Construction Project Management System

Specification of SRS Software Requirements

Summary.

Specification of requirements for

construction project management system

Authors

Arellano Aramburo Jocelyn Astrid

Bravo Flores Selegna Odracir

Díaz Reyes Luis Ignacio

Maldonado Hernandez Cinthia Jazmin

Professor

Ray Brunnet Parra Galaviz

Technological University of Tijuana

Tijuana Baja California, México

Document card

Fecha	Revisión	Autor	Verificación de calidad

Por el Cliente	Por la Empresa Suministradora

CONTENTS

1. Introduction
    1.1 Purpose
    1.2 Scope
    1.3 Personnel involved
    1.4 Definitions, Acronyms and Abbreviations
    1.5 Summary
2. General Description
    2.1 Product overview
    2.2 Product functionality
    2.3 User characteristics
    2.4 Constraints
    2.5 Assumptions and dependencies
    2.6 Foreseeable system evolution
3. Specific Requirements
    3.1 External Interfaces
    3.2. Functions
    3.3 Performance Requirements
    3.4. Design Constraints
    3.5. System Attributes
4. Appendix
    4.1 Test cases
    4.2 Tablas propuestas para revisión de código
    4.3 Bases de datos de Firebase.
    4.4 Partes del código.

Document Sheet

INTRODUCTION

Se propone desarrollar una aplicación móvil para optimizar la gestión de proyectos de construcción. Permitirá la comunicación en tiempo real entre el líder del proyecto y el cliente, facilitando el seguimiento del progreso, la asignación de tareas y el ajuste de cronogramas ante retrasos por clima.

Además, mejorará la seguridad con notificaciones en tiempo real sobre personal no autorizado en el área de herramientas. Este sistema aumentará la eficiencia, optimizará la toma de decisiones y garantizará la seguridad y calidad en la construcción, haciendo la industria más segura, eficiente y transparente.

1.1 Purpose

Este sistema, a través de sensores y una aplicación móvil, facilita la comunicación en tiempo real entre el líder del proyecto de construcción y el cliente. La aplicación permite al líder crear y gestionar proyectos, asignando tareas específicas con fechas estimadas de finalización, las cuales pueden ajustarse en caso de condiciones climáticas que retrasen el progreso de la construcción.

Además, la aplicación móvil mejora la seguridad al recibir notificaciones instantáneas si se detecta la presencia de personal no autorizado en el área de almacenamiento de herramientas. Con esta herramienta, los procesos del proyecto se optimizarán, se reducirán costos y se garantizará la calidad de la obra. La visualización de datos en tiempo real permite una toma de decisiones proactiva, permitiendo a los usuarios anticiparse a posibles problemas y asegurar la ejecución exitosa del proyecto.

1.2 Scope

La aplicación móvil se centra en facilitar la comunicación y gestión en tiempo real de proyectos de construcción. Permite a los líderes de proyecto realizar un seguimiento del progreso, asignar tareas y ajustar los cronogramas según las condiciones cambiantes, como retrasos por condiciones climáticas.

Mediante la integración con sensores inteligentes, la aplicación recibe y muestra notificaciones sobre cambios ambientales críticos y riesgos de seguridad, como la presencia de personal no autorizado en áreas restringidas, como el almacén de herramientas. Los usuarios pueden gestionar los cronogramas del proyecto, supervisar las tareas asignadas y mantenerse informados a través de alertas instantáneas.

1.3 Personnel Involved

Información personal	
Nombre	Arellano Aramburo Jocelyn Astrid
Rol	Lider de proyecto
Contacto	0322103847@ut-tijuana.edu.mx
Responsabilidades	Programadora

Información del personal	
Nombre	Bravo Flores Selegna Odracir
Rol	Analista del Sistema
Contacto	0322103684@ut-tijuana.edu.mx
Responsabilidades	Programadora

Información del personal	
Nombre	Díaz Reyes Luis Ignacio
Rol	Responsable de la Documentación
Contacto	0323106001@ut-tijuana.edu.mx
Responsabilidades	Documentación y programar

Información del personal	
Nombre	Maldonado Hernandez Cinthia Jazmin
Rol	Responsable de la Documentación
Contacto	0322103754@ut-tijuana.edu.mx
Responsabilidades	Documentación y programar

1.4 Definiciones, acrónimos y abreviaturas

Nombre	Descripción
SRS	Software Requirements Specification
CMPS	Construction Proyect Managment System
JS	Java Script
PHP	Hypertext Preprocessor
MySQL	My Structured Query Language
NA	No Aplica
RF	Requerimientos Funcionales
RNF	Requerimientos no Funcionales
COD	Código
NombreRq	Nombre Requerimiento

1.5 Resumen

Este documento presenta una breve descripción sobre la funcionalidad del software planeado, así como también los involucrados en la elaboración del sistema y documentación requerida.

