Comunicación entre componentes
Segunda Parte

Ph. D. Lenin G. Lemus Zúñiga*

October 6, 2019

Contents

1 Objetivo

Partiendo de la aplicacióin básica, creada siguiendo las indicaciones el fichero practicaNG801.pdf, se va a agregar la lógica necesaria para mostrar y editar tareas a realizar (los "todos").

*lenin.lemus@hotmail.com

2 Prerequisitos

Acción recomendada: Realizar las indicaciones del fichero con la practicaNG801.pdf y continuar con las indicaciones de esta práctica.
De forma alternativa, descargue de la página lemus.webs.upv.es. Descomprima el fichero en la carpeta c:
cursoNG8. Verifique que la aplicación está operativa, ejecutando los siguientes comandos:

```
c:\> cd c:\cursoNG8\practicaNG801
c:\cursoNG8\practicaNG801> npm i
c:\cursoNG8\practicaNG801> ng start -o
```

Listing 1: Aplicación operativa

En el navegador por defecto aparecerá la aplicación. Mediante el uso de un editor, se sugiere Visual Studio Code. Abra la carpeta c:
cursoNG8
practicaNG801 y proceda a realizar los cambios que se detallan en las siguientes secciones.

2.1 Creación y configuración de los modelos

Crear los modelos siguientes:

- todo
- address
- student
- visitedSites

```
c:\cursoNG8practicaNG801> ng generate class models/todo
c:\cursoNG8practicaNG801> ng generate class models/address
c:\cursoNG8practicaNG801> ng generate class models/student
c:\cursoNG8practicaNG801> ng generate class models/visitedSites
```

2.1.1 fichero todo.ts

```
export interface ITodo {
    id?: string;
    title: string;
    description: string;
    isCompleted: boolean;
}
export class Todo implements ITodo {
    id?: string;
    title: string;
    description: string;
    isCompleted: boolean;
```

```
    index?: number;

    constructor(o?: ITodo) {
        this.id = o && o.id || undefined;
        this.title = o && o.title || 'Title';
        this.description = o && o.description || 'Description';
        this.isCompleted = o && o.isCompleted || false;
    }
}
```

2.1.2 fichero address.ts

```
export interface IAddress {
    streetType: string;
    streetName: string;
    streetNumber: string;
    zipcode: string;
    city: string;
    state: string;
    country: string;
}
export class Address implements IAddress {
    streetType: string;
    streetName: string;
    streetNumber: string;
    zipcode: string;
    city: string;
    state: string;
    country: string;

    constructor(o?: IAddress){
        this.streetType = o && o.streetType || 'Av.';
        this.streetName = o && o.streetName || 'Puerto';
        this.streetNumber = o && o.streetNumber || 'S/N';
        this.zipcode = o && o.zipcode || '46001';
        this.city = o && o.city || 'València';
        this.state = o && o.state || 'Valencia';
        this.country = o && o.country || 'Spain';
    }
}
```

2.1.3 fichero student.ts

```
import { Address } from './address';
```

```
import { Mobile } from './mobile';
import { Phone } from './phone';

export interface IStudent {
    username: string;
    password: string;
    dni: string;
    dniLetter: string;
    firstName: string;
    middleName?: string;
    lastName: string;
    lastNameOptional?: string;
    address: Address;
    emails: Array<Mobile>;
    phones: Array<Phone>;
}

export class Student implements IStudent {
    username: string;
    password: string;
    dni: string;
    dniLetter: string;
    firstName: string;
    middleName: string;
    lastName: string;
    lastNameOptional: string;
    address: Address;
    emails: Array<Mobile>;
    phones: Array<Phone>;

    constructor(u?: IStudent) {
        this.username = u && u.username || '';
        this.password = u && u.username || '';
        this.dni = u && u.username || '';
        this.dniLetter = u && u.username || '';
        this.firstName = u && u.username || '';
        this.middleName = u && u.username || '';
        this.lastName = u && u.username || '';
        this.lastNameOptional = u && u.username || '';
        this.address.streetType = u && u.address && u.address.streetType || '';
        this.address.streetName = u && u.address && u.address.streetName || '';
        this.address.streetNumber = u && u.address && u.address.streetNumber || '
        this.address.zipcode = u && u.address && u.address.zipcode || '';
        this.address.city = u && u.address && u.address.city || '';
        this.address.state = u && u.address && u.address.state || '';
```

```
        this.address.country = u && u.address && u.address.country || '';

        this.emails = new Array();
        for (const email of (u && u.emails) ) {
            this.emails.push( Object.assign({}, email));
        }

        this.phones = new Array();
        for (const phone of (u && u.phones) ) {
            this.phones.push( Object.assign({}, phone));
        }
    }
}
```

