

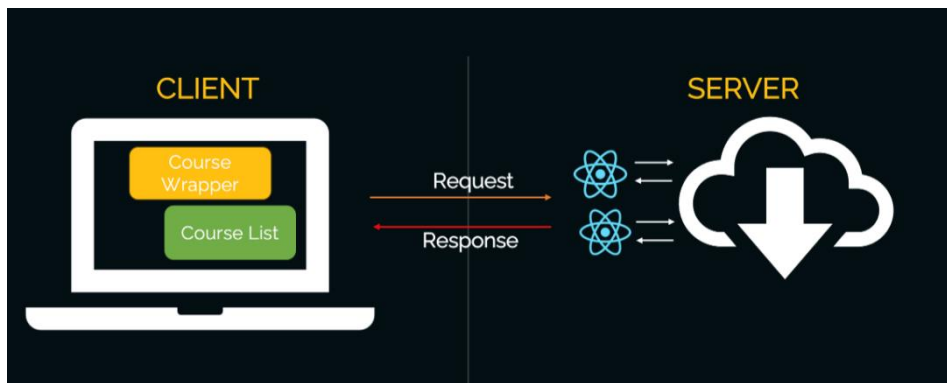
CUADERNO PERSONAL

DESARROLLO DE APLICACIONES WEB

SEMANA 11 – Server Side Props – Next Auth

Server Side Props

Server Side Props es una técnica en Next.js que permite obtener y renderizar datos en el servidor antes de enviar la página al cliente. Esto es útil para aplicaciones que necesitan mostrar datos que cambian con frecuencia o requieren acceso a recursos protegidos.

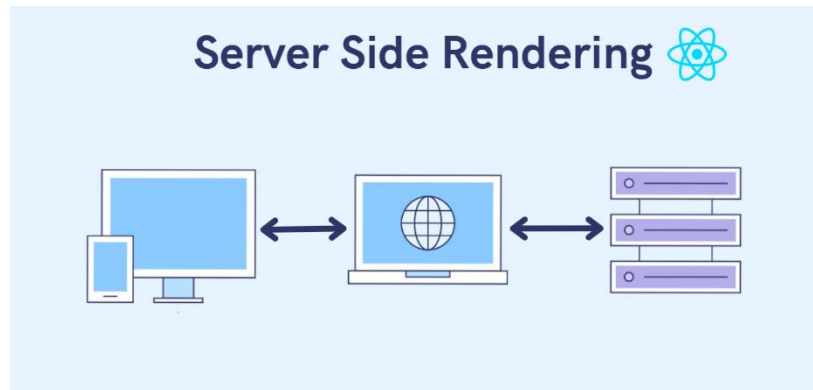


getServerSideProps

El método `getServerSideProps` se utiliza en Next.js para obtener datos en el lado del servidor en cada solicitud. Este método se ejecuta en el servidor, lo que significa que los datos están siempre actualizados y pueden utilizarse para renderizar la página antes de que se envíe al cliente.

- **Características de `getServerSideProps`:**
 - Se ejecuta en el servidor en cada solicitud.
 - Puede acceder a la solicitud y respuesta del cliente.
 - Ideal para páginas que requieren datos siempre actualizados o protegidos.
- **Ventajas:**
 - **Datos actualizados:** Siempre devuelve datos actualizados, lo que es ideal para contenido dinámico.

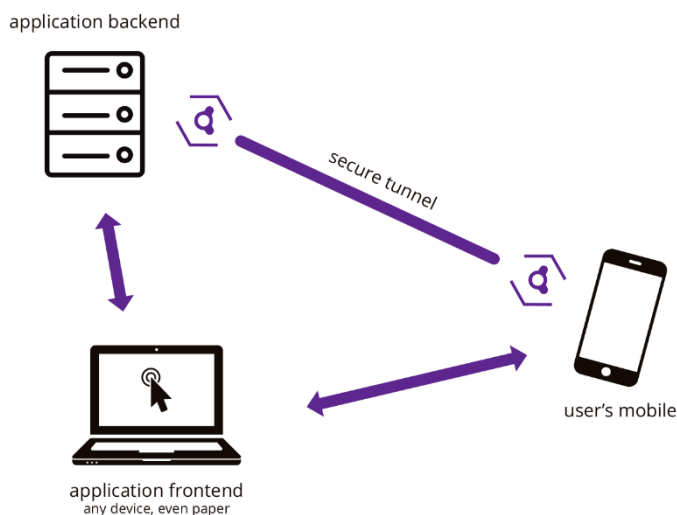
- **Seguridad:** Los datos se obtienen en el servidor, lo que protege la información sensible.
- **Desventajas:**
 - **Rendimiento:** Puede aumentar el tiempo de carga de la página, ya que cada solicitud implica una llamada al servidor.



