TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Mexicali

Ing. Sistemas Computacionales

Desarrollo de Aplicaciones Web

Docente: Jose Ramon Bogarin Valenzuela

“Tarea U3 – Problema de conectividad Frontend y Backend para aplicación ToDo”

Alumna: Rodriguez Herrera Maria Fernanda

Mexicali, B.C. **28-Oct-2025**

• **Backend:** en el backend de la aplicación, específicamente en el archivo de ***TodoServicesImpl.java*** pude notar que el estatus que se le asignaba a la tarea siempre era “COMPLETED”, por lo que no tomaba en cuenta los demás estados. Para solucionarlo se quitó la línea de código que establecía el estatus como completado que es ***todoEntity.setStatus(TodoStatus.COMPLETED);***

```
@Override
public TodoEntity updateTodoById(int idTodo, UpdateTodoRequest entity) {
    TodoEntity todoEntity = todoRepository.findById((long)idTodo).orElse(null);
    if(todoEntity!=null){
        todoEntity.setTitle(entity.getTitle());
        todoEntity.setDescription(entity.getDescription());
        // el error viene desde aqui
        todoEntity.setStatus(TodoStatus.COMPLETED);
        todoRepository.save(todoEntity);
        return todoEntity;
    }
    return null;
}
```

Y se reemplazó a como debería de estar, que en este caso es ***todoEntity.setStatus(entity.getStatus());***, ya que se debe obtener el estatus que se esta seleccionando con el ***getStatus*****:**

```
@Override
public TodoEntity updateTodoById(int idTodo, UpdateTodoRequest entity) {
    TodoEntity todoEntity = todoRepository.findById((long)idTodo).orElse(null);
    if(todoEntity!=null){
        todoEntity.setTitle(entity.getTitle());
        todoEntity.setDescription(entity.getDescription());
        // el error viene desde aqui
        //todoEntity.setStatus(TodoStatus.COMPLETED);
        todoEntity.setStatus(entity.getStatus());
        todoRepository.save(todoEntity);
        return todoEntity;
    }
    return null;
}
```

• **Frontend:** en el caso del frontend, en el archivo de ***useTodos.ts*** se tiene esto, lo cual alterna entre los estados de las tareas comparándolas entre COMPLETED o PENDING lo cual, de nueva cuenta, no toma en cuenta los demás estatus. Para solucionarlo se quitó la línea de código ***const newStatus: TodoStatus = todo.status === 'COMPLETED' ? 'PENDING' : 'COMPLETED';***

```
const toggleTodo = async (id: number) => {
    const todo = todos.find(t => t.id === id);
    if (!todo) return; // Si no existe la tarea, no hace nada

    try {
        // Alterna entre COMPLETED y PENDING
        const newStatus: TodoStatus = todo.status === 'COMPLETED' ? 'PENDING' : 'COMPLETED';
        // Actualiza en el backend
        const updatedTodo = await todosApi.updateTodo(id, {
            title: todo.title,
            description: todo.description,
            status: newStatus
        });
        // Actualiza el estado local reemplazando la tarea modificada
        setTodos(todos.map(t => t.id === id ? updatedTodo : t));
    } catch (error) {
        handleApiError(error, 'Error al actualizar la tarea');
    }
};
```

Y se cambió por ***const newStatus: TodoStatus = todo.status;***, la cual asigna a la variable entre los valores de los estados de las tareas:

```
const toggleTodo = async (id: number) => {
    const todo = todos.find(t => t.id === id);
    if (!todo) return; // Si no existe la tarea, no hace nada

    try {
        // Alterna entre COMPLETED y PENDING
        //const newStatus: TodoStatus = todo.status === 'COMPLETED' ? 'PENDING' : 'COMPLETED';
        const newStatus: TodoStatus = todo.status;
        // Actualiza en el backend
        const updatedTodo = await todosApi.updateTodo(id, {
            title: todo.title,
            description: todo.description,
            status: newStatus
        });
        // Actualiza el estado local reemplazando la tarea modificada
        setTodos(todos.map(t => t.id === id ? updatedTodo : t));
    } catch (error) {
        handleApiError(error, 'Error al actualizar la tarea');
    }
};
```