2.1.4 fichero visited-sites.ts

```
export interface IVisitedSite {
    id?: string;
    name: string;
    x: string;
    y: string;
    description: string;
    year: number;
    country: string;
    city: string;
    startDate: Date;
    endDate: Date;
}
export class VisitedSite implements IVisitedSite {

    id: string;
    name: string;
    x: string;
    y: string;
    description: string;
    year: number;
    country: string;
    city: string;
    startDate: Date;
    endDate: Date;

    constructor(o?: IVisitedSite) {
        this.id = o && o.id || undefined;
        this.name = o && o.name || 'name';
        this.x = o && o.x || 'x';
```

```
        this.y = o && o.y || 'y';
        this.description = o && o.description || 'description';
        this.year = o && o.year || 1996;
        this.country = o && o.country || 'country';
        this.city = o && o.city || 'city';
        this.startDate = o && o.startDate || new Date(1990, 1, 1);
        this.endDate = o && o.endDate || new Date(1990, 1, 1);
    }
}
```

3 Configuración del módulo Diary

El procedimiento para modificar el aspecto del módulo Diary, consiste en modificar los siguientes ficheros:

- diary.module.ts
- todo-listing.component.ts
- todo-listing.component.html
- todo-listing.component.scss
- todo-detail.component.ts
- todo-detail.component.html
- todo-detail.component.scss
- todo-create-update.component.ts
- todo-create-update.component.html
- todo-create-update.component.scss

3.1 diary.module.ts

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { DiaryRoutingModule } from './diary-routing.module';
import { TodoListingComponent } from './todo-listing/todo-listing.component';
import { TodoDetailComponent } from './todo-detail/todo-detail.component';
import { TodoCreateUpdateComponent } from './todo-create-update/todo-create-updat
import { MaterialDesignModule } from '../material-design/material-design.module';
import { ReactiveFormsModule } from '@angular/forms';
@NgModule({
  declarations: [TodoListingComponent, TodoDetailComponent, TodoCreateUpdateCompo
```

```
  imports: [
    CommonModule,
    MaterialDesignModule,
    ReactiveFormsModule,
    DiaryRoutingModule
  ]
})
export class DairyModule { }
```

3.2 todo-listing.component.ts

```
import { Component, OnInit } from '@angular/core';
import { FormGroup, FormBuilder, Validators } from '@angular/forms';
import { Todo } from 'src/app/models/todo';

@Component({
  selector: 'app-todo-listing',
  templateUrl: './todo-listing.component.html',
  styleUrls: ['./todo-listing.component.scss']
})
export class TodoListingComponent implements OnInit {
\textcolor{red}{  sidenavActive=true;
  todoForm: FormGroup;
  createUpdate=false;
  selectedTodo: Todo = undefined;
  selectedIndex:number = undefined;

  DiaryList: Array<Todo> = [
    { id: '1111', title:'Todo 1', description: 'description Todo 1', isCompleted:
    { id: '2222', title:'Todo 2', description: 'description Todo 2', isCompleted:
    { id: '3333', title:'Todo 3', description: 'description Todo 3', isCompleted:
    { id: '4444', title:'Todo 4', description: 'description Todo 4', isCompleted:
    { id: '5555', title:'Todo 5', description: 'description Todo 5', isCompleted:
    { id: '6666', title:'Todo 6', description: 'description Todo 6', isCompleted:
    { id: '7777', title:'Todo 7', description: 'description Todo 7', isCompleted:
    { id: '8888', title:'Todo 8', description: 'description Todo 8', isCompleted:
    { id: '9999', title:'Todo 9', description: 'description Todo 9', isCompleted:
  ];

  constructor(fb: FormBuilder) {
    this.todoForm = fb.group({
      title: ['', Validators.required],
      description: ['', Validators.required],
      isCompleted: false
    });
```