- **Carga del servidor:** Incrementa la carga del servidor debido a la ejecución en cada solicitud.

NextAuth.js

NextAuth.js es una biblioteca para implementar autenticación en aplicaciones Next.js. Ofrece soporte para múltiples proveedores de autenticación, incluyendo OAuth, y permite personalizar la lógica de autenticación.



Instalación y Configuración

Para instalar NextAuth.js en tu proyecto, sigue estos pasos:

1. Instalar dependencias:

```
bash
```

```
Copy code
```

```
npm install next-auth
```

2. Configurar la API de autenticación:

Crea un archivo [...nextauth].js en la carpeta pages/api/auth para definir la configuración de NextAuth.js.

jsx

Copy code

```
// pages/api/auth/[...nextauth].js
```

```
import NextAuth from 'next-auth';
```

```
import Providers from 'next-auth/providers';
```

```
export default NextAuth({
```

```
  providers: [
```

```
    Providers.Google({
```

```
      clientId: process.env.GOOGLE_CLIENT_ID,
```

```
      clientSecret: process.env.GOOGLE_CLIENT_SECRET,
```

```
    }},
```

```
    // Agrega más proveedores según sea necesario
```

```
  ],
```

```
  database: process.env.DATABASE_URL, // URL de la base de datos
```

```
  secret: process.env.NEXTAUTH_SECRET, // Secreto para la firma de JWT
```

```
});
```

- **Proveedores de autenticación:** NextAuth.js admite varios proveedores, como Google, Facebook, GitHub, etc. Define los proveedores en la configuración.
- **Base de datos:** Especifica una URL de base de datos si deseas almacenar sesiones o datos de usuario.
- **Secreto:** Define un secreto para asegurar la firma de JWT.



Obtener Información de la Sesión

NextAuth.js facilita el acceso a la información de la sesión mediante el hook `useSession`.

Ejemplo de uso de `useSession`:

```
jsx
Copy code
import { useSession, signIn, signOut } from 'next-auth/react';

function HomePage() {
  const { data: session, status } = useSession();

  if (status === 'loading') {
    return <p>Cargando...</p>;
  }

  return (
    <div>
      {!session ? (
        <>
          <p>No estás autenticado</p>
          <button onClick={() => signIn()}>Iniciar sesión</button>
        </>
      ) : (
        <>
          <p>Hola, {session.user.name}</p>
          <button onClick={() => signOut()}>Cerrar sesión</button>
        </>
      )}
    </div>
  );
}

export default HomePage;
```

En este ejemplo, `useSession` se utiliza para verificar si el usuario está autenticado y mostrar el contenido correspondiente. La función `signIn` inicia sesión y `signOut` cierra la sesión.

Proveedores y Credenciales

Además de los proveedores OAuth, NextAuth.js permite configurar autenticación basada en credenciales personalizadas.

Callbacks y Verificación de Credenciales

NextAuth.js permite personalizar la lógica de autenticación mediante callbacks que se ejecutan en diferentes puntos del flujo de autenticación.

Ejemplo de callback:

```
jsx
```



Copy code

//

pages/api/auth/[...nextauth].js

```
import NextAuth from 'next-auth';
import Providers from 'next-auth/providers';

export default NextAuth({
  providers: [
    // Configurar proveedores aquí
  ],
  callbacks: {
    async signIn(user, account, profile) {
      // Ejecutar lógica personalizada en el inicio de sesión
      if (account.provider === 'google') {
        // Verificar si el correo electrónico está en una lista permitida
        if (profile.email.endsWith('@example.com')) {
          return true;
        } else {
          return false;
        }
      }
    }
  }
});
```

```
    return true;
  },
  async session(session, user) {
    // Modificar el objeto de sesión antes de retornarlo
    session.userId = user.id;
    return session;
  },
});
```

En este ejemplo, el callback `signIn` verifica que el correo electrónico del usuario pertenezca a un dominio específico antes de permitir el inicio de sesión. El callback `session` modifica el objeto de sesión para incluir el ID del usuario.

