Manifiesto de la Fast Task App

1. Visión General

- Objetivo: una To-Do List ligera y offline-first, que guarde datos en el navegador (Local Storage), permita filtros, animaciones y sea instalable como PWA.

- Estructura: 1. Task (modelo de datos) 2. Store (persistencia en Local Storage) 3. UI (interacción con el DOM y animaciones) 4. Service Worker (cacheo de assets y offline) 5. CSS (diseño responsivo, variables, animaciones)

2. Lógica JavaScript

2.1. Modelo de datos: class Task

- Propiedades: • id → Date.now().toString() garantiza unicidad. • text → contenido de la tarea. • completed → booleano, por defecto false. - Por qué: agrupa en un objeto todos los atributos de una tarea.

2.2. Persistencia: class Store

- Métodos estáticos (no requiere new): • getTasks(): lee y parsea JSON de localStorage. • addTask(task): añade al array y guarda. • removeTask(id): filtra y guarda. • toggleTask(id): invierte completed. - Por qué: desacopla la lógica de almacenamiento.

2.3. Interfaz y eventos: class UI

2.3.1. Inicialización - static init(): carga tareas (loadTasksFromLocal) y enlaza eventos (bindEvents).

2.3.2. Binding de eventos - Formulario (#task-form) → submit → handleAdd. - Filtros (.filter-btn) → click → handleFilter.

2.3.3. Añadir tareas (handleAdd) - preventDefault() para no recargar. - Trim del input y validación. - Creación de Task, storage y render con animación. - Limpieza del input.

2.3.4. Filtrado (handleFilter + renderFiltered) - Marca botón activo, limpia lista y repinta según filter.

2.3.5. Renderizado de cada tarea: - Crea con data-id y clases. - innerHTML con span y botones. - Listeners para completar y borrar. - Swipe-to-delete para móvil. - Append al ul.

2.3.6. Completar y eliminar: - toggleComplete(): invierte completed y togglea clase. - removeTask(): anima salida y borra en Store y DOM.

3. Service Worker (SW)

- install: precache de assets: index.html, manifest.json, CSS, JS, imágenes. - fetch: responde desde cache o red. - manifest.json: start_url, scope, display, theme_color, background_color, icons. - Registro SW en main.js: navigator.serviceWorker.register('./service-worker.js')

4. Diseño CSS

4.1. Variables globales (Root)

```
:root { --primary: #4a90e2; --secondary: rgba(245,245,245,0.8); --bg: #fff; --text: #333;
--radius: .5rem; }
```

4.2. Layout básico

- display flex y centrado. - .todo-container: fondo blanco, sombra, border-radius.

4.3. Fuentes y texto

- Fuente Exo importada. - font-weight 700 en títulos y botones.

4.4. Botones y formularios

- .task-form flex, gap. - input con focus animado y bordes redondeados. - botón hover con scale y sombra.

4.5. Lista y animaciones

- .task-item con padding, background, transition. - .enter y .exit con keyframes fadeIn/fadeOut. - .completed tacha texto y cambia color.

4.6. Footer PWA

- Barra fija bottom con fondo semitransp. y blur. - Iconos SVG con hover de hue-rotate y scale.