2. DESCRIPCIÓN GENERAL

2.1 Perspectiva del producto

El Sistema de Gestión de Proyectos de Construcción es una aplicación móvil desarrollada con React Native y JavaScript, diseñada para facilitar la creación, monitoreo y administración de proyectos de construcción. Permite a los usuarios registrarse, gestionar proyectos, asignar tareas y recibir notificaciones, mejorando la organización y eficiencia del trabajo.

La aplicación ofrece una interfaz intuitiva con una barra de navegación que incluye "Inicio", "Proyectos Archivados", "Notificaciones" y "Perfil", brindando acceso rápido a las funcionalidades principales. Su estructura modular permite futuras expansiones y mejoras.

2.2 Funcionalidad del producto

El sistema proporciona herramientas clave para la gestión eficiente de proyectos de construcción. A continuación, se detallan sus principales funcionalidades:

2.2.1 Registro e inicio de sesión

- Los usuarios pueden registrarse con un correo y una nueva contraseña.
- Durante el registro, deben ingresar su número de teléfono y un nombre de usuario.
- Los usuarios pueden iniciar sesión directamente si ya tienen una cuenta.

2.2.2 Creación y gestión de proyectos

- Desde la pantalla de inicio, los usuarios pueden crear proyectos pulsando el botón "Crear Proyecto".
- Se ingresan los datos del proyecto y, al crearse, aparece en la lista de proyectos activos.
- Al seleccionar un proyecto, se accede a su información general, lista de tareas y miembros asignados.

2.2.3 Roles y permisos de usuarios

- Los gerentes pueden invitar a nuevos miembros al proyecto con los roles de:
    - **Gerente**: Puede invitar a otros gerentes y usuarios de menor rango, asignar tareas y finalizar proyectos.
    - **Supervisor**: Puede ver y gestionar tareas asignadas, pero no crear o finalizar proyectos.
    - **Miembro**: Solo puede ver sus tareas y completarlas.

2.2.4 Gestión de tareas

- Los gerentes pueden crear tareas y asignarlas a los miembros del proyecto, estableciendo niveles de prioridad.
- Al completar una tarea, el usuario debe agregar un comentario obligatorio sobre su ejecución.
- Las tareas asignadas aparecen en la sección de notificaciones del usuario.

2.2.5 Notificaciones y comunicación

- Los usuarios reciben notificaciones cuando:
    - Son invitados a un proyecto.
    - Se les asigna una tarea.
    - Hay actualizaciones importantes en el proyecto.

2.2.6 Archivado de proyectos

- Los proyectos finalizados se mueven a la sección "Proyectos Archivados".
- En esta sección, los proyectos solo pueden consultarse en modo de solo lectura.

2.3 Características de los usuarios

2.3.1 Gerente

- Responsable de la gestión general del proyecto.
- Puede invitar a otros gerentes y miembros.
- Asigna tareas y finaliza proyectos.

2.3.2 Supervisor

- Supervisa las tareas asignadas y el progreso del proyecto.
- Puede gestionar tareas propias y colaborar en la ejecución del proyecto.

2.3.3 Miembro

- Recibe tareas asignadas por el gerente.
- Debe completar sus tareas y agregar comentarios sobre su ejecución.