Crear Usuario Basado en OAuth y Login Personalizado

Con NextAuth.js, puedes crear un usuario automáticamente al iniciar sesión con un proveedor OAuth, o implementar un login personalizado.

Crear usuario con OAuth:

Cuando un usuario inicia sesión con OAuth, NextAuth.js puede crear automáticamente un registro de usuario en la base de datos configurada.

Login personalizado:

Para implementar un login personalizado, define una interfaz de usuario y utiliza el proveedor de credenciales para validar las credenciales del usuario.

Middlewares

Next.js y NextAuth.js permiten la implementación de middlewares para proteger rutas y ejecutar lógica personalizada antes de servir una página.

Ejemplo de middleware de autenticación:

jsx

Copy code

```
import { getSession } from 'next-auth/react';

export default async function middleware(req, res, next) {
  const session = await getSession({ req });

  if (!session) {
    // Redirigir a la página de inicio de sesión si no está autenticado
    res.writeHead(302, { Location: '/login' });
    res.end();
    return;
  }

  // Continuar con la solicitud si está autenticado
  next();
}
```

Este middleware verifica si el usuario tiene una sesión activa antes de permitir el acceso a una página protegida. Si el usuario no está autenticado, se redirige a la página de inicio de sesión.

Practica:

Código:

```
// components/AdminLayout.tsx
"use client";
import React from "react";

interface AdminLayoutProps {
  children: React.ReactNode;
}

const AdminLayout: React.FC<AdminLayoutProps> = ({ children }) => {
  return (
    <div className="min-h-screen flex flex-col">
```

```
<header className="bg-gray-800 text-white p-4">
  <h1 className="text-3xl">Sitio Web con React</h1>
</header>
<div className="flex flex-1">
  <aside className="w-64 bg-gray-700 text-white p-4 flex flex-col
justify-between">
    <div>
      <nav>
        <ul>
          <li className="mb-2">
            <a href="#" className="hover:underline">
              Home
            </a>
          </li>
          <li className="mb-2">
            <a href="#" className="hover:underline">
              Products
            </a>
          </li>
          <li className="mb-2">
            <a href="#" className="hover:underline">
              Services
            </a>
          </li>
          <li className="mb-2">
            <a href="#" className="hover:underline">
              Contact
            </a>
          </li>
          <li className="mb-2">
            <a href="#" className="hover:underline">
              Other
            </a>
          </li>
        </ul>
      </nav>
    </div>
    <div className="mt-4">
      <div className="flex flex-col items-center">
        
        <span>Cristian Torres Cordova</span>
      </div>
    </div>
  </aside>
  <main className="flex-1 p-4 bg-gray-100">{children}</main>
```



```

        </div>
        <footer className="bg-gray-800 text-white p-4 text-center">
            &copy; Derechos reservados: Cristian Torres Cordova 2024
        </footer>
    </div>
    );
};

export default AdminLayout;

// components/CreatePostForm.tsx
"use client";
import React, { useState } from "react";

interface NewPost {
    userId: number;
    id: number;
    title: string;
    body: string;
}

interface Props {
    onAddPost: (newPost: NewPost) => void;
}

const CreatePostForm: React.FC<Props> = ({ onAddPost }) => {
    const [userId, setUserId] = useState<number>(0);
    const [id, setId] = useState<number>(0);
    const [title, setTitle] = useState<string>("");
    const [body, setBody] = useState<string>("");

    const handleSubmit = (event: React.FormEvent) => {
        event.preventDefault();
        const newPost: NewPost = {
            userId,
            id,
            title,
            body,
        };
        onAddPost(newPost);
        // Limpiar los campos después de agregar la publicación
        setUserId(0);
        setId(0);
        setTitle("");
        setBody("");
    };

    return (
        <form

