

Modulo 13 - Reactive Extension (RxJS) y React

Las líneas de código y ejecución del programa deben usar:

- La generación de un proyecto Angular.
- La generación de componentes.
- La generación de servicios.
- Librería RxJS.
- Desarrollo de estilos CSS.
- Compilación y comprobación local en el navegador.

Desarrolla los pasos que habrá que seguir para definir todo el código de la siguiente manera:

1. Crea un proyecto Angular con la herramienta Angular CLI.
2. Genera dos componentes para formulario de tareas pendientes y lista de las mismas con su estado pendiente o realizado.
3. Genera un servicio para centralizar el uso de las tareas, con métodos para cambiar el estado de la tarea de pendiente a realizado.
4. Implementa los estilos CSS de forma global en el archivo styles.css.
5. Implementa en el servicio la librería RxJS con un método que retorne un *observable* con el número de tareas pendientes.
6. Genera un componente de barra superior para suscribir al número de tareas pendientes.
7. Comprueba el funcionamiento en el navegador.

1. Crea un proyecto Angular con la herramienta Angular CLI.

Abrimos la terminal(cmd) en el directorio donde queremos crear el proyecto y ponemos:

```
ng new test_modulo_13
```

2. Genera dos componentes para un formulario de tareas pendientes y lista de las mismas con su estado pendiente o realizado.

- Creamos 2 componentes poniendo en la terminal:

```
ng generate component todo-form  
ng generate component todo-list
```

- Creamos un modelo para representar las tareas. En la carpeta **src/app**, creamos un archivo llamado **todo.model.ts**:

```
export interface Todo {  
  id: number;  
  title: string;  
  completed: boolean;  
}
```

- Implementamos el formulario. Abrimos el archivo **todo-form.component.ts** en la carpeta **src/app/todo-form** y agregamos el siguiente código:

```
import { Component, Output, EventEmitter } from '@angular/core';
import { Todo } from '../todo.model';

@Component({
  selector: 'app-todo-form',
  templateUrl: './todo-form.component.html',
  styleUrls: ['./todo-form.component.css']
})
export class TodoFormComponent {
  newTodo: string = '';

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

  onAddTodo() {
    if (this.newTodo.trim() !== '') {
      this.addTodo.emit(this.newTodo);
      this.newTodo = '';
    }
  }
}
```

Ahora creamos un form en **todo-form.component.html**:

```
<div class="form-container">
  <input [(ngModel)]="newTodo" placeholder="Nueva tarea" />
  <button (click)="onAddTodo()">Agregar</button>
</div>
```

- Implementamos la lista de tareas.

Abrimos el archivo **todo-list.component.ts** en la carpeta **src/app/todo-list** y agregamos el siguiente código:

```
import { Component, Input } from '@angular/core';
import { Todo } from '../todo.model';

@Component({
  selector: 'app-todo-list',
  templateUrl: './todo-list.component.html',
```

```

    styleUrls: ['./todo-list.component.css']
  })
}
export class TodoListComponent {
  @Input() todos: Todo[] = [];

  toggleCompletion(todo: Todo) {
    todo.completed = !todo.completed;
  }

  removeTodo(todo: Todo) {
    this.todos = this.todos.filter((t) => t !== todo);
  }
}

```

Creamos el componente en **todo-list.component.html**:

```

<ul>
  <li *ngFor="let todo of todos" [ngClass]="{'completed':
todo.completed}">
    <input type="checkbox" [(ngModel)]="todo.completed"
(change)="toggleCompletion(todo)" />
    {{ todo.title }}
    <button (click)="removeTodo(todo)">Eliminar</button>
  </li>
</ul>

```

- Creamos el componente principal.

Abrimos el archivo **app.component.ts** en la carpeta `src/app` y agregamos el siguiente código para crear y administrar la lista de tareas:

```

import { Component } from '@angular/core';
import { Todo } from './todo.model';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  todos: Todo[] = [];
}

```

```

    addTodo(title: string) {
      const newTodo: Todo = {
        id: this.todos.length + 1,
        title: title,
        completed: false
      };
      this.todos.push(newTodo);
    }
  }
}

```

- Actualizar el archivo de la plantilla principal.