```
    this.selectedIndex = undefined;
    this.selectedTodo = undefined;
  }

  toggleCreateUpdate(){
    this.createUpdate = ! this.createUpdate;
  }

  ngOnInit() {
  }

  setSelectedValues(todo:Todo, indice:number){
    this.selectedIndex = indice;
    this.selectedTodo = todo;
    this.createUpdate = false;
  }

  processCreateUpdateTodo(todo){
    let auxTodo = JSON.parse(todo);
    if( auxTodo.index === undefined ){
      console.log('crear');
      this.DiaryList.push(auxTodo);
      this.selectedIndex = this.DiaryList.length-1;
      this.selectedTodo =  this.DiaryList[this.selectedIndex];
      this.createUpdate = false;
    }
    else{
      const indice = auxTodo.index;
      delete(auxTodo.index);
      this.DiaryList[indice]=auxTodo;
      this.selectedTodo=auxTodo;
      console.log("update");
    }
    this.createUpdate = false;
  }

  deleteTodo(index: number) {
    const todoAux = this.DiaryList[index];
    console.log(todoAux);
    this.DiaryList.splice(index, 1);
  }

  prepareUpdate(){
    this.createUpdate=true;
    let auxTodo = Object.assign({}, this.selectedTodo);
```

```
    auxTodo.index = this.selectedIndex;
    console.log(auxTodo);
    this.selectedTodo = auxTodo;
  }

  prepareCreate(){
    this.createUpdate=true;
    let auxTodo = new Todo();
    auxTodo.index = undefined;
    this.selectedTodo = auxTodo;
  }
  get date(): string {
    const fecha = new Date().toLocaleString();
    console.log(fecha);
    return fecha;
  }
  showNewTodoForm(){
    this.selectedTodo = undefined;
    this.selectedIndex = undefined;
    this.createUpdate = true;
  }
}

}
```

3.3 todo-listing.component.html

```
<ng-container>
  <mat-toolbar class="header" color="primary">
    <span>
      <img src="assets/img/angular/angular_whiteTransparent.svg" class="angular-l
      Lenin Lemus' Dairy
    </span>
    <button mat-button (click)="sidenav.toggle()">
      <mat-icon class="icon" aria-hidden="false">dehaze</mat-icon>
    </button>
    <span class="spacer"></span>
    <mat-icon routerLink="/principal" class="icon" aria-hidden="false" aria-label
    <button mat-button (click)="showNewTodoForm()">New Todo</button>
    <button mat-button (click)="sidenav.toggle()">Toggle Show-Hide Sidenav</butto
    <button mat-button (click)="toggleCreateUpdate()">Toggle Show-Hide CreateUpda
  </mat-toolbar>

  <mat-sidenav-container class="container">
```

```
    <mat-sidenav #sidenav mode="side" opened class="sidenav" >
      <ul *ngFor="let todo of DiaryList; let indice=index">
        <li (click)="setSelectedValues(todo, indice)">{{todo.title}}</li>
      </ul>
    </mat-sidenav>

    <mat-sidenav-content>
      <h4 *ngIf="!selectedTodo && !createUpdate"> Seleccione una tarea o pulse en

      <app-todo-detail *ngIf="selectedTodo && !createUpdate" [myTodo]="selectedTo

      <app-todo-create-update *ngIf="createUpdate" [myTodo]="selectedTodo" [myInd

    </mat-sidenav-content>

  </mat-sidenav-container>

  <mat-toolbar color="primary" class="footer">Footer</mat-toolbar>
</ng-container>
```

3.4 todo-listing.component.scss

