Universidad Nacional, Costa Rica

Sede Chorotega,

Campus Nicoya

Curso:

Programación IV

Trabajo:

Investigación Tanstack Form

Profesor:

Lawrence Fowks Peña

Estudiantes:

Andrés Alvarado Quesada
Angélica María Ortiz Barrantes
Greilyn María Esquivel Salazar
Katheryn Méndez Quirós
Krystel Fabiana Salazar Chavarría

Tabla de Contenido

Tanstack Form

Según TanStack Form Docs. (n.d.), el Tanstack Form es una biblioteca open source creada con el propósito de abordar las dificultades relacionadas con el control del estado en formularios web. Nace como una solución a una problemática frecuente: muchos frameworks web carecen de herramientas integrales para gestionar formularios, lo que obliga a los desarrolladores a construir sus propias implementaciones o a usar librerías con funcionalidades limitadas. Soporta esquemas de validación (Zod) el cual se ampliará en los siguientes puntos.

Características.

Según Uche, R. (n.d.), Tanstack posee las siguientes características:

- ***Compatibilidad con TypeScript:*** Garantiza la seguridad y robustez de los tipos.
- ***Componentes de interfaz de usuario sin encabezado:*** Proporciona un control total sobre la representación y el comportamiento de los elementos del formulario.
- ***Diseño independiente del marco:*** Se integra a la perfección con React y otros marcos.

Ventajas.

Según Shirdhankar (2023), Tanstack posee las siguientes ventajas:

- ***Gestión del estado sucio:*** Realiza un seguimiento de los cambios en los campos del formulario y proporciona información instantánea a los usuarios.
- ***Componentes de shadcn:*** Los componentes prediseñados garantizan accesibilidad y un estilo perfecto.
- ***Validación dinámica:*** Zod valida los campos en tiempo real, lo que garantiza un manejo sólido de la entrada de usuario.

Desventajas.

Según TanStack Form Docs. (n.d.), Tanstack posee las siguientes desventajas:

- ***Curva de Aprendizaje para Usuarios Nuevos.*** La sintaxis y enfoque de TanStack Form pueden requerir un tiempo de adaptación. Conceptos como "subscripciones granulares" o el manejo de stores adaptables pueden ser confusos al principio.

- ***Overkill para Proyectos Simples.*** Si el formulario es básico (1-2 campos sin validación compleja), usar TanStack Form puede ser excesivo frente a alternativas más simples como: Manejo manual con useState, librerías ultra-ligeras como Felte (para Svelte) o VeeValidate (Vue).
- ***Dependencia de TanStack/React.*** Aunque se considera un "framework-agnóstico", su diseño se mantiene orientado a React (hooks, etc.). En otros frameworks (ej: Vue con Composition API), la integración suele necesitar de más configuración.
- ***Validación por Defecto Menos Intuitiva:*** A diferencia de React Hook Form (que tiene un sistema de validación incorporado sencillo), TanStack Form delega la validación a librerías externas (Zod, Yup, etc.). Esto agrega un paso adicional de configuración (aunque también ofrece más flexibilidad).

Guía Rápida: Usar TanStack Form, Zod y API Platzi Users

1. ¿Qué es TanStack Form, Zod y la API de Platzi?

- **TanStack Form:** Librería de React para manejar formularios de forma sencilla y eficiente.
- **Zod:** Librería de validaciones que nos permite asegurar que los datos enviados cumplen las reglas necesarias.

Librerias a instalar:

1. Instalar React si no tienes un proyecto creado

```
npm create vite@latest
```

Luego eliges:

- Framework: React
- TypeScript + SWC

Entras a la carpeta:

```
cd nombre-del-proyecto
```

Instalar dependencias:

```
npm install
```

2. Instalar TanStack Form y Zod

```
install @tanstack/react-form zod
```

- **API de Platzi Users:** API que permite crear, consultar, actualizar y eliminar usuarios mediante peticiones HTTP (si se quisiera cambiar la api, se cambia el siguiente link en el código).

```
https://api.escuelajs.co/api/v1/users
```

2. Crear esquema de validación con Zod

Primero creamos las reglas que nuestros datos deben cumplir usando Zod.

Creamos un archivo llamado 'userSchema.ts' donde definimos los campos que validamos:

```
import { z } from 'zod';

export const userSchema = z.object({
  name: z.string().min(3, 'Name must have at least 3 characters'),
  email: z.string().email('Invalid email address'),
  password: z.string().min(6, 'Password must have at least 6 characters'),
});
```

3. Crear el formulario usando TanStack Form

TanStack Form nos permite manejar los formularios en React.

