*Santiago Lorenzatti*

*Diseño de sistemas de Información | Universidad Tecnológica Nacional FRSFCO*

*Gestor de tareas Domésticas*

**Introducción:**

En esta segunda iteración del proyecto Gestor de Tareas Domésticas Compartidas, se incorporan nuevas funcionalidades orientadas a mejorar la experiencia de los usuarios y fomentar una participación activa en el cumplimiento de las tareas. En esta etapa se destacan dos ejes principales: por un lado, se implementa un Historial de Tareas Completadas, que permite llevar un registro claro y accesible de las responsabilidades ya realizadas. Por otro, se introduce un sistema de gamificación que asigna puntos a los usuarios según su nivel de participación, incentivando el cumplimiento de las tareas de manera lúdica y motivadora.

Estos cambios no solo enriquecen la funcionalidad del sistema, sino que también refuerzan su objetivo principal: facilitar la organización equitativa y eficiente de las tareas domésticas compartidas, promoviendo una dinámica colaborativa entre los integrantes del hogar.

**Objetivo:**

El objetivo de esta segunda iteración es continuar con la evolución del sistema Gestor de Tareas Domésticas Compartidas, mejorando su funcionalidad y fomentando un mayor compromiso de los usuarios. Se busca ofrecer herramientas que no solo permitan gestionar las tareas del hogar, sino también motivar a los usuarios mediante un sistema de puntos y recompensas virtuales (gamificación), y proporcionar visibilidad sobre el trabajo ya realizado mediante un historial claro y detallado.

**Alcance:**

Durante esta fase del desarrollo, se incorporaron las siguientes funcionalidades:

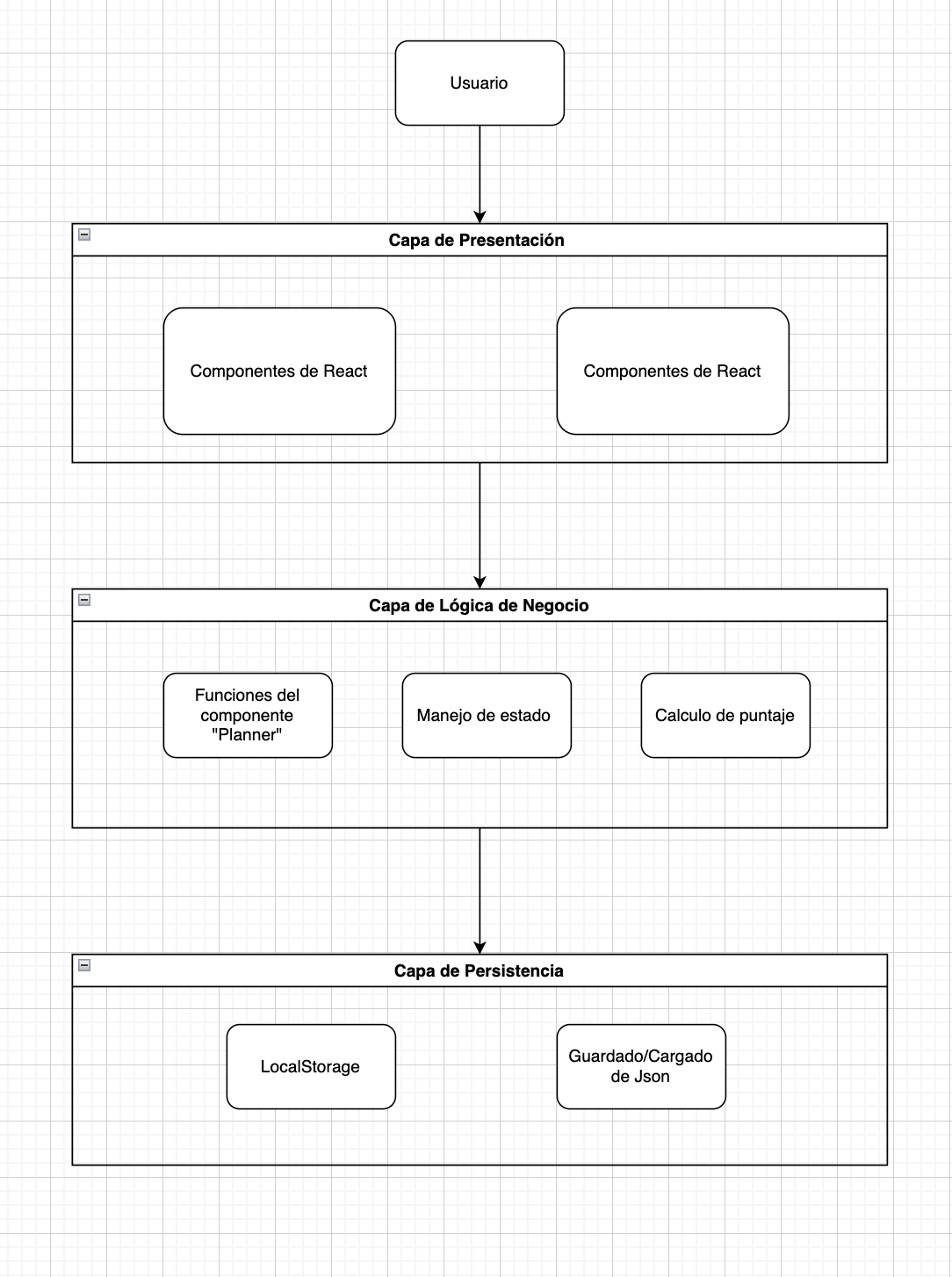
* Gamificación: implementación de un sistema de puntos asociado a los usuarios que completan tareas, con el fin de incentivar la participación de manera lúdica.
* Filtrado de tareas: los usuarios pueden alternar entre la vista de tareas activas y el historial de tareas realizadas, mejorando la usabilidad de la aplicación.
* Persistencia de datos: las tareas y el puntaje de los usuarios se almacenan localmente (localStorage) para mantener la información entre sesiones.

