benchmark de frameworks front-end por (*Interactive Results*, s. f.), se evalúa el rendimiento de diversas tecnologías en tareas fundamentales como la creación, actualización y eliminación de elementos en el DOM. La medición se realiza con métricas de duración en milisegundos y se establece un intervalo de confianza del 95%, lo que proporciona una comparación objetiva entre los frameworks. Los resultados preliminares indican que Vue.js y React tienden a mostrar tiempos de ejecución más rápidos en ciertas pruebas, mientras que Angular, aunque ofrece una estructura más robusta, presenta una ligera penalización en velocidad. Cabe destacar que los métodos de medición han cambiado en versiones recientes del benchmark, lo que puede influir en los valores obtenidos. En general, la página advierte que los resultados pueden variar dependiendo del navegador y el hardware utilizado, por lo que se recomienda interpretarlos con precaución.

- **Satisfacción del usuario**

Los datos de satisfacción de los frameworks React, Angular y Vue.js reflejan la percepción de los desarrolladores sobre su experiencia de uso y eficiencia en proyectos front-end. Según la encuesta de **State of JavaScript 2024**, React tiene una satisfacción del 82.95%, Vue.js alcanza el 77.32%, mientras que Angular presenta una satisfacción significativamente menor con 42.62%.

- **Tasa de preferencia frente a otros Frameworks**

Los datos de popularidad de los frameworks React, Angular y Vue.js reflejan tendencias clave en el ecosistema de desarrollo front-end, evidenciando la evolución en la adopción de tecnologías por parte de los desarrolladores. Según la encuesta realizada por Stack Overflow en 2024, React se posiciona como el framework más utilizado, alcanzando una popularidad del 39.5%, seguido por Angular con 17.1% y Vue.js con 15.4% (Tkrotoff, 2024).

## Mantenibilidad

- **Documentación del framework**

Según pauli pali (Ocrubre,2020). La tabla compara Angular, React y Vue.js en cuanto a su soporte para diversas características arquitectónicas que impactan la mantenibilidad del software. Angular destaca por ofrecer soporte nativo para el tipado estático y los espacios de nombres (namespaces), además de facilitar la minimización de dependencias; su soporte para la inmutabilidad es parcial y la automatización de pruebas se consigue mediante librerías externas. React, por su parte, soporta la inmutabilidad de forma robusta y permite el tipado estático a través de librerías externas, pero carece de soporte inherente para espacios de nombres y, según la tabla, para la minimización de dependencias; la automatización de pruebas también depende de herramientas externas. Finalmente, Vue.js ofrece tipado estático mediante addons opcionales y presenta un soporte parcial para espacios de nombres, inmutabilidad y minimización de dependencias, mientras que la automatización de pruebas se logra parcialmente a través de librerías externas.

**Table 4.** *Summary of maintainability improvement methods that each framework supports.*

Architecture	Angular	React	Vue.js
Static typing	Yes	Yes, through external library	Yes, through optional addon
Namespaces	Yes	No	Partly
Immutability	Partly	Yes	Partly
Minimizing dependencies	Yes	No	Partly
Test automation	Yes, through external library	Yes, through external library	Partly, through external library

tupeliPauli 2024

## Arquitectura de framework

Según tupeli Pauli (Ocrubre,2020). presenta un resumen de los estilos arquitectónicos soportados por los frameworks Angular, React y Vue.js. Se observa que los tres frameworks (Angular, React y Vue.js) soportan la Arquitectura Basada en Componentes (CBA). En cuanto a la Arquitectura Dirigida por Eventos (EDA), Angular no la soporta directamente, mientras que React puede implementarla mediante librerías externas y Vue.js a través de addons opcionales. La Arquitectura Orientada a Servicios (SOA) es soportada por Angular, pero no por React ni Vue.js. Finalmente, la arquitectura AOA (cuyas siglas podrían referirse a Arquitectura Orientada a Aspectos es soportada por Angular y Vue.js, y en React se puede implementar como un patrón de diseño.

**Table 3.** Summary of architectural styles that each framework support.

Architecture	Angular	React	Vue.js
CBA	Yes	Yes	Yes
EDA	No	Yes, through external library	Yes, through optional addon
SOA	Yes	No	No
AOA	Yes	Yes, as a pattern	Yes

tupeliPauli 2024

**Ventajas y desventajas por framework**

Los hallazgos extraídos reflejan que cada framework presenta fortalezas particulares en función del contexto de uso, experiencia del equipo y requerimientos del proyecto:

React destaca por su flexibilidad, amplio ecosistema y rendimiento optimizado a través del Virtual DOM. Es ideal para aplicaciones altamente interactivas, modulares o que requieren integración con múltiples bibliotecas externas. Sin embargo, su falta de estructura predeterminada puede representar un reto para mantener la consistencia en equipos grandes.