Creamos un archivo 'RegistroForm.tsx' donde implementamos el formulario, conectamos la validación de Zod y configuramos el envío de datos a la API.

```
    import { useForm } from '@tanstack/react-form';

    import { userSchema } from './userSchema';

    import { zodValidator } from '@tanstack/zod-form-adapter';

    const RegistroForm = () => {
    const form = useForm({
       defaultValues: {
          name: '',
          email: '',
          password: '',
       },
    validatorAdapter: zodValidator,
    onSubmit: async ({ value }) => {
          await fetch('https://api.escuelajs.co/api/v1/users', {
          method: 'POST',
          headers: { 'Content-Type': 'application/json' },
          body: JSON.stringify({
                name: value.name,
                email: value.email,
                password: value.password,
          avatar: 'https://api.lorem.space/image/face?w=640&h=480',
          }),
       });
      },
    });

    return (
    <form onSubmit={form.handleSubmit}>
          <input {...form.register('name')} placeholder="Name" />
          <input {...form.register('email')} placeholder="Email" />
          <input {...form.register('password')} type="password"
    placeholder="Password" />
          <button type="submit">Register</button>
     </form>
    );
};

    export default RegistroForm;
```

4. Conexión con la API de Platzi Users

Cuando enviamos el formulario, los datos se mandan mediante un `fetch` a la API de Platzi.

La API espera recibir:

- name
- email
- password
- avatar (es requerido, por eso enviamos uno fijo)

El método que utilizamos es POST, y enviamos los datos en formato JSON.

Paso 4.1. onSubmit se activa

Cuando el usuario hace clic en el botón Register del formulario, se ejecuta la función form.handleSubmit, la cual invoca a la función onSubmit.

Esta función onSubmit recibe un objeto que incluye una propiedad llamada value. Dentro de value se encuentran los datos que el usuario ingresó en los campos del formulario: name, email y password.

Paso 4.2. fetch hace la petición

Dentro de onSubmit, usamos fetch para hacer una petición HTTP POST a la API de Platzi: https://api.escuelajs.co/api/v1/users

De esta manera se le está diciendo a la API que queremos crear un nuevo usuario y que esos son los datos.

Paso 4.3. Configuración del fetch

`method: 'POST'`	Queremos crear un nuevo recurso en la API (no solo leer datos).

`headers: { 'Content-Type': 'application/json' }`	Le decimos a la API que le estamos enviando datos en formato JSON.
`body: JSON.stringify({...})`	Enviamos los datos reales que el usuario llenó.

5. Notas rápidas

- Zod define las reglas de validación de los campos.
- TanStack Form gestiona el estado y eventos del formulario.
- zodValidator conecta TanStack Form con Zod para validar automáticamente.
- fetch realiza la petición POST a la API de Platzi Users.
- No se manejan errores en esta versión para simplificar la implementación.

Referencias

Dreezy. (n.d.). *GitHub - Dreezy305/Vite-TanstackForm-shAdcn-ts-zoD*. GitHub. https://github.com/Dreezy305/Vite-TanstackForm-shadcn-ts-zod

Idris, B. (2025b, February 12). Mastering Form State and Validation with Tanstack and Zod. *Medium*. https://medium.com/@bankoleidris/mastering-form-state-and-validation-with-tanstack-and-zod-0f3cb998c589

TanStack. (n.d.). *form/docs/overview.md at main · TanStack/form*. GitHub. https://github.com/TanStack/form/blob/main/docs/overview.md

TanStack form. (n.d.). https://tanstack.com/form/latest

This Dot Media. (2025, April 16). *What Makes TanStack Form Different from Other Form State Managers?* [Video]. YouTube. https://www.youtube.com/watch?v=TkL2QoccTo0

Shirdhankar, S. (2025, 23 enero). Effortless Form Validation in React: TanStack, Zod, and shadcn Integration. *Medium*. https://medium.com/@siddhesh.shirdhankar18/effortless-form-validation-in-react-tanstack-zod-and-shadcn-integration-159bcf4d7f46

Uche, R. (n.d.). *TanStack Form: all-in-one form handling for REACT*. https://blog.openreplay.com/tanstack-form--all-in-one-form-handling-for-react