

Manual De Usuarios

Programa de manejo de jardín personal.

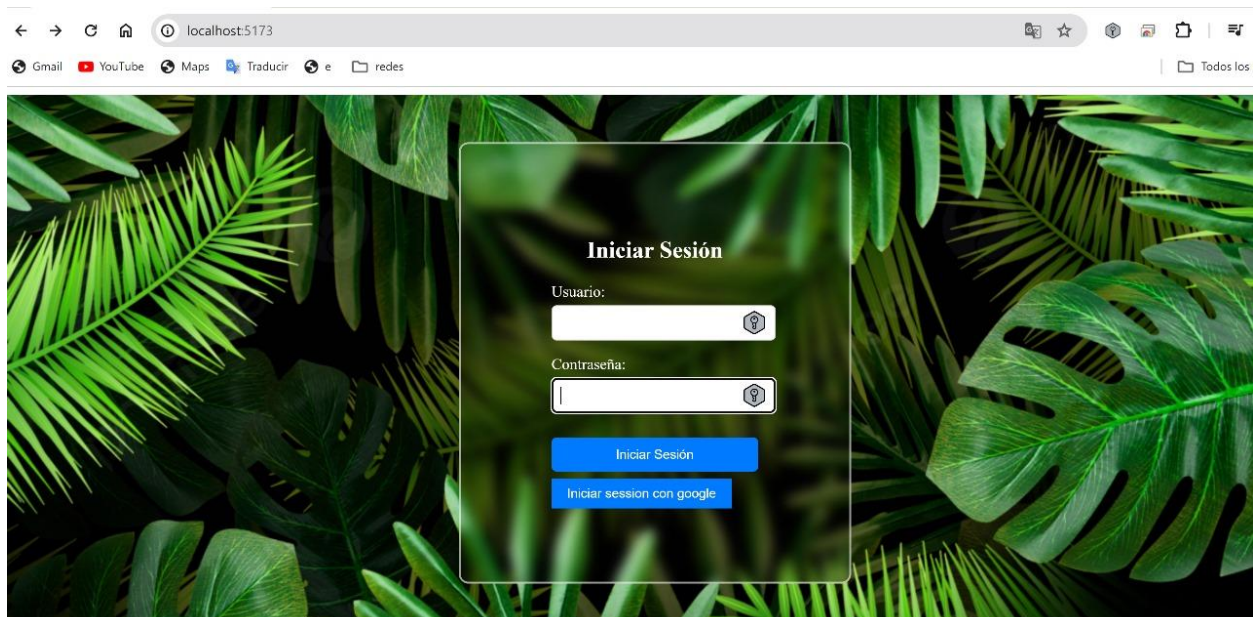
✓ Acceder a la API

Para comenzar, abriremos la API en la cual se trabajó, pensando en el usuario y el cuidado de sus plantas.

✓ Completar el Formulario de Inicio de Sesión

En la página de inicio de sesión, proporciona tu nombre de usuario (o correo electrónico) y tu contraseña en el formulario correspondiente, para que pueda iniciar sesión en la API.

