UNIVERSIDAD TECNOLÓGICA DEL VALLE DEL MEZQUITAL

MATERIA:

DESARROLLO WEB INTEGRAL

DOCENTE:

ING. SANTIAGO LABRA HERNÁDEZ

ALUMNO:

ALDO URIEL MARTINEZ ACEVEDO

TRABAJO:

ANGULAR Y TAILWIND

CUATRIMESTRE: 9 **GRUPO:** “B”

1. Introducción

En esta práctica, desarrollamos una aplicación To-Do List utilizando:

- Angular 17 (con Signals para manejo de estado reactivo).
- Tailwind CSS para estilizado.
- Formularios reactivos con validaciones.
- Persistencia de datos en localStorage.

El objetivo fue implementar las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) con un filtrado dinámico y una interfaz funcional.

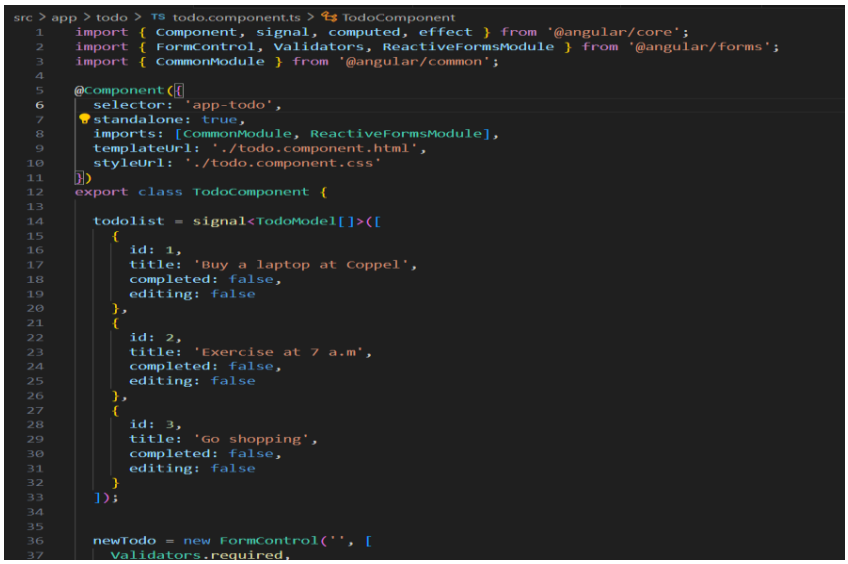
2. Desarrollo

Pasos realizados

1. Configuración inicial

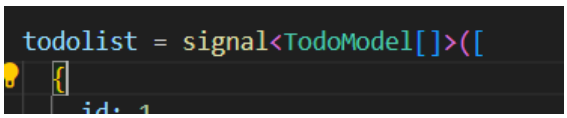
Se creó un proyecto Angular con Tailwind CSS.

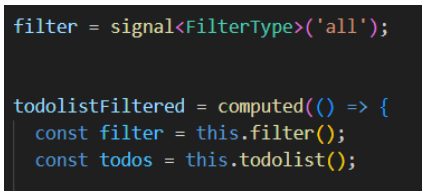
Se definieron las interfaces y tipos:



2. Implementación de Signals

Señales principales:





3.Funciones CRUD

Añadir tarea:

```
addTodo() {
  const newTodoTitle = this.newTodo.value?.trim();
  if (this.newTodo.valid && newTodoTitle) {
    this.todolist.update((prev_todos) => [
      ...prev_todos,
      {
        id: Date.now(),
        title: newTodoTitle,
        completed: false,
        editing: false
      }
    ]);
    this.newTodo.reset();
  }
}
```

Editar tarea:

```
updateTodoEditingMode(todoId: number) {
  this.todolist.update((prev_todos) =>
    prev_todos.map((todo) => ({
      ...todo,
      editing: todo.id === todoId ? true : false
    }))
  );
}
```

Eliminar tarea:

```
removeTodo(todoId: number) {
  this.todolist.update((prev_todos) =>
    prev_todos.filter((todo) => todo.id !== todoId)
  );
}
```

4. Filtrado

Lógica de filtros:

```
todolistFiltered = computed(() => {
  const filter = this.filter();
  const todos = this.todolist();

  switch (filter) {
    case 'active':
      return todos.filter((todo) => !todo.completed);
    case 'completed':
      return todos.filter((todo) => todo.completed);
    default:
      return todos;
  }
});
```

5. Persistencia con localStorage

Guardado automático:

```
constructor() {
  effect(() => {
    localStorage.setItem('todos', JSON.stringify(this.todolist()));
  });
}
```

Carga inicial:

```
ngOnInit() {
  const storage = localStorage.getItem('todos');
  if (storage) {
    this.todolist.set(JSON.parse(storage));
  }
}
```

3. Resultados Obtenidos

Aplicación funcional:

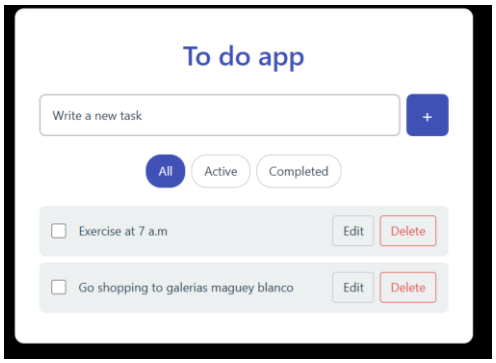
- Lista de tareas con CRUD completo.
- Filtros dinámicos (All, Active, Completed).
- Persistencia de datos al recargar.

Interfaz:

- Estilo oscuro con Tailwind CSS.
- Responsive y accesible.

Validaciones:

- Input requiere mínimo 5 caracteres.
- Solo tareas no completadas pueden editarse.



Conclusión

Con esta práctica aprendí a usar Angular Signals para manejar el estado de la aplicación sin complicaciones, usando signal(), computed() y effect(). También descubrí lo práctico que es Tailwind CSS para dar estilos rápido con sus clases predefinidas. Aprendí a validar formularios con FormControl y a guardar datos en el navegador con localStorage. Además, mejoré en TypeScript usando tipos bien definidos . Aunque tuve algunos problemas con la configuración de Tailwind, al final todo funcionó y ahora entiendo mejor cómo armar una app con Angular.