Blue text on a black background

AI-generated content may be incorrect.

Desarrollo e implantación de sistemas de software (Gpo 102)

Sprint 2 (Advanced Web) (Francisco)

Luis Omar Olmedo Ortiz (A01643557)

Rodrigo Yael Morales Luna (A01637721)

Omar Michel Carmona Villalobos (A01644146)

Luis Daniel Gacría Espinoza (A01643058)

Héctor Emil Grijalva Sánchez (A01643459)

Abril de 2025

**Descripción de la Implementación de los Requisitos Clave**

**1. Visualización de Trabajadores Disponibles**

* **Requisito:** Mostrar una lista de trabajadores disponibles con menos de 4 tareas asignadas.
* **Implementación:**
  + Se creó un endpoint en el backend (GET /api/workers/available) que consulta la colección workers en Firebase.
  + El endpoint filtra los trabajadores con menos de 4 tareas asignadas y devuelve la lista.
  + En el frontend, se utiliza fetch para obtener los datos del endpoint y mostrarlos en una tabla dentro del componente TaskAssignment.

**2. Visualización de Tareas Pendientes**

* **Requisito:** Mostrar una lista de tareas pendientes (estado "pending") en tiempo real.
* **Implementación:**
  + Se creó un endpoint en el backend (GET /api/tasks?status=pending) que consulta la colección tasks en Firebase y filtra las tareas con estado "pending".
  + En el frontend, se utiliza fetch para obtener las tareas pendientes y mostrarlas en una tabla dentro del componente TaskAssignment.

**3. Marcar Tareas como Completadas**

* **Requisito:** Permitir que los usuarios marquen una tarea como completada y que esta desaparezca de la lista de tareas pendientes.
* **Implementación:**
  + Se creó un endpoint en el backend (PUT /api/tasks/{taskId}/status) que actualiza el estado de una tarea en Firebase a "completed".
  + En el frontend, se agregó un botón en la tabla de tareas pendientes que llama a este endpoint.
  + Después de marcar una tarea como completada, se actualiza el estado local en el frontend para eliminar la tarea de la lista de pendientes.

**4. Asignación de Tareas**

* **Requisito:** Asignar tareas a trabajadores disponibles según las reglas definidas.
* **Implementación:**
  + Se configuró un cliente WebSocket en el frontend para enviar solicitudes de asignación de tareas al backend.
  + El backend selecciona un trabajador disponible utilizando reglas predefinidas (por ejemplo, asignar a trabajadores con menos tareas).
  + Las tareas asignadas se envían a través de WebSocket para actualizar la interfaz en tiempo real.

**5. Comunicación en Tiempo Real**

* **Requisito:** Actualizar la interfaz en tiempo real cuando se asignan nuevas tareas.
* **Implementación:**
  + Se configuró un servidor WebSocket en el backend para enviar actualizaciones en tiempo real.
  + En el frontend, se utilizó la biblioteca @stomp/stompjs para suscribirse a las actualizaciones y actualizar las tablas de tareas asignadas.

**6. Diseño de la Interfaz**

* **Requisito:** Mostrar tablas para trabajadores disponibles y tareas pendientes.
* **Implementación:**
  + Se diseñaron tablas en el componente TaskAssignment para mostrar los datos obtenidos del backend.
  + Se aplicaron estilos CSS para mejorar la apariencia y la usabilidad.