```

```

onSubmit={handleSubmit}
className="space-y-4 p-4 bg-white shadow-md rounded-md"
>
<div>
  <label className="block text-sm font-medium text-gray-700">
    ID de Usuario:
  </label>
  <input
    type="number"
    value={userId}
    onChange={(e) => setUserId(parseInt(e.target.value))}
    required
    className="mt-1 p-2 w-full border border-gray-300 rounded-md"
  />
</div>
<div>
  <label className="block text-sm font-medium text-gray-
700">ID:</label>
  <input
    type="number"
    value={id}
    onChange={(e) => setId(parseInt(e.target.value))}
    required
    className="mt-1 p-2 w-full border border-gray-300 rounded-md"
  />
</div>
<div>
  <label className="block text-sm font-medium text-gray-700">
    Titulo:
  </label>
  <input
    type="text"
    value={title}
    onChange={(e) => setTitle(e.target.value)}
    required
    className="mt-1 p-2 w-full border border-gray-300 rounded-md"
  />
</div>
<div>
  <label className="block text-sm font-medium text-gray-700">
    Cuerpo:
  </label>
  <textarea
    value={body}
    onChange={(e) => setBody(e.target.value)}
    required
    className="mt-1 p-2 w-full border border-gray-300 rounded-md"
  />
</div>

```

```

        <div>
          <button
            type="submit"
            className="w-full py-2 px-4 bg-blue-600 text-white rounded-md
            hover:bg-blue-700"
          >
            Añadir
          </button>
        </div>
      </form>
    );
  };

export default CreatePostForm;

// components/DeletePostButton.tsx
"use client";
import React from "react";

interface Props {
  postId: number;
  onDeletePost: (postId: number) => void;
}

const DeletePostButton: React.FC<Props> = ({ postId, onDeletePost }) => {
  const handleDelete = () => {
    onDeletePost(postId);
  };

  return <button onClick={handleDelete}>Eliminar</button>;
};

export default DeletePostButton;

// components/EditPostForm.tsx
"use client";
import React, { useState } from "react";

interface Post {
  userId: number;
  id: number;
  title: string;
  body: string;
}

interface Props {
  post: Post;
  onUpdatePost: (updatedPost: Post) => void;
  onCancel: () => void;
}

```

```

}

const EditPostForm: React.FC<Props> = ({ post, onUpdatePost, onCancel }) => {
  const [userId, setUserId] = useState<number>(post.userId);
  const [id, setId] = useState<number>(post.id);
  const [title, setTitle] = useState<string>(post.title);
  const [body, setBody] = useState<string>(post.body);

  const handleSubmit = (event: React.FormEvent) => {
    event.preventDefault();
    const updatedPost: Post = {
      userId,
      id,
      title,
      body,
    };
    onUpdatePost(updatedPost);
  };

  return (
    <form
      onSubmit={handleSubmit}
      className="space-y-4 p-4 bg-white shadow-md rounded-md"
    >
      <div>
        <label className="block text-sm font-medium text-gray-700">
          ID de Usuario:
        </label>
        <input
          type="number"
          value={userId}
          onChange={(e) => setUserId(parseInt(e.target.value))}
          required
          className="mt-1 p-2 w-full border border-gray-300 rounded-md"
        />
      </div>
      <div>
        <label className="block text-sm font-medium text-gray-700">ID:</label>
        <input
          type="number"
          value={id}
          onChange={(e) => setId(parseInt(e.target.value))}
          required
          className="mt-1 p-2 w-full border border-gray-300 rounded-md"
        />
      </div>
    </div>
  );
};

```

```

        <label className="block text-sm font-medium text-gray-700">
          Título:
        </label>
        <input
          type="text"
          value={title}
          onChange={(e) => setTitle(e.target.value)}
          required
          className="mt-1 p-2 w-full border border-gray-300 rounded-md"
        />
      </div>
      <div>
        <label className="block text-sm font-medium text-gray-700">
          Cuerpo:
        </label>
        <textarea
          value={body}
          onChange={(e) => setBody(e.target.value)}
          required
          className="mt-1 p-2 w-full border border-gray-300 rounded-md"
        />
      </div>
      <div className="flex space-x-2">
        <button
          type="submit"
          className="py-2 px-4 bg-blue-600 text-white rounded-md
hover:bg-blue-700"
        >
          Actualizar
        </button>
        <button
          type="button"
          onClick={onCancel}
          className="py-2 px-4 bg-gray-600 text-white rounded-md
hover:bg-gray-700"
        >
          Cancelar
        </button>
      </div>
    </form>
  );
};
export default EditPostForm;

// components/LoginForm.tsx
"use client";
import React, { useState } from "react";

interface Props {