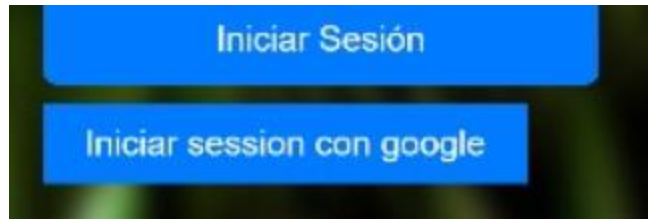
✓ Inicio de sesión



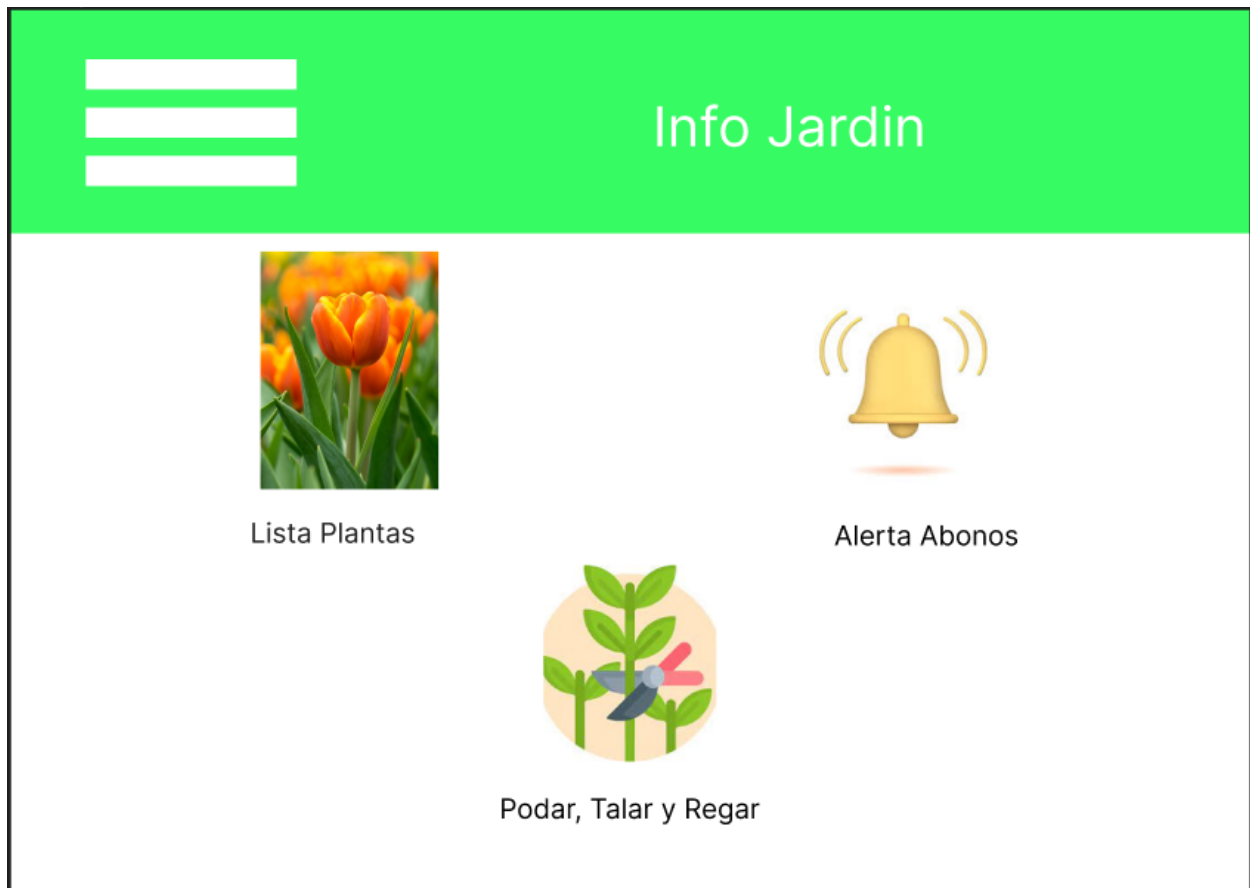
Opción 1: Usar credenciales de administrador.

- Usuario: usuario@gmail.com
- Contraseña: Password

Opción 2: usar una cuenta de Email vigente.



- ✓ Seleccionar la acción deseada por el usuario



Opción 1: lista de plantas

Se selecciona Agregar planta.



Posteriormente de seleccionar la planta deseada procedemos a completar el formulario.

Demostración.

Agregar nueva planta

Nombre:

URL de la imagen:

Descripción:

Enviar

Llenamos los espacios con la información requerida.

Agregar nueva planta

Nombre:

Rosas

URL de la imagen:

<https://upload.wikimedia.org/wikipedia/commons/thumb/c>

Descripción:

principales de la familia de las rosáceas.
Se denomina rosa a la flor de los miembros
de este género y rosal a la planta.