Angular ofrece una arquitectura completa e integrada, lo cual resulta beneficioso en proyectos empresariales de gran escala. Su estructura opinada, soporte para pruebas, formularios y enrutamiento nativos brindan mantenibilidad, y escalabilidad. No obstante, su curva de aprendizaje es más pronunciada, y la complejidad del código inicial suele ser mayor debido al uso de TypeScript y su sistema de inyección de dependencias.

Vue.js se posiciona como una herramienta de rápida adopción, gracias a su sintaxis clara, documentación accesible y enfoque progresivo. Permite construir aplicaciones robustas sin requerir una gran carga conceptual. Aunque presenta un rendimiento competitivo, su comunidad es más pequeña en comparación con React y Angular, y su presencia en entornos corporativos es más limitada.

**Métricas relevantes observadas**

Las métricas extraídas de fuentes como State of JavaScript (2024), Tkrotoff (2025) y el benchmark de Krausest (s.f.) reflejan los siguientes patrones:

**Usabilidad:**

- Vue.js requiere menos líneas de código para implementar funcionalidades básicas.
- React ofrece un equilibrio entre simplicidad y control, pero demanda configuración adicional para tareas comunes.
- Angular es más verboso, pero provee herramientas integradas que mejoran la robustez y la coherencia del desarrollo.

### **Velocidad de aprendizaje:**

- Vue.js es el más accesible para nuevos desarrolladores.
- React se posiciona en un punto intermedio, aunque requiere tiempo para dominar su ecosistema.
- Angular presenta mayor complejidad técnica inicial.

### **Rendimiento:**

- En operaciones DOM, Vue y React presentan mejores tiempos de renderizado.
- Angular es ligeramente más lento, aunque más estable en contextos complejos.

### **Satisfacción:**

- React (82.95%) y Vue.js (77.32%) mantienen altos niveles de satisfacción.
- Angular registra menor preferencia (42.62%).

Estas observaciones no pretenden establecer un “**mejor**” framework universal, sino proporcionar una base para seleccionar el más adecuado según necesidades específicas de mantenibilidad, usabilidad y contexto técnico.

Identificación de ventajas y desventajas de cada framework en términos de usabilidad y mantenibilidad.

### **React: Proyectos Modulares, Interactivos y Escalables**

React es ideal cuando necesitas flexibilidad, interactividad y una alta escalabilidad con libertad en la arquitectura.

#### **Dashboards interactivos con actualización en tiempo real**

- **Ejemplo:** Panel de control para analítica, ventas, métricas de usuarios (tipo Google Analytics).
- **Justificación:** Su Virtual DOM y sistema de estados lo hacen perfecto para interfaces reactivas.

#### **Sistemas de diseño y bibliotecas de componentes reutilizables**

- **Ejemplo:** Creación de un Design System personalizado para una empresa.
- **Justificación:** Arquitectura basada en componentes y facilidad de composición.

#### **Aplicaciones SPA altamente dinámicas**

- **Ejemplo:** Plataforma de reservas, editores visuales, redes sociales.
- **Justificación:** React permite dividir la UI en unidades pequeñas y actualizar partes específicas.

#### **Aplicaciones móviles + web**



- **Ejemplo:** App de e-commerce hecha con React Native (mobile) + React (web).
- **Justificación:** Reutilización de lógica y componentes entre plataformas.

#### **Proyectos que necesitan integrarse con otras tecnologías**

- **Ejemplo:** Microfrontend, aplicaciones híbridas con otras bibliotecas (Chart.js, Three.js).
- **Justificación:** React es fácil de integrar gracias a su naturaleza no-opinionada.

### **Angular: Proyectos Empresariales, Escalables y de Gran Equipo**

Angular es ideal para empresas que requieren estructura rígida, herramientas integradas y mantenibilidad a largo plazo.

#### **Sistemas ERP / CRM complejos**

- **Ejemplo:** Software para manejo de clientes, facturación, inventario.
- **Justificación:** Angular incluye todo (enrutamiento, formularios, inyección de dependencias) y escala bien con grandes equipos.

#### **Aplicaciones gubernamentales o bancarias**

- **Ejemplo:** Portal ciudadano, gestión de licencias, banca en línea.
- **Justificación:** Necesitan seguridad, pruebas, validaciones, control de errores y estabilidad.

#### **Aplicaciones multipágina (MPA) robustas**

- **Ejemplo:** Intranet empresarial, plataformas administrativas modulares.
- **Justificación:** Angular puede manejar rutas complejas y lazy loading eficientemente.

#### **Sistemas internos con ciclos de vida largos**

- **Ejemplo:** Aplicaciones internas para control de producción, RRHH o logística.
- **Justificación:** Angular facilita actualizaciones, pruebas y mantenibilidad por años.

#### **Proyectos con requerimientos legales de calidad de software**

- **Ejemplo:** Proyectos certificados o auditables.
- **Justificación:** Angular es compatible con normas ISO por su estructura modular y capacidad de pruebas integradas.

### **Vue.js: Proyectos Ligeros, Rápidos y de Entrega Ágil**

Vue.js es ideal para startups, PYMEs y desarrolladores que requieren velocidad de desarrollo, curva de aprendizaje baja y buen rendimiento inicial.