```
.card {
  max-width: 400px;
}

.header-image {
  background-image: url('https://material.angular.io/assets/img/examples/shiba1.j
  background-size: cover;
}

.container {
  display: flex;
  flex-direction: column;
  position: absolute;
  position: absolute;
  top: 60px;
  bottom: 60px;
  left: 0;
  right: 0;
}

.sidenav {
  display: flex;
```

```
  align-items: center;
  justify-content: center;
  width: 200px;
  background: rgba(255, 0, 0, 0.5);
}

.header {
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
}

.footer {
  position: fixed;
  bottom: 0;
  left: 0;
  right: 0;
}
.angular-logo {
  margin: 0 4px 3px 0;
  height: 35px;
  vertical-align: middle;
}
.spacer {
  flex: 1 1 auto;
}

li { cursor: pointer; }
```

3.5 todo-detail.component.ts

```
\textcolor{red}{import { Component, OnInit, Input, Output, EventEmitter } from '@
\textcolor{red}{import { Todo } from 'src/app/models/todo';}

@Component({
  selector: 'app-todo-detail',
  templateUrl: './todo-detail.component.html',
  styleUrls: ['./todo-detail.component.scss']
})
export class TodoDetailComponent implements OnInit {
  @Input()
  myTodo:Todo;
  @Input()
  myIndex:number;
```

```
  @Output()
  eventUpdateTodo = new EventEmitter<string>();


  ngOnInit() { }

  sendEventUpdateTodo(){
    this.eventUpdateTodo.emit(`{"index": \${this.myIndex} }`);
  }
  get date(): string {
    const fecha = new Date().toLocaleString();
    console.log(fecha);
    return fecha;
  }
}
```

3.6 todo-detail.component.html

```
<form [formGroup]="todoForm" (ngSubmit)="onSubmit()">
  <div class="container">
    <mat-form-field>
      <input matInput formControlName="title" placeholder="Title">
    </mat-form-field>
    <mat-checkbox matInput formControlName="isCompleted">Is Completed?</mat-check
    <mat-form-field>
      <textarea matInput formControlName="description" rows="6" placeholder="Writ
    </mat-form-field>
    <input type="submit" name="submit">
  </div>
</form>
```

3.7 todo-detail.component.scss

```
.card {
    max-width: 400px;
  }

  .header-image {
    background-image: url('https://material.angular.io/assets/img/examples/shiba1
    background-size: cover;
  }
```

3.8 todo-create-update.component.ts

```
import { Component, OnInit, Input, Output, EventEmitter } from '@angular/core';
import { FormGroup, FormBuilder } from '@angular/forms';

@Component({
  selector: 'app-todo-create-update',
  templateUrl: './todo-create-update.component.html',
  styleUrls: ['./todo-create-update.component.scss']
})
export class TodoCreateUpdateComponent implements OnInit {

  todoForm: FormGroup;

  @Input()
  myTodo;
  @Input()
  myIndex;

  @Output()
  eventCreateUpdateTodo = new EventEmitter<string>();

  constructor(private _fb: FormBuilder) {
    this.createTodoForm();
   }

  ngOnInit(): void {
    if(this.myTodo !== undefined){
      this.todoForm.patchValue({
        title: this.myTodo.title,
        description: this.myTodo.description,
        isCompleted: this.myTodo.isCompleted
      });
    }

  }

   createTodoForm(){
    this.todoForm = this._fb.group({
       title: [''],
       description: [''],
       isCompleted:[false]
     });
   }
  onSubmit(){
```

```
    let aux = Object.assign({}, this.todoForm.value);
    if(this.myIndex === undefined ){
      aux.index=undefined;
    }
    else{
      aux.index=this.myIndex;
    }
    console.log(JSON.stringify(aux));
    this.eventCreateUpdateTodo.emit(JSON.stringify(aux));
  }
}
```

3.9 todo-create-update.component.html