Enviar

Presione enviar.

Enviar

Luego visualice sus resultados en pantalla.

Agregar planta



Listado de plantas



Rosas

El género Rosa está compuesto por un conocido grupo de arbustos generalmente espinosos y floridos representantes principales de la familia de las rosáceas. Se denomina rosa a la flor de los miembros de este género y rosal a la planta.

Eliminar planta

Es posible eliminar cada entrada.

Eliminar planta

Cualquier registro será eliminado si el botón es usado.

Agregar planta

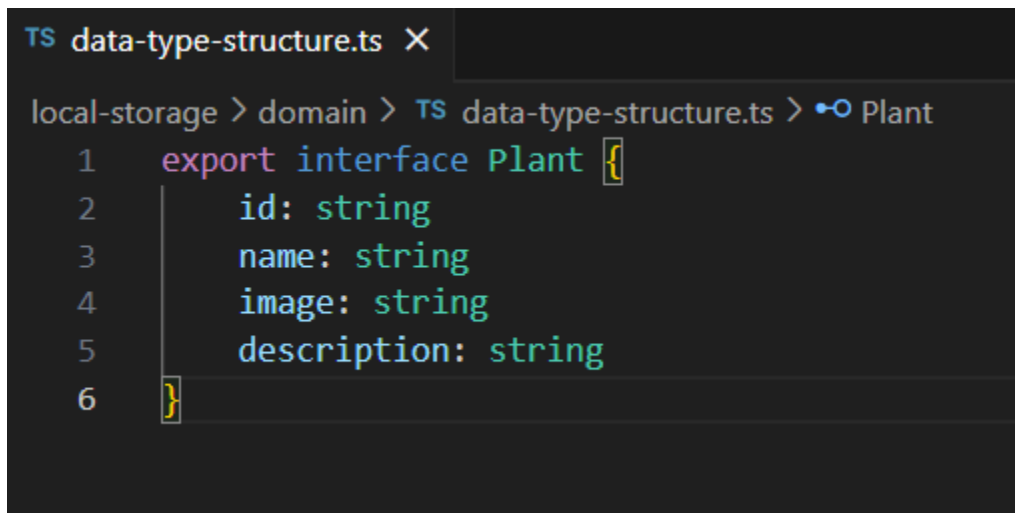


Listado de plantas

Manual técnico

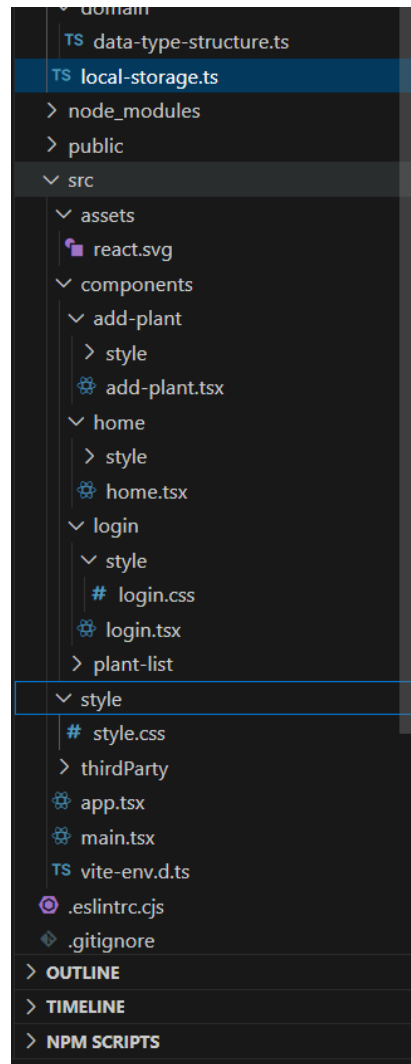
Manual técnico del programa de gestión de jardines personales.