### Sitios web interactivos o landing pages avanzadas

- **Ejemplo:** Página de producto, micrositios con animaciones o formularios.
- **Justificación:** Vue permite integración progresiva y optimización rápida.

### MVPs (productos mínimos viables)

- **Ejemplo:** Prototipo de plataforma de cursos o red social pequeña.
- **Justificación:** Se puede escalar gradualmente a medida que crece el producto.

### Sistemas administrativos de pequeña o mediana escala

- **Ejemplo:** Aplicaciones para control de inventarios, agendas o reservas.
- **Justificación:** Vue es rápido de configurar, mantener y enseñar a nuevos desarrolladores.

### Aplicaciones con equipos pequeños o en solitario

- **Ejemplo:** SaaS personal, app freelance para cliente específico.
- **Justificación:** Vue ofrece estructura clara sin requerir muchas decisiones técnicas.

### Integración en aplicaciones existentes

- **Ejemplo:** Añadir una sección interactiva a una app legacy.
- **Justificación:** Vue puede insertarse en pequeños componentes sin alterar el sistema completo.

## Recomendaciones según el tipo de proyecto.

Según **Piastou, Mikita. (2023)** Estos Son los contextos de uso de los Frameworks más usados en la actualidad

### React

- **Tipo de aplicaciones:**
  - Aplicaciones web dinámicas y de alto rendimiento (e.g., dashboards, redes sociales).
  - Aplicaciones de una sola página (SPA).
  - Aplicaciones móviles usando React Native.
  - Aplicaciones que requieren integración con otras bibliotecas o herramientas.
- **Contexto de uso:**
  - Ideal para proyectos con cambios frecuentes en el estado de la UI.
  - Ampliamente adoptado por empresas grandes (e.g., Facebook, Instagram).
  - Uso frecuente en proyectos modulares debido a su arquitectura basada en componentes.
- **Ventajas:**
  - Flexibilidad y facilidad de integración con otras bibliotecas.
  - Fuerte comunidad y ecosistema robusto.

- Virtual DOM para un rendimiento eficiente.

## Angular

- **Tipo de aplicaciones:**
  - Aplicaciones empresariales a gran escala (e.g., sistemas CRM/ERP).
  - Aplicaciones multipágina (MPA) con requerimientos complejos.
  - Aplicaciones que demandan una estructura sólida y herramientas integradas.
- **Contexto de uso:**
  - Proyectos con equipos grandes debido a su estructura estricta y opinada.
  - Uso en aplicaciones que requieren funcionalidad compleja integrada (e.g., formularios avanzados, enrutamiento robusto).
  - Preferido por empresas que buscan soluciones completas "todo en uno".
- **Ventajas:**
  - Proporciona una arquitectura completa (MVC, DI, RxJS).
  - Fuertes herramientas de desarrollo y pruebas integradas.
  - Estándar en entornos corporativos.

## Vue.js

- **Tipo de aplicaciones:**
  - Aplicaciones ligeras y rápidas (e.g., sitios web interactivos).
  - Proyectos con equipos pequeños o medianos.
  - Proyectos que buscan una curva de aprendizaje rápida.
- **Contexto de uso:**
  - Ideal para proyectos con necesidades rápidas de desarrollo.
  - Usado en startups y PYMEs por su flexibilidad y simplicidad.
  - Comúnmente adoptado en aplicaciones que requieren interacción con otras tecnologías o frameworks.
- **Ventajas:**
  - Fácil de aprender y configurar.
  - Tamaño ligero y enfoque progresivo.
  - Excelente documentación y comunidad en crecimiento.

## Conclusiones

## Referencias

Tkrotoff. (2025). *Front-end frameworks popularity (React, Vue, Angular and Svelte)*. GitHub Gist. Recuperado de <https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190>

State of JavaScript. (2022). *Front-end frameworks*. Recuperado el 12 de mayo de 2025, Recuperado de <https://2022.stateofjs.com/en-US/libraries/front-end-frameworks>

Moraguez, E. R. (2025). *Comparativa de Frameworks Front-end: React vs. Vue vs. Angular*. LovTechnology. Recuperado el 27 de mayo de 2025, de <https://lovtechnology.com/comparativa-de-frameworks-front-end-react-vs-vue-vs-angular/>

*Interactive results*. (s. f.). <https://krausest.github.io/js-framework-benchmark/current.html>

Pano, A., Graziotin, D., & Abrahamsson, P. (2018). Factors and actors leading to the adoption of a JavaScript framework. *Empirical Software Engineering*, 23(6), 3503-3534. <https://doi.org/10.1007/s10664-018-9613-x>

Tupeli Pauli (October 2020). ENSURING MAINTAINABILITY OF JAVASCRIPT WEB APPLICATIONS Recuperado de [TupeliPauli.pdf](#)

Sanjay, K. C. (2024). *Comparative Study of Front-end Frameworks : React and Angular*. Theseus. <https://urn.fi/URN:NBN:fi:amk-2024080724089>