2.4 Restricciones

- **Tiempo de desarrollo**: El sistema debe completarse en un plazo de tres meses.
- **Disponibilidad de hardware**: No requiere sensores externos, pero depende de la conectividad de los dispositivos.
- **Infraestructura tecnológica**: Se requiere conexión a internet para la sincronización en tiempo real.
- **Presupuesto limitado**: Se utilizarán herramientas y servicios gratuitos o de bajo costo.
- **Capacitación de usuarios**: La aplicación debe ser intuitiva para usuarios con conocimientos básicos de tecnología.
- **Compatibilidad de dispositivos**: Compatible con dispositivos Android (9+) e iOS (12+).
- **Conectividad en entornos remotos**: Se debe manejar la falta de conexión con mecanismos de almacenamiento temporal.
- **Procesamiento de datos**: La base de datos Firestore debe optimizarse para evitar problemas de rendimiento.

2.5 Suposiciones y dependencias

2.5.1 Suposiciones

- Se espera que los usuarios tengan dispositivos compatibles con la aplicación.
- Se asume que los gerentes invitarán correctamente a los miembros.
- Se espera que los usuarios registren y completen tareas dentro del flujo de trabajo establecido.

2.5.2 Dependencias

- **Dependencia del hardware**: La funcionalidad depende de que los usuarios tengan dispositivos compatibles.
- **Dependencia de la infraestructura tecnológica**: Firestore es esencial para el almacenamiento y sincronización de datos.
- **Dependencia de la conectividad**: Algunas funcionalidades pueden verse afectadas sin conexión a internet.
- **Dependencia de la capacitación**: Se requiere una guía inicial para nuevos usuarios.

2.6 Evolución previsible del sistema

El sistema está diseñado para futuras mejoras, incluyendo:

- **Soporte offline**: Posibilidad de trabajar sin conexión y sincronizar datos posteriormente.
- **Expansión de reportes**: Generación de informes detallados sobre progreso y productividad.
- **Integración con otras herramientas**: Posibilidad de conectarse con sistemas externos para mejorar la gestión de proyectos.
- **Mejoras en seguridad**: Implementación de autenticación multifactor y cifrado de datos.
- **Alertas avanzadas**: Notificaciones inteligentes basadas en actividad y urgencia de las tareas.

3. Requisitos específicos

Código	Nombre Requerimiento	Tipo	Fuente del requerimiento	Prioridad del requerimiento	Descripción del requerimiento
RF-001	Recepción de datos en tiempo real	Funcional	Gerente	Alta	La aplicación debe recibir datos de sensores de temperatura y presencia a través de servicios web.
RF-002	Notificaciones críticas	Funcional	Gerente	Alta	La aplicación debe alertar al usuario en tiempo real sobre condiciones climáticas adversas que afecten el proceso constructivo y la presencia no autorizada en zonas restringidas
RF-003	Visualización de datos	Funcional	Gerente	Alta	La aplicación debe mostrar un panel sencillo con niveles actuales de temperatura y así como el estado de las zonas críticas con detalles de seguridad.
RF-005	Historial de alertas	Funcional	Gerente	Alta	Permite consultar un registro de las alertas y notificaciones generadas por los sensores.
RF-006	Gestión de usuarios	Funcional	Gerente	Alta	Autenticación de usuarios con roles básicos (administrador y cliente).
RF-007	Gestión de proyecto	Funcional	Cliente	Alta	La aplicación debe permitir el registro, seguimiento y actualización de las tareas que conforman el proyecto de construcción

Requerimientos no funcionales

Código	Nombre de requerimiento	Tipo	Fuente del requerimiento	Prioridad del requerimiento	Descripción del requerimiento
RNF-001	Tiempo de respuesta	No funcional	Usuarios finales	Alta	El sistema debe procesar consultas y registros en un tiempo no mayor a 3 segundos bajo carga normal.
RNF-002	Interfaz intuitiva	No funcional	Usuarios Finales	Media	La aplicación debe ser fácil de usar, con un diseño responsivo que permita el acceso desde diferentes tamaños de dispositivos móviles.
RNF-003	Encriptación de datos	No funcional	Departamento de seguridad	Alta	El sistema debe contar con encriptación de datos sensibles.
RNF-004	Disponibilidad	No funcional	Gerente, IT	Alta	El sistema debe estar disponible 24/7 con un tiempo de inactividad no mayor al 1% mensual.
RNF-005	Compatibilidad multiplataforma	No funcional	Usuarios finales	Media	Desarrollar la aplicación en React Native para su compatibilidad con Android e iOS.