```

```

    onLoginSuccess: () => void;
  }

const LoginForm: React.FC<Props> = ({ onLoginSuccess }) => {
  const [username, setUsername] = useState<string>("");
  const [password, setPassword] = useState<string>("");
  const [error, setError] = useState<string | null>(null);

  const handleLogin = async (event: React.FormEvent) => {
    event.preventDefault();
    try {
      const response = await fetch(
        "https://jsonplaceholder.typicode.com/users"
      );
      const users = await response.json();
      const user = users.find(
        (u: any) => u.username === username && u.password === password
      );

      if (user) {
        onLoginSuccess();
      } else {
        setError("Usuario no valido");
      }
    } catch (err) {
      setError("An error occurred while trying to log in");
    }
  };

  return (
    <div className="max-w-md mx-auto mt-10 p-4 bg-white shadow-md rounded-md">
      <h1 className="text-2xl font-bold mb-4">Ingresar al Sistema</h1>
      <form onSubmit={handleLogin} className="space-y-4">
        <div>
          <label className="block text-sm font-medium text-gray-700">
            Usuario
          </label>
          <input
            type="text"
            value={username}
            onChange={(e) => setUsername(e.target.value)}
            required
            className="mt-1 p-2 w-full border border-gray-300 rounded-md"
          />
        </div>
        <div>
          <label className="block text-sm font-medium text-gray-700">
            Contraseña

```

```

        </label>
        <input
          type="password"
          value={password}
          onChange={(e) => setPassword(e.target.value)}
          required
          className="mt-1 p-2 w-full border border-gray-300 rounded-md"
        />
      </div>
      {error && <div className="text-red-600">{error}</div>}
      <button
        type="submit"
        className="w-full py-2 px-4 bg-blue-600 text-white rounded-md
hover:bg-blue-700"
      >
        Ingresar
      </button>
    </form>
  </div>
);
};

export default LoginForm;

// components/LoginForm.tsx
"use client";
import React, { useState } from "react";

interface Props {
  onLoginSuccess: () => void;
}

const LoginForm: React.FC<Props> = ({ onLoginSuccess }) => {
  const [username, setUsername] = useState<string>("");
  const [password, setPassword] = useState<string>("");
  const [error, setError] = useState<string | null>(null);

  const handleLogin = async (event: React.FormEvent) => {
    event.preventDefault();
    try {
      const response = await fetch(
        "https://jsonplaceholder.typicode.com/users"
      );
      const users = await response.json();
      const user = users.find(
        (u: any) => u.username === username && u.password === password
      );

      if (user) {