En el archivo **app.component.html**, agregamos los componentes app-todo-form y app-todo-list:

```

<div class="container">
  <app-todo-form (addTodo)="addTodo($event)"></app-todo-form>
  <app-todo-list [todos]="todos"></app-todo-list>
</div>

```

Y ahora modificamos el **app.module.ts** para agregar **FormsModule**:

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { TodoFormComponent } from './todo-form/todo-form.component';
import { TodoListComponent } from './todo-list/todo-list.component';
import { FormsModule } from '@angular/forms';

@NgModule({
  declarations: [
    AppComponent,
    TodoFormComponent,
    TodoListComponent
  ],
  imports: [

```

```

    FormsModule,
    BrowserModule,
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

3. Genera un servicio para centralizar el uso de las tareas, con métodos para cambiar el estado de la tarea de pendiente a realizado.

- Creamos un nuevo servicio llamado TodoService con el siguiente comando Angular CLI:

ng generate service todo

Ahora implementamos el servicio en **todo.service.ts** :

```

import { Injectable } from '@angular/core';
import { Todo } from '../todo.model';

@Injectable({
  providedIn: 'root',
})
export class TodoService {
  private todos: Todo[] = []; // lista de tareas

  constructor() {}

  // devuelve la lista de tareas
  getTodos(): Todo[] {
    return this.todos;
  }

  // agrega una nueva tarea a la lista de tareas
  addTodo(newTodo: string) {

    const todo: Todo = {

```

```

        id: this.todos.length + 1,
        title: newTodo,
        description: "Descripción opcional",
        completed: false,
    };
    this.todos.push(todo);
    console.log("Lista de tareas:", newTodo);
}
// borra la tarea
toggleCompletion(todo: Todo) {
    todo.completed = !todo.completed;
}

// Cambia el estado de una tarea de pendiente a realizada
markAsCompleted(index: number) {
    if (index >= 0 && index < this.todos.length) {
        this.todos[index].completed = true;
    }
}
}

```

- Registramos el servicio en el módulo principal (**app.module.ts**) en la sección providers:

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { TodoFormComponent } from './todo-form/todo-form.component';
import { TodoListComponent } from './todo-list/todo-list.component';
import { FormsModule } from '@angular/forms';
import { TodoService } from './todo.service'; // Importamos el servicio

@NgModule({
  declarations: [
    AppComponent,
    TodoFormComponent,
    TodoListComponent
  ],
  imports: [

```

```

    FormsModule,
    BrowserModule,
    AppRoutingModule
  ],
  providers: [TodoService], // Registramos el servicio aquí
  bootstrap: [AppComponent]
}))
export class AppModule { }

```

Ahora vamos a **todo-list.component.html** y modificamos el input para que cuando los usuarios marquen o desmarquen el checkbox, el método `markAsCompleted` se llamará y cambiará el estado de la tarea de "pendiente" a "realizada" .

```

<ul>
  <li *ngFor="let todo of todos" [ngClass]="{'completed':
  todo.completed}">
    <span>{{ todo.title }}</span>
    <span *ngIf="todo.completed"
class="completed-task">Realizado</span>
    <span *ngIf="!todo.completed" class="pending-task">Pendiente</span>
    <input type="checkbox" class="checkbox"
[ (ngModel) ]="todo.completed"
(change)="markAsCompleted(todos.indexOf(todo))" />
    <button (click)="removeTodo(todo)">Eliminar</button>
  </li>
</ul>

```

4. Implementa los estilos CSS de forma global en el archivo `styles.css`.

Creemos los estilos css en `styles.css`:

```

@import
url('https://fonts.googleapis.com/css?family=Lato&display=swap');

*{
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
body {

```

```
font-family: 'Helvetica Neue', sans-serif;
color: #404d5f;
margin-top: 80px;
}
```

```
.form-container {
  max-width: 940px;
  margin: 0 auto;
  padding: 2rem;
  display: flex;
  justify-content: center;
  text-align: center;
}
```

```
button{
  width: 150px;
  height: 40px;
  margin-left: 18px;
  border-radius: 30px;
  background:#047738 ;
  color: white;
  font-size: 20px;
  border: 1px solid #02602c;
}
```

```
input{
  font-size: 20px;
  border-radius: 5px;
  border: 3px solid #047738;
}
```

```
ul{
  text-align: center;
  font-size: 20px;
}
```

```
li{
  margin-bottom: 5px;
}
```

```
.completed-task {
  color: green;
  font-weight: bold;
}
```

```
span{
  margin-left: 45px;
```



```

    margin-right: 2px;
}
.pending-task {
    color: red;
    font-weight: bold;
}
.checkbox{
    padding: 15px;
    margin-right: 2px;
}

```

5. Implementa en el servicio la librería RxJS con un método que retorne un observable con el número de tareas pendientes.

Este servicio tiene un Observable llamado `pendingTasks$`, que emite el número de tareas pendientes. Este Observable se inicializa en el constructor y se actualiza cada vez que se realizan cambios en las tareas. Puedes suscribirte a `pendingTasks$` en cualquier componente que necesite conocer el número de tareas pendientes:

```

import { Injectable } from '@angular/core';
import { Todo } from '../todo.model';
import { Observable, BehaviorSubject } from 'rxjs';

@Injectable({
  providedIn: 'root',
})
export class TodoService {
  private todos: Todo[] = [];
  private pendingTasksSubject = new BehaviorSubject<number>(0);
  pendingTasks$: Observable<number> =
    this.pendingTasksSubject.asObservable();

  constructor() {
    this.updatePendingTasksCount(); // Actualiza el contador al inicio
  }

  private updatePendingTasksCount() {
    const pendingTasks = this.todos.filter((todo) =>
!todo.completed).length;
    this.pendingTasksSubject.next(pendingTasks);
  }
}

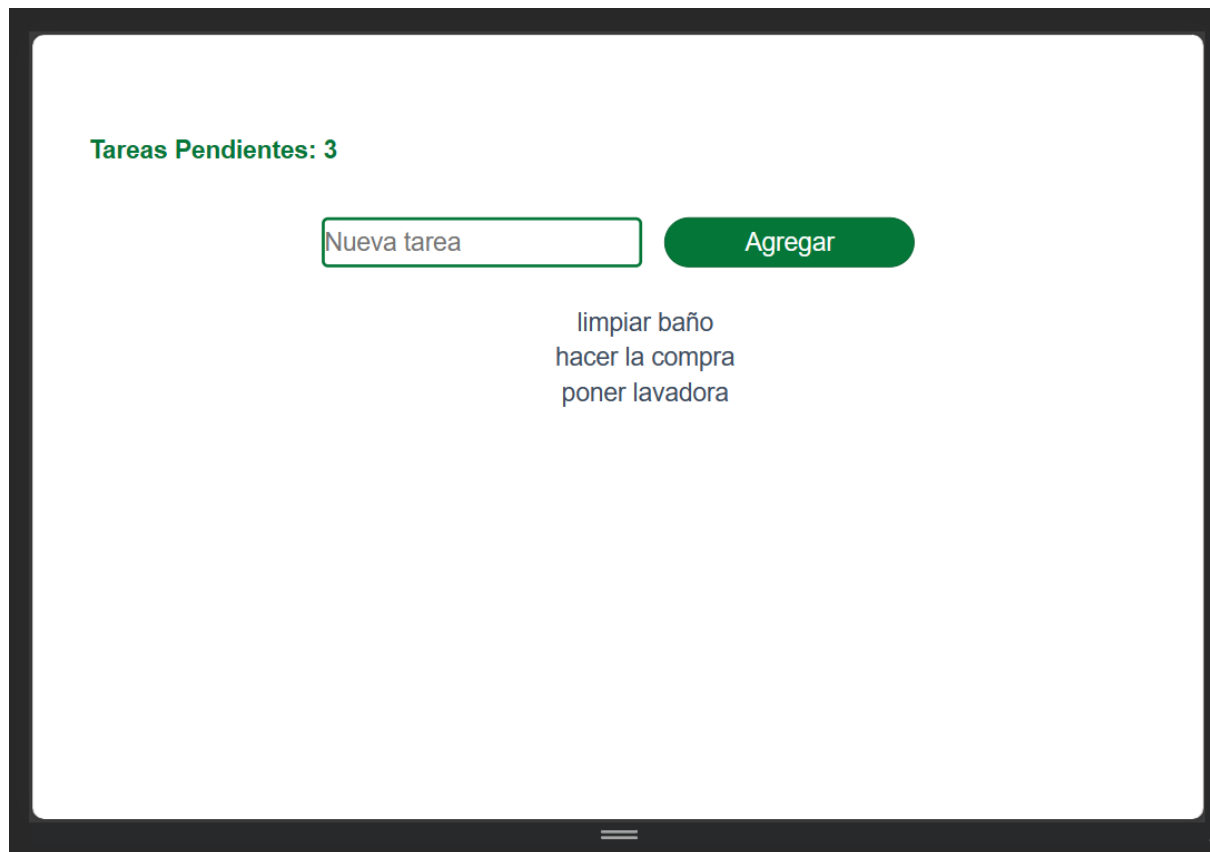
```

```
addTodo(newTodo: string) {
  const todo: Todo = {
    id: this.todos.length + 1,
    title: newTodo,
    description: "Descripción opcional",
    completed: false,
  };
  this.todos.push(todo);
  this.updatePendingTasksCount();
}

markAsCompleted(index: number) {
  if (index >= 0 && index < this.todos.length) {
    this.todos[index].completed = true;
  }
  this.updatePendingTasksCount();
}

toggleCompletion(todo: Todo) {
  todo.completed = !todo.completed;
  this.updatePendingTasksCount();
}

removeTodo(todo: Todo) {
  this.todos = this.todos.filter((t) => t !== todo);
  this.updatePendingTasksCount();
}
}
```



6. Genera un componente de barra superior para suscribir al número de tareas pendientes.
- Generamos el componente de barra superior poniendo en la terminal:

ng generate component top-bar

- En el archivo **top-bar.component.ts** hacemos lo siguiente:

```
import { Component, Input, OnInit } from '@angular/core';
import { TodoService } from '../todo.service';
import { Todo } from '../todo.model';

@Component({
  selector: 'app-top-bar',
  templateUrl: './top-bar.component.html',
  styleUrls: ['./top-bar.component.css']
})
export class TopBarComponent implements OnInit {
  @Input() todos: Todo[] = [];
  pendingTasksCount: number = 0;

  constructor(public todoService: TodoService) {}
}
```

```

ngOnInit() {
  this.todoService.pendingTasks$.subscribe(count => {
    this.pendingTasksCount = count;
  });
  this.todoService.updatePendingTasksCount();
}
}

```

- En el archivo **top-bar.component.html** agregamos el marcador de posición para mostrar el número de tareas pendientes:

```

<div class="top-bar">
  <span>
    <p class="tarear">Tareas Pendientes: {{
pendingTasksCount}}</p></span>
</div>

```

- Utilizamos el componente **top-bar** en la aplicación. Lo agregamos en el archivo HTML del componente raíz:

```

<div class="container">
  <app-top-bar [todos]="todos"></app-top-bar>

  <app-todo-form (addTodo)="addTodo($event)"></app-todo-form>
  <app-todo-list [todos]="todos"></app-todo-list>

</div>

```

- Ahora en **todo.service.ts** vamos a crear la función `toggleCompletion` y `markAsCompleted` para controlar el estado de las tareas:

```

import { Injectable } from '@angular/core';
import { Todo } from '../todo.model';
import { Observable, BehaviorSubject } from 'rxjs';

@Injectable({
  providedIn: 'root',
})
export class TodoService {

```

```

private todos: Todo[] = [];
public pendingTasksSubject = new BehaviorSubject<number>(0);
pendingTasks$: Observable<number> =
this.pendingTasksSubject.asObservable();

public updatePendingTasksCount() {
    const pendingTasks = this.todos.filter((todo) =>
!todo.completed).length;
    this.pendingTasksSubject.next(pendingTasks);
}

addTodo(newTodo: string) {
    const todo: Todo = {
        id: this.todos.length + 1,
        title: newTodo,
        description: "Descripción opcional",
        completed: false,
    };
    this.todos.push(todo);
    this.updatePendingTasksCount();
}

getTodos(): Todo[] {
    return this.todos;
}

markAsCompleted(index: number) {
    if (index >= 0 && index < this.todos.length) {
        this.todos[index].completed = true;
    }
    this.updatePendingTasksCount();
}

toggleCompletion(todo: Todo) {
    if (!todo.completed) {
        this.pendingTasksSubject.next(this.pendingTasksSubject.value - 1);
    }
    todo.completed = !todo.completed;
}

removeTodo(todo: Todo) {
    this.todos = this.todos.filter((t) => t !== todo);
    this.pendingTasksSubject.next(this.pendingTasksSubject.value - 1);
    this.updatePendingTasksCount();
}
}

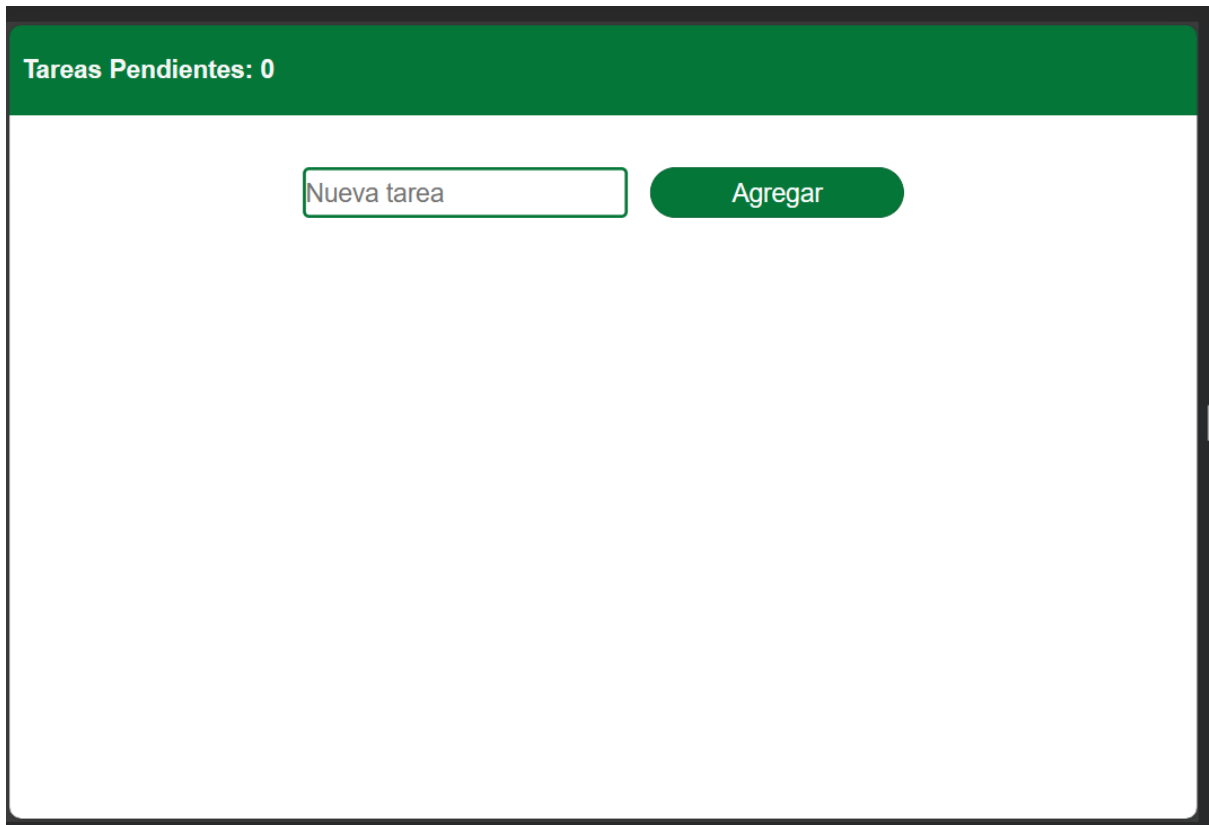
```

7. Comprueba el funcionamiento en el navegador.

El la terminal ponemos: `ng serve`

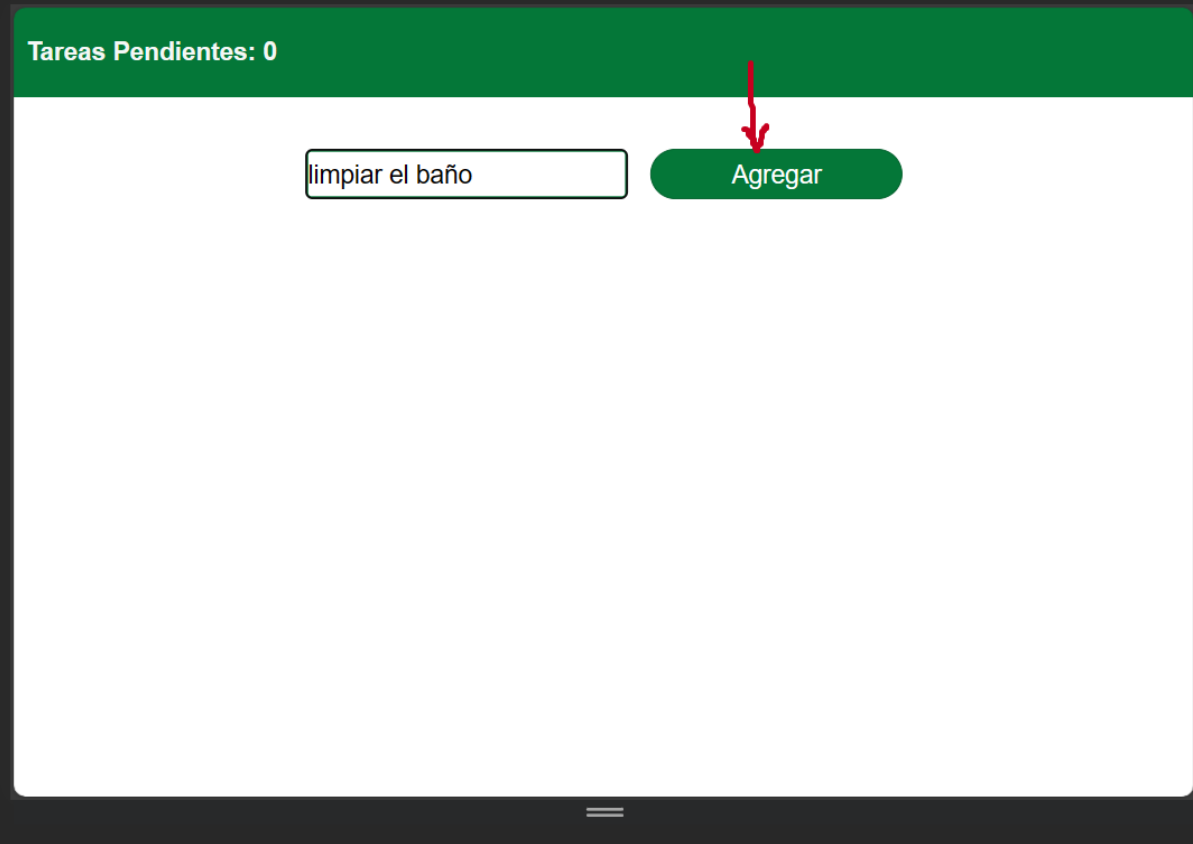
Ahora en el navegador ponemos: <http://localhost:4200/>

Aqui esta la pagina principal:



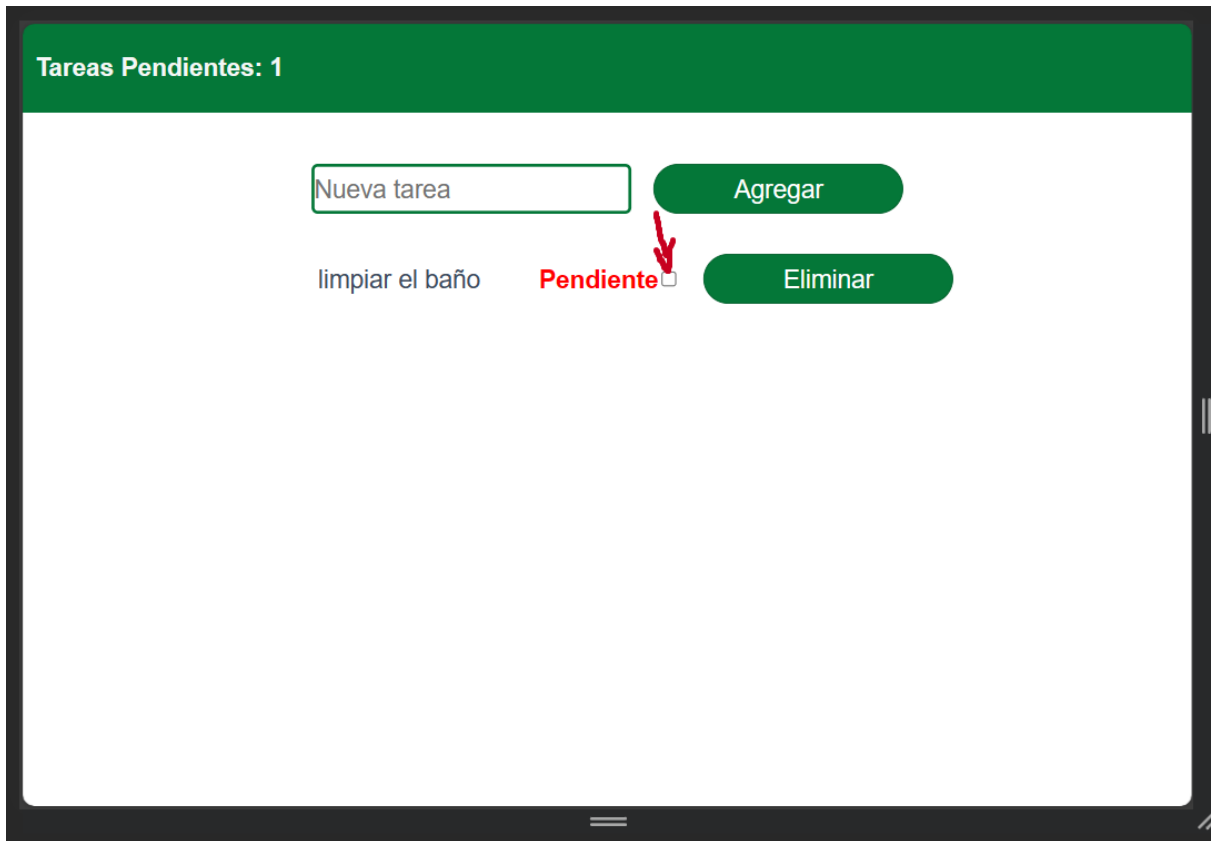
The screenshot displays a web application interface. At the top, there is a dark green header bar with the text "Tareas Pendientes: 0" in white. Below the header, the main content area is white. It features a text input field with the placeholder text "Nueva tarea" and a green button labeled "Agregar".

Insertamos tareas:

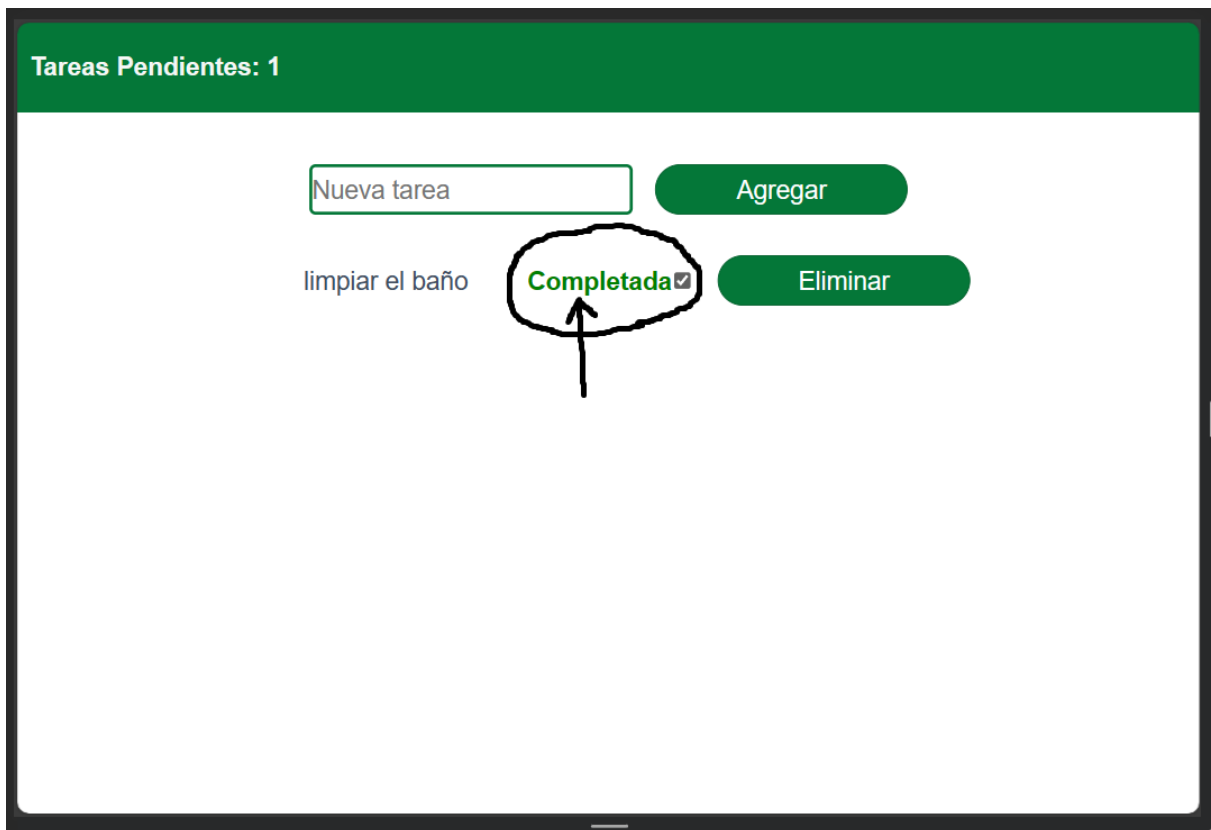


The image shows a mobile application interface for task management. At the top, there is a green header bar with the text "Tareas Pendientes: 0". Below the header, there is a white input field containing the text "limpiar el baño". To the right of the input field is a green button with the text "Agregar". A red arrow points down to the "Agregar" button. At the bottom of the screen, there is a dark grey navigation bar with a white hamburger menu icon.

Ahora marcamos la tarea pendiente:



Y se pasará a completada:



Ahora vamos a eliminar tareas:

Tareas Pendientes 3

Nueva tarea

Agregar

limpiar el baño

Completada

Eliminar

hacer la compra

Pendiente

Eliminar

hacer los deberes

Pendiente

Eliminar

Tareas Pendientes 2

Nueva tarea

Agregar

hacer la compra

Pendiente

Eliminar

hacer los deberes

Pendiente

Eliminar