Estas funcionalidades se integran al núcleo de la aplicación existente, manteniendo un diseño enfocado en la simplicidad, claridad y colaboración entre los miembros del hogar.

**Elección de la arquitectura:**

Para esta segunda iteración del proyecto Gestor de Tareas Domésticas Compartidas, se optó por una arquitectura en capas. Esta decisión responde a la necesidad de mantener una estructura clara, organizada y escalable, que permita separar responsabilidades y facilitar tanto el desarrollo como el mantenimiento del sistema a lo largo del tiempo.

La arquitectura en capas permite dividir el sistema en componentes bien definidos, como presentación, lógica de negocio y acceso a datos. Esta separación mejora la reutilización de código, la legibilidad y la capacidad de realizar pruebas unitarias. Además, facilita la incorporación de nuevas funcionalidades (como la gamificación o el historial de tareas completadas) sin comprometer otras partes del sistema.



**Flujo de datos entre capas:**

El sistema implementa un flujo de datos unidireccional entre las distintas capas, lo cual garantiza orden y previsibilidad en la ejecución de las acciones. Las interacciones comienzan en la capa de presentación, donde los usuarios generan eventos como la creación, edición o finalización de una tarea. Estas acciones son procesadas por funciones que conforman la lógica de aplicación, la cual determina cómo deben modificarse los datos. Finalmente, los cambios son almacenados o leídos desde la capa de persistencia, que actualmente se implementa mediante localStorage. Este flujo garantiza la coherencia de los datos y facilita la trazabilidad de cada operación dentro del sistema.

|  |  |  |
| --- | --- | --- |
| **CAPA** | **DESCRIPCIÓN** | **EJEMPLOS EN CÓDIGO** |
| Presentación | Interfaz de usuario. Gestiona la interacción con el usuario, inputs y visualización de datos. | Componentes React: EventForm, EventList, TaskHistory, Planner, CSS aplicado |
| Lógica de Negocio (Aplicación) | Contiene las reglas del sistema, cómo se manejan las tareas, filtros, estado. No se comunica con la UI directamente | Funciones como handleAddTask, handleToggleComplete, handleDelete. |
| Persistencia / Datos | Donde se almacenan y consultan los datos. | Actualmente: localStorage. En un futuro: API RESTFUL o base de datos vía backend (Express.js, etc.) |

**Categorización de errores:**Durante el desarrollo del sistema Gestor de Tareas Domésticas Compartidas, se identificaron y clasificaron distintos tipos de errores con el objetivo de analizarlos y abordarlos de manera eficiente. Esta categorización permite no solo entender mejor el origen y la naturaleza de los problemas encontrados, sino también priorizar su resolución según el impacto que tienen en el funcionamiento general del sistema. A continuación, se detallan los errores detectados, organizados por nivel de gravedad.

|  |  |  |  |
| --- | --- | --- | --- |
| **Gravedad** | **Descripción** | **Impacto** | **Acción requerida** |
| Alta | Fallo al registrar un nuevo usuario (no se almacena en el localStorage) | Impide el acceso a nuevos usuarios, bloqueando la principal funcionalidad | Corregir de forma prioritaria |
| Alta | Las tareas no se guardan al regístralas | El sistema pierde los datos y no cumple su propósito | Revisión y testeo urgente a la capa de almacenamiento |
| Media | El sistema permite campos vacíos en la creación de tareas | Puede generar tareas incompletas o confusas | Validación adicional en Frontend y Bakend |
| Media | El sistema permite crear tareas con fechas anteriores a la actual | Puede generar confusión o una mala organización al permitir tareas en el pasado | No solicitar fecha al usuario y utilizar “Date.now()” |
| Media | El historial de tareas completadas no se muestra correctamente | No afecta a tareas activas, pero genera inconsistencias y confusión | Corrección en el renderizado del historial |
| Baja | Error en el estilo de los botones en versión mobile | Estético, sin impacto funcional | Corrección cuando haya disponibilidad |
| Baja | El espacio para la descripción permite agrandarse, deformando la estructura | Estético, sin impacto funcional | Corrección cuando haya disponibilidad |

**Endpoints de la API RESTFul:**

Si bien el sistema actualmente funciona con almacenamiento local (localStorage), en una implementación real con backend sería necesario definir una serie de métodos HTTP que permitan la gestión remota de los datos. La siguiente tabla presenta una simulación de cómo podrían estructurarse los endpoints de una API RESTful para este sistema, permitiendo operaciones CRUD sobre las tareas, acceso a los usuarios, y funcionalidades vinculadas a la gamificación.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Recurso | Método HTTP | Endpoint | Descripción | Código de respuesta |
| Tareas | GET | /api/tasks | Obtiene todas las tareas registradas | 200 OK |
| POST | /api/tasks | Crea una tarea nueva | 201 Created |
| PUT | /api/tasks/:id/completed | Marca una tarea como completada o pendiente | 200 OK |
| DELETE | /api/tasks/:id | Elimina una tarea específica por ID | 204 No content |
| Usuarios | GET | /api/users | Lista todos los usuarios registrados para asignación | 200 OK |
| Gamificación | GET | /api/users/:id/score | Devuelve el puntaje acumulado del usuario | 200 OK |
| PUT | /api/users/:id/score | Actualiza el puntaje al completar una tarea | 200 OK |