```



```

};

export default LoginForm;

// pages/index.tsx
"use client";
import React, { useState } from "react";
import PostList from "../../components/PostList";
import LoginForm from "../../components/LoginForm";
import AdminLayout from "../../components/AdminLayout";

const Home = () => {
  const [isAuthenticated, setIsAuthenticated] = useState<boolean>(false);

  const handleLoginSuccess = () => {
    setIsAuthenticated(true);
  };

  return (
    <div>
      {isAuthenticated ? (
        <AdminLayout>
          <PostList />
        </AdminLayout>
      ) : (
        <LoginForm onLoginSuccess={handleLoginSuccess} />
      )}
    </div>
  );
};

export default Home;

import type { Metadata } from "next";
import { Inter } from "next/font/google";
import "../globals.css";

const inter = Inter({ subsets: ["latin"] });

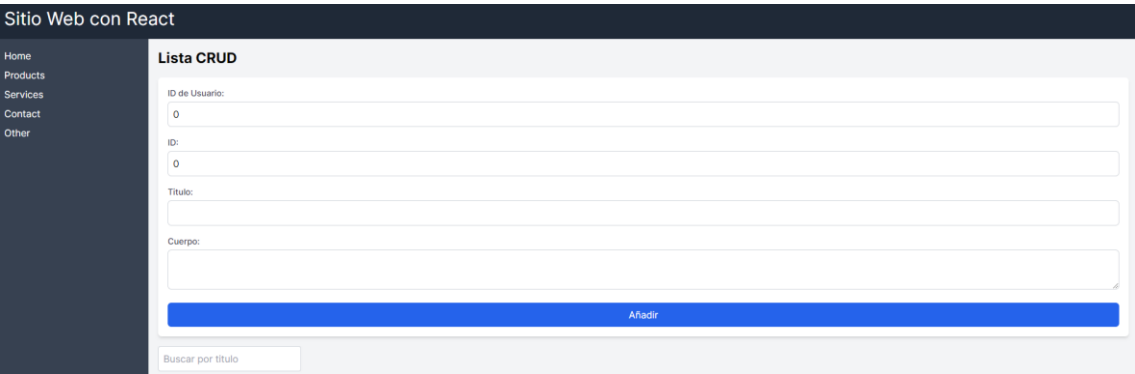
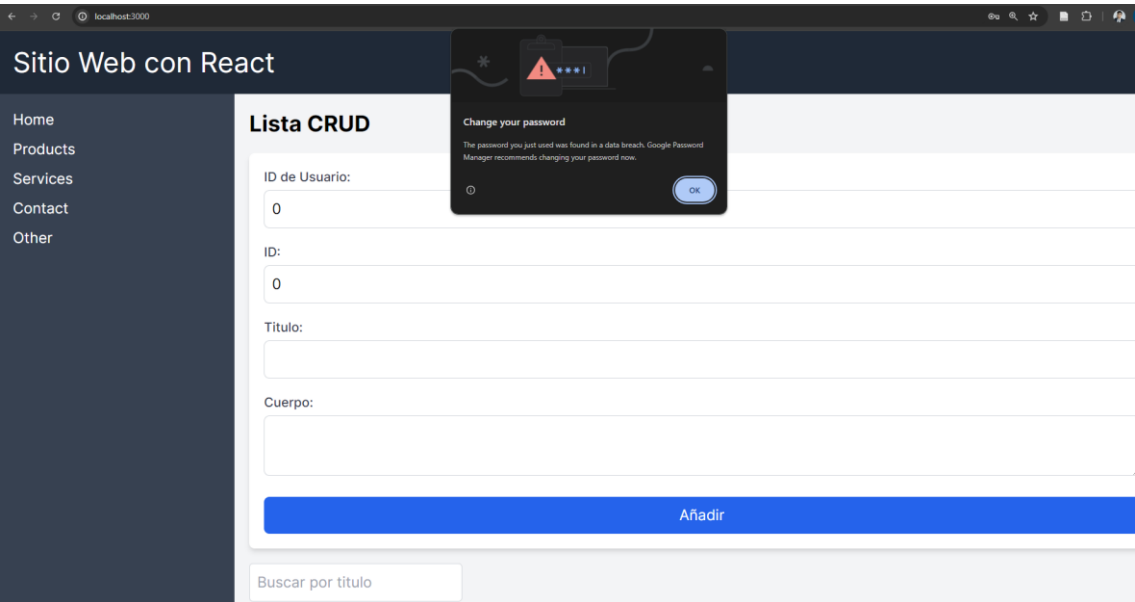
export const metadata: Metadata = {
  title: "Create Next App",
  description: "Generated by create next app",
};


export default function RootLayout({
  children,
}: Readonly<{
  children: React.ReactNode;
}>) {

```

```
return (  
  <html lang="en">  
    <body className={inter.className}>{children}</body>  
  </html>  
);  
}
```

Vista desde el LocalHost:






Cristian Torres Cordova

Buscar por título

ID de Usuario	ID	Título	Cuerpo	Acciones
1	1	sunt aut facere repellat provident occaecati excepturi optio reprehenderit	quia et suscipit suscipit recusandae consequuntur expedita et cum reprehenderit molestiae ut ut quas totam nostrum rerum est autem sunt rem eveniet architecto	Editar Eliminar
1	2	qui est esse	est rerum tempore vitae sequi sint nihil reprehenderit dolor beatae ea dolores neque fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis qui aperiam non debitis possimus qui neque nisi nulla	Editar Eliminar
1	3	ea molestias quasi exercitationem repellat qui ipsa sit aut	et iusto sed quo iure voluptatem occaecati omnis eligendi aut ad voluptatem doloribus vel accusantium quis pariatur molestiae porro eius odio et labore et velit aut	Editar Eliminar
1	4	eum et est occaecati	ullam et saepe reiciendis voluptatem adipisci sit amet autem assumenda provident rerum culpa quis hic commodi nesciunt rem tenetur doloremque ipsam iure quis sunt voluptatem rerum illo velit	Editar Eliminar
1	5	nesciunt quas odio	repudiandae veniam quaerat sunt sed alias aut fugiat sit autem sed est voluptatem omnis possimus esse voluptatibus quis est aut tenetur dolor neque	Editar Eliminar
1	6	dolorem eum magni eos aperiam quia	ut aspernatur corporis harum nihil quis provident sequi mollitia nobis aliquid molestiae perspiciatis et ea nemo ab reprehenderit accusantium quas voluptate dolores velit et doloremque molestiae	Editar Eliminar
1	7	magnam facilis autem	dolore placeat quibusdam ea quo vitae magni quis enim qui quis quo nemo aut saepe quidem repellat excepturi ut quia sunt ut sequi eos ea sed quas	Editar Eliminar
1	8	dolorem dolore est ipsam	dignissimos aperiam dolorem qui eum facilis quibusdam animi sint suscipit qui sint possimus cum quaerat magni maiores excepturi ipsam ut commodi dolor voluptatum modi aut vitae	Editar Eliminar

12345678910111213

Derechos reservados: Cristian Torres Cordova 2024



Cristian Torres Cordova

nu

ID de Usuario	ID	Título	Cuerpo	Acciones
3	23	maxime id vitae nihil numquam	veritatis unde neque eligendi quae quod architecto quo neque vitae est illo sit tempora doloremque fugit quod et et vel beatae sequi ullam sed tenetur perspiciatis	Editar Eliminar
3	28	delectus ullam et corporis nulla voluptas sequi	non et quaerat ex quae ad maiores maiores recusandae totam aut blanditiis mollitia quas illo ut voluptatibus voluptatem similique nostrum eum	Editar Eliminar
5	45	ut numquam possimus omnis eius suscipit laudantium iure	est natus reiciendis nihil possimus aut provident ex et dolor repellat pariatur est nobis rerum repellendus dolorem autem	Editar Eliminar
10	95	id minus libero illum nam ad officiis	earum voluptatem facere provident blanditiis velit laboriosam pariatur accusamus odio saepe cumque dolor qui a dicta ab doloribus consequatur omnis corporis cupiditate eaque assumenda ad nesciunt	Editar Eliminar
10	96	quaerat velit veniam amet cupiditate aut numquam ut sequi	in non odio excepturi sint eum labore voluptates vitae quia qui et inventore itaque rerum veniam non exercitationem delectus aut	Editar Eliminar

1

Derechos reservados: Cristian Torres Cordova 2024

Home

Products

Services

Contact

Other

Lista CRUD

ID de Usuario:

122

ID:

222


Título:

Machu

Cuerpo:

Picchu

Añadir



Cristian Torres Cordova

Buscar por título

ID de Usuario	ID	Título	Cuerpo	Acciones
10	97	quas fugiat ut perspiciatis vero provident	eum non blanditis soluta porro quibusdam voluptas vel voluptatem qui placeat dolores qui velit aut vel inventore aut cumque culpa explicabo aliquid at perspiciatis est et voluptatem dignissimos dolor itaque sit nam	<div>Editar</div> <div>Eliminar</div>
10	98	laboriosam dolor voluptates	doloremque ex facilis sit sint culpa soluta assumenda eligendi non ut eius sequi ducimus vel quasi veritatis est dolores	<div>Editar</div> <div>Eliminar</div>
10	99	temporibus sit alias delectus eligendi possimus magni	quo deleniti praesentium dicta non quod aut est molestias molestias et officia quis nihil itaque dolorem quia	<div>Editar</div> <div>Eliminar</div>
10	100	at nam consequatur ea labore ea harum	cupiditate quo est a modi nesciunt soluta ipsa voluptas error itaque dicta in autem qui minus magnam et distinctio eum accusamus ratione error aut	<div>Editar</div> <div>Eliminar</div>
122	222	Machu	Picchu	<div>Editar</div> <div>Eliminar</div>

123

Derechos reservados: Cristian Torres Cordova 2024

Home

Products

Services

Contact

Other

Lista CRUD

ID de Usuario:

10

ID:

99

Título:

temporibus sit alias delectus eligendi possimus magni

Cuerpo:

quo deleniti praesentium dicta non quod aut est molestias

Actualizar

Cancelar