Estructura formados por Strings



```
TS data-type-structure.ts X
local-storage > domain > TS data-type-structure.ts > Plant
1  export interface Plant {
2      id: string
3      name: string
4      image: string
5      description: string
6  }
```

Un "string" en programación es una secuencia de caracteres, como letras, números y símbolos, que se utilizan para representar texto. En muchos lenguajes de programación, los strings se escriben entre comillas simples (") o dobles (""). Por ejemplo, "hola mundo" es un string que contiene la frase "hola mundo". Los strings son una parte fundamental de la programación y se utilizan para manipular y representar texto en aplicaciones y software.



Estructura base del programa.

```

1  import { Plant } from "../domain/data-type-structure"
2
3  class LocalPlantStorage {
4
5      plantlist: Plant[]
6      constructor() {
7          // Cargar datos del localStorage al inicializar la instancia
8          this.plantlist = this.loadPlantsFromLocalStorage();
9      }
10
11     addPlant(plant: Plant) {
12         this.plantlist.push(plant);
13         // Guardar la lista actualizada en el localStorage
14         this.savePlantsToLocalStorage(this.plantlist);
15     }
16
17     deletePlant(plantID: string) {
18         const indice = this.plantlist.findIndex(elemento => elemento.id === plantID);
19         if (indice !== -1) {
20             this.plantlist.splice(indice, 1);
21             // Guardar la lista actualizada en el localStorage
22             this.savePlantsToLocalStorage(this.plantlist);
23         }
24     }
25
26     // Método para guardar la lista de plantas en el localStorage
27     private savePlantsToLocalStorage(plants: Plant[]) {
28         localStorage.setItem('plants', JSON.stringify(plants));
29     }
30
31     // Método para cargar la lista de plantas desde el localStorage
32     private loadPlantsFromLocalStorage(): Plant[] {
33         const plantsJSON = localStorage.getItem('plants');
34         if (plantsJSON) {
35             return JSON.parse(plantsJSON);
36         }
37         return []; // Si no hay datos en el localStorage, devolver un arreglo vacío

```

Se forman los constructores y la lógica de la lista.

```

import { useState } from "react";
import { v4 as uuidv4 } from "uuid";
import localPlantStorage from "../../local-storage/local-storage";
import "../style/style.css"; // Archivo de estilos

const Formulario = () => {
  // Estado para almacenar los valores del formulario
  const [nombre, setNombre] = useState("");
  const [imagen, setImagen] = useState("");
  const [descripcion, setDescripcion] = useState("");

  // Manejadores de cambios en los inputs
  const handleNombreChange = event => {
    setNombre(event.target.value);
  };

  const handleImagenChange = event => {
    setImagen(event.target.value);
  };

  const handleDescripcionChange = event => {
    setDescripcion(event.target.value);
  };

  // Manejador de envío del formulario
  const handleSubmit = event => {
    event.preventDefault();
    // Aquí puedes enviar los datos a donde los necesites, por ejemplo, a través de una función prop
    localPlantStorage.addPlant({
      id: uuidv4(),
      name: nombre,
      image: imagen,
      description: descripcion
    });
    // Limpia el formulario después del envío
    setNombre("");
  };
};

```

```

    setNombre("");
    setImagen("");
    setDescripcion("");
  };

  return (
    <div className="formulario-container">
      <h2 className="header">Agregar nueva planta</h2>
      <form onSubmit={handleSubmit}>
        <div className="form-group">
          <label htmlFor="nombre">Nombre:</label>
          <input
            type="text"
            id="nombre"
            value={nombre}
            onChange={handleNombreChange}
            required
          />
        </div>
        <div className="form-group">
          <label htmlFor="imagen">URL de la imagen:</label>
          <input
            type="text"
            id="imagen"
            value={imagen}
            onChange={handleImagenChange}
            required
          />
        </div>
        <div className="form-group">
          <label htmlFor="descripcion">Descripción:</label>
          <textarea
            id="descripcion"
            value={descripcion}
            onChange={handleDescripcionChange}
          />
        </div>
      </form>
    </div>
  );

```

```

        <label htmlFor="descripcion">Descripcion</label>
        <textarea
          id="descripcion"
          value={descripcion}
          onChange={handleDescripcionChange}
          required
        />
      </div>
      <button type="submit">Enviar</button>
    </form>
  </div>
);
};

export default Formulario;

```

Creación de la lista.