3.1. Interfaces Externas

3.1.1. Interfaz de Usuario (UI)

- **Aplicación Móvil:** La interfaz de usuario de la aplicación móvil debe ser intuitiva y fácil de usar, con un diseño responsivo adaptado a dispositivos Android e iOS. Debe incluir funcionalidades para la visualización de datos en tiempo real, gestión de tareas, notificaciones y configuración de alertas. La navegación debe ser fluida y optimizada para una experiencia de usuario eficiente en entornos de obra.

3.1.2. Interfaz de Hardware

- **Sensores:** El sistema debe ser capaz de conectarse y recibir datos de sensores ambientales, como temperatura y presencia. La comunicación con estos sensores debe ser estable y segura, utilizando protocolos como HTTP, garantizando una baja latencia y transmisión confiable.

3.1.3. Interfaz de Comunicación

- **Notificaciones Push:** El sistema debe integrarse con servicios de notificaciones en tiempo real para alertar a los usuarios sobre eventos críticos, como cambios ambientales o detección de presencia en zonas restringidas.
- **APIs Externas:** El sistema debe ser capaz de interactuar con APIs externas para complementar su funcionalidad, permitiendo, por ejemplo, la obtención de datos meteorológicos o la integración con sistemas ERP utilizados en la gestión de proyectos de construcción.

3.2. Funciones

3.2.1. Monitoreo Ambiental

- **Recopilación de Datos:** El sistema debe recopilar datos en tiempo real desde sensores instalados en la obra.

- **Visualización de Datos:** Los datos deben ser procesados y representados gráficamente en la aplicación móvil y web, permitiendo un análisis intuitivo de las condiciones ambientales.
- **Alertas Ambientales:** Se deben generar notificaciones cuando los valores de temperatura o humedad excedan los límites configurados, asegurando una respuesta oportuna ante condiciones adversas.

3.2.2. Gestión de Seguridad

- **Detección de Presencia:** El sistema debe detectar actividad en zonas no autorizadas mediante sensores de presencia, activando protocolos de seguridad.
- **Alertas de Seguridad:** Notificaciones inmediatas deben ser enviadas a los administradores y personal autorizado en caso de intrusiones.
- **Configuración de Zonas Críticas:** Los usuarios deben poder configurar y modificar zonas de acceso restringido a través de la aplicación web.

3.2.3. Visualización y Análisis de Datos

- **Gráficos Históricos:** Los usuarios deben poder visualizar gráficos históricos de temperatura, humedad y eventos de seguridad para el análisis de tendencias.
- **Exportación de Informes:** Los datos registrados deben poder exportarse en formato PDF para su análisis y documentación.

3.2.4. Reportes Automatizados

- **Generación de Reportes:** El sistema debe generar reportes periódicos que incluyan condiciones ambientales, incidentes de seguridad y progreso del proyecto.

3.2.5. Gestión de Tareas

- **Creación de Tareas:** Los usuarios deben poder crear y asignar tareas con fechas de finalización estimadas.
- **Ajuste de Fechas:** Se debe permitir la reprogramación de tareas en función de imprevistos, como retrasos por condiciones climáticas.

3.3. Requisitos de Rendimiento

3.3.1. Disponibilidad

- **Tiempo de Actividad:** El sistema debe garantizar una disponibilidad del 99.9%, asegurando su funcionamiento continuo para los usuarios.

3.3.2. Escalabilidad

- **Crecimiento de Usuarios:** La arquitectura del sistema debe permitir el crecimiento en número de usuarios sin degradar el rendimiento.

3.4. Restricciones de Diseño

3.4.1. Compatibilidad

- **Dispositivos Móviles:** La aplicación móvil debe ser compatible con Android e iOS.
- **Navegadores Web:** La aplicación web debe ser compatible con los principales navegadores sin requerir configuraciones adicionales.

3.4.2. Seguridad

- **Encriptación:** Todas las transmisiones de datos entre la aplicación móvil, la aplicación web y los servidores deben estar encriptadas para garantizar la seguridad y privacidad de la información.

3.4.3. Usabilidad

- **Interfaz Intuitiva:** La interfaz debe ser fácil de usar, reduciendo el tiempo de aprendizaje requerido para los usuarios.

3.5. Atributos del Sistema

3.5.1. Fiabilidad

- **Tolerancia a