```
<form [formGroup]="todoForm" (ngSubmit)="onSubmit()">
  <div class="container">
    <mat-form-field>
      <input matInput formControlName="title" placeholder="Title">
    </mat-form-field>
    <mat-checkbox matInput formControlName="isCompleted">Is Completed?</mat-check
    <mat-form-field>
      <textarea matInput formControlName="description" rows="6" placeholder="Writ
    </mat-form-field>
    <input type="submit" name="submit">
  </div>
</form>
```

3.10 todo-create-update.component.scss

```
.container {
    display: flex;
    flex-direction: column;
}

.container > * {
    width: 70%;
    margin-left: 200px;
    margin-right: 200px;
}
```

4 Aplicación en funcionamoiento

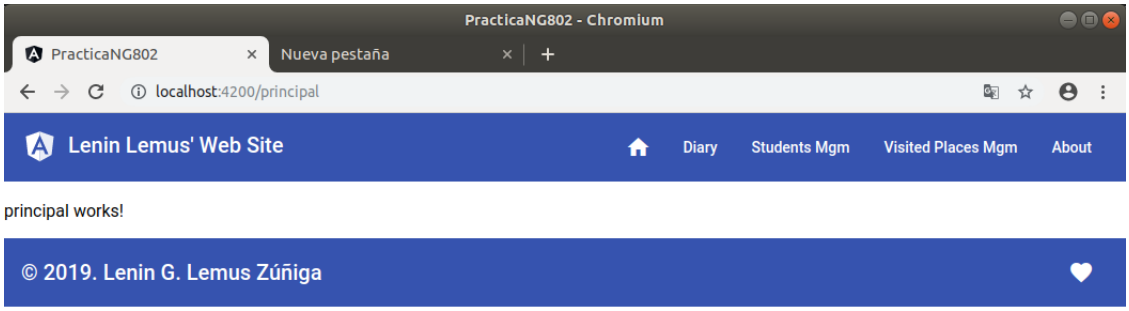

Figure 1: Pantalla principal de la Aplicación

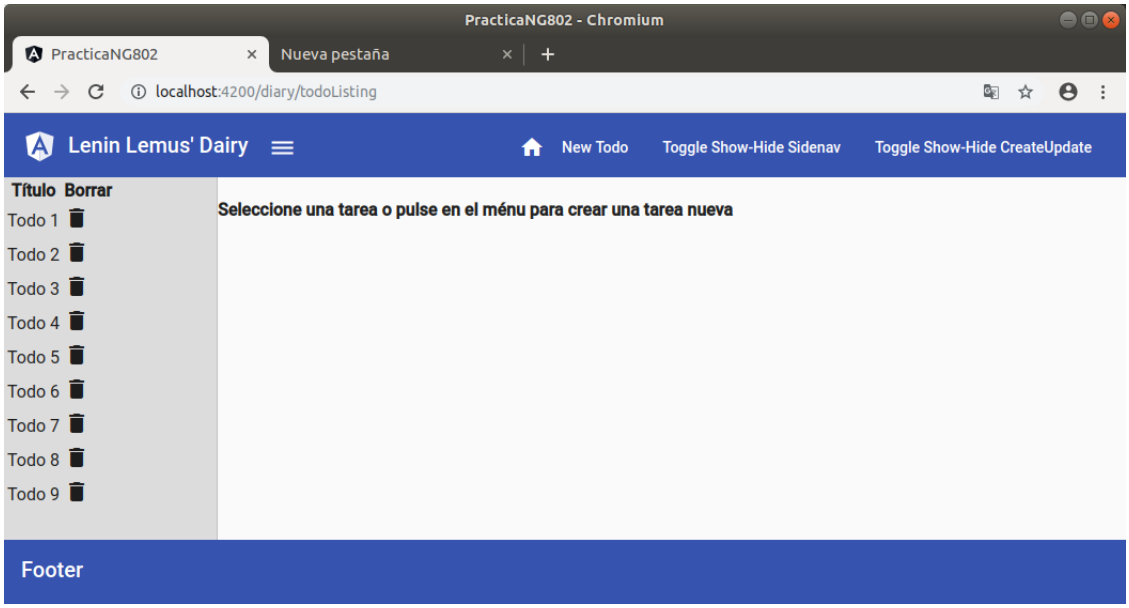

Figure 2: Aspecto de la pantalla principal de Diary

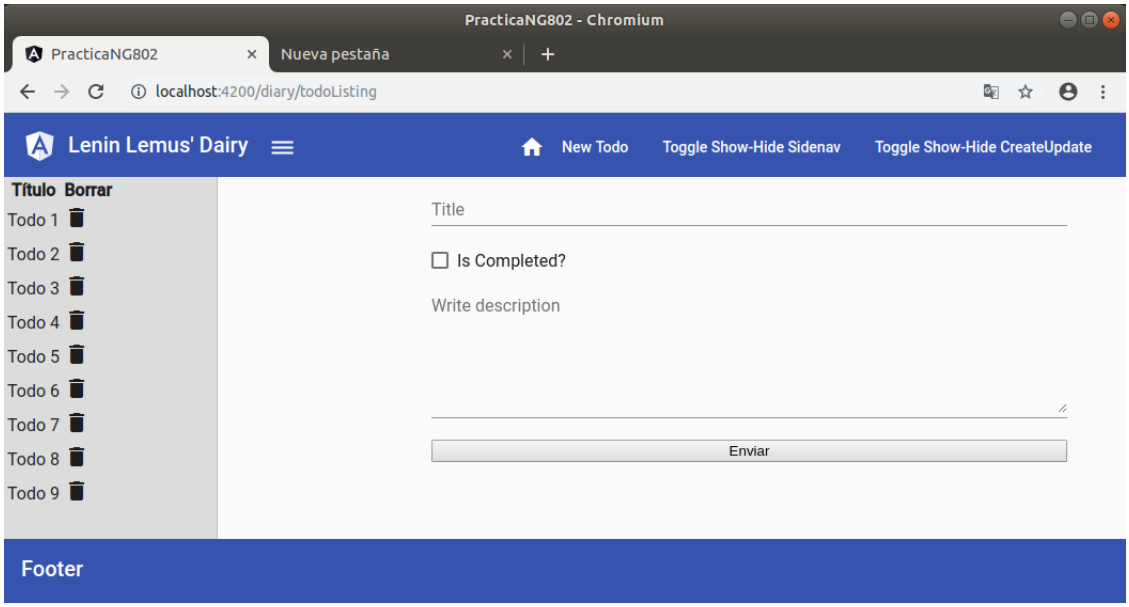

Figure 3: Al pulsar la entrada del menú "New Todo" Aparece un formulario para crear el nuevo todo.

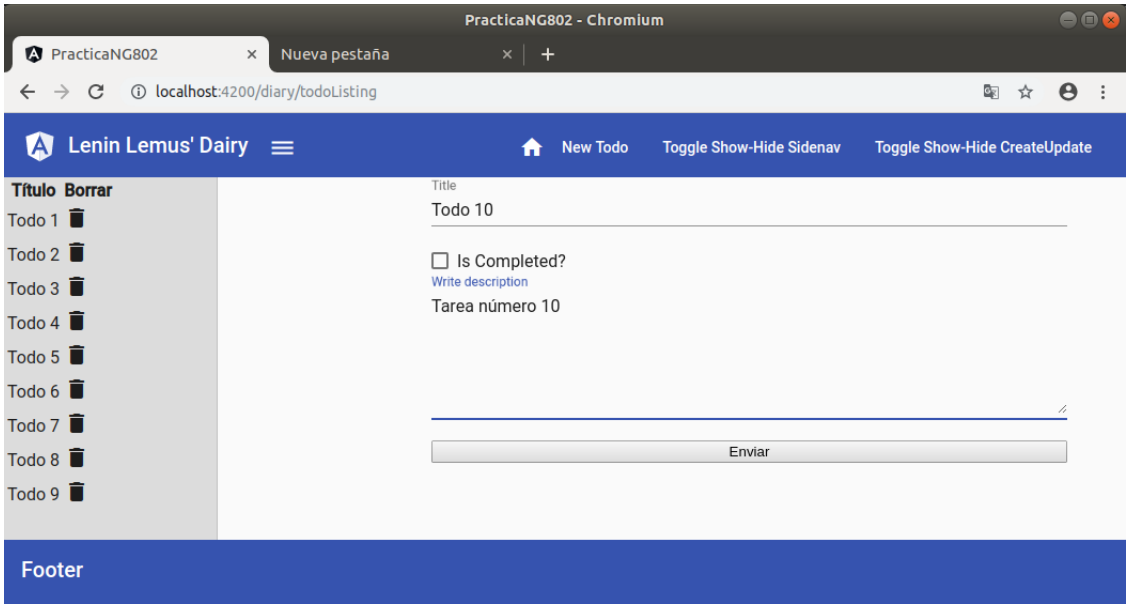

Figure 4: Una vez que se han completado los campos del todo se pusa el botón envíar para que se agregue el nuevo todo a la lista.

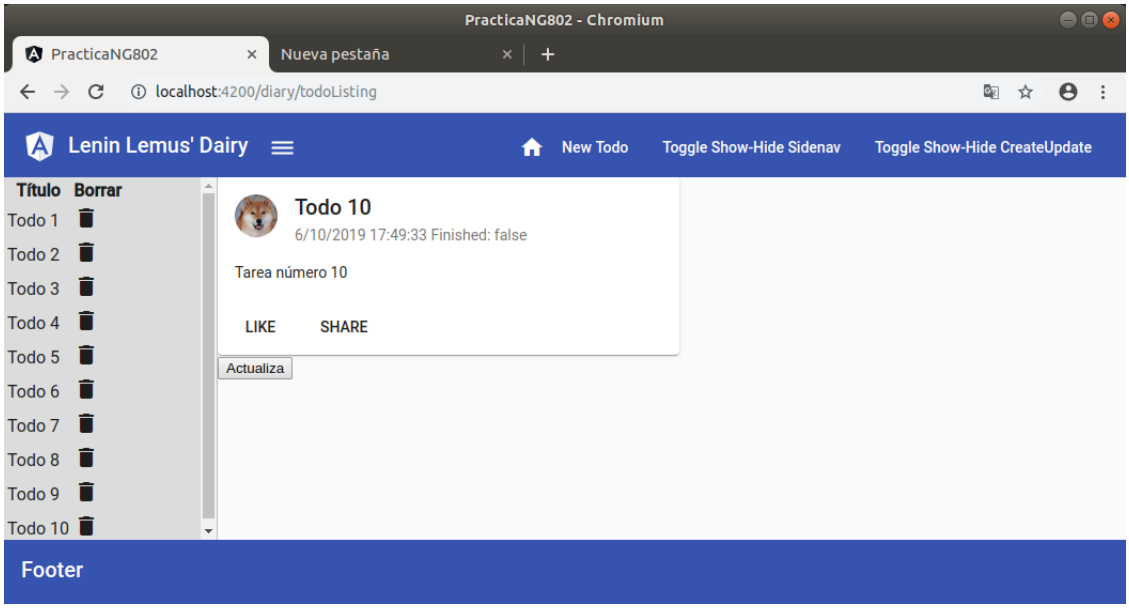

Figure 5: Datos de la tarea que ha sido agregada. Al pulsar el botón actualiza, se muestran los datos de la tarea en un formulario en donde se pueden modificar la información.

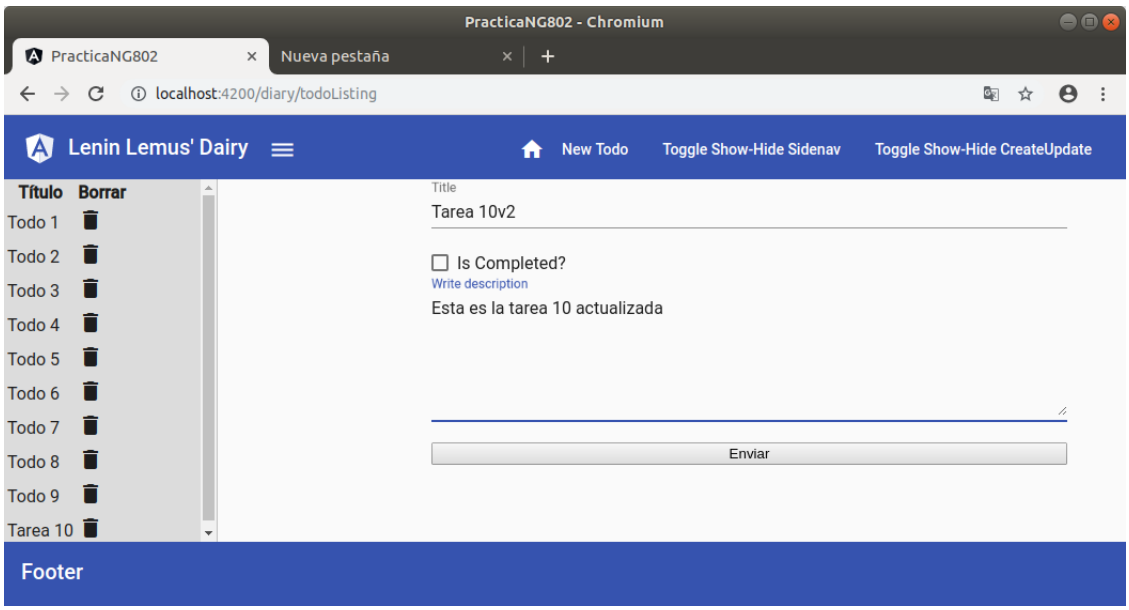

Figure 6: Si se modifican los datos de la tarea, estos cambios se almacenan en el vector al pulsar el botón enviar.

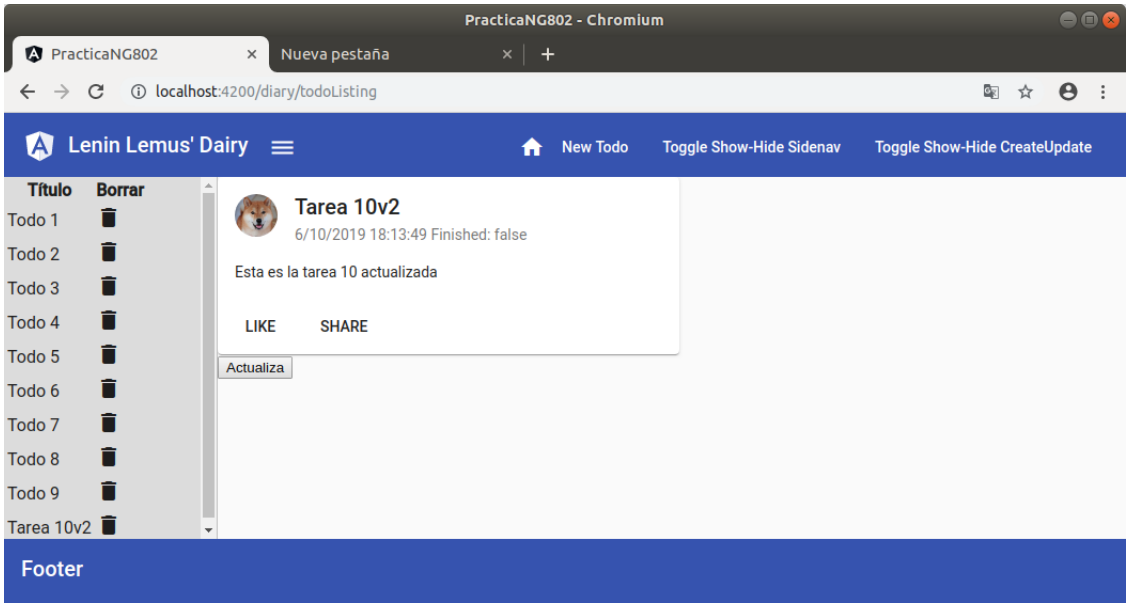

Figure 7: Tarea Actualizada.