Creación de las opciones en el menú de inicio.

```
import { Link } from "react-router-dom";
import "../style/style.css"; // Archivo CSS para estilos

function App() {
  return (
    <div className="App">
      <header className="header">
        <h1>Información Jardín</h1>
      </header>
      <div className="options-container">
        <Link to="/plant-list">
          <div className="option">
            
            <h2>Lista de plantas</h2>
          </div>
        </Link>
        <Link to="/abono">
          <div className="option">
            
            <h2>Alerta abonos</h2>
          </div>
        </Link>
      </div>
    </div>
  );
}
```

Creación del login.

```
import { useState } from "react";
import FirebaseApp from "../../thirdParty/firebase";
import "../style/login.css";
import { useNavigate } from "react-router-dom";

const LoginForm = () => {
  // Definir el estado para el nombre de usuario y la contraseña
  const navigate = useNavigate();
  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");

  // Manejar cambios en el campo de nombre de usuario
  const handleUsernameChange = (event: any) => {
    setUsername(event.target.value);
  };

  // Manejar cambios en el campo de contraseña
  const handlePasswordChange = (event: any) => {
    setPassword(event.target.value);
  };

  // Manejar envío del formulario
  const handleSubmit = async (event: any) => {
    try {
      event.preventDefault();
      const loginUser = await FirebaseApp.loginWithEmailAndPassword(
        username,
        password
      );
      /**
       * TODO: add local storage
       */
      if(loginUser.user.uid){
        navigate("/home");
      }
    } catch (err){}
```



```

37
38 };
39
40
41 const LogInWithGoogle = async () => {
42   try {
43     const googleAuthResponse = await FirebaseAuth.signInWithGoogle();
44     /**
45      * TODO: add local storage
46      */
47     if(googleAuthResponse?.idToken){
48       navigate("/home");
49     }
50   } catch (err){}
51 };
52
53 return (
54   <div className="login-container">
55     <h2>Iniciar Sesión</h2>
56     <form onSubmit={handleSubmit} className="login-form">
57       <div className="form-group">
58         <label htmlFor="username">Usuario:</label>
59         <input
60           type="text"
61           id="username"
62           value={username}
63           onChange={handleUsernameChange}
64           className="form-control"
65         />
66       </div>
67       <div className="form-group">
68         <label htmlFor="password">Contraseña:</label>
69         <input
70           type="password"
71           id="password"

```

```

37
38   };
39
40
41   const LogInWithGoogle = async () => {
42     try {
43       const googleAuthResponse = await FirebaseAuth.signInWithCredential(
44         /**
45          * TODO: add local storage
46          */
47         googleAuthResponse?.idToken){
48         navigate("/home");
49       }
50     } catch (err){}
51   };
52
53   return (
54     <div className="login-container">
55       <h2>Iniciar Sesión</h2>
56       <form onSubmit={handleSubmit} className="login-form">
57         <div className="form-group">
58           <label htmlFor="username">Usuario:</label>
59           <input
60             type="text"
61             id="username"
62             value={username}
63             onChange={handleUsernameChange}
64             className="form-control"
65           />
66         </div>
67         <div className="form-group">
68           <label htmlFor="password">Contraseña:</label>
69           <input
70             type="password"
71             id="password"

```

```

        type="password"
        id="password"
        value={password}
        onChange={handlePasswordChange}
        className="form-control"
      />
    </div>
    <button type="submit" className="btn-login">
      Iniciar Sesión
    </button>
  </form>
  <button onClick={LogInWithGoogle}>Iniciar session con google</button>
</div>
);
};

export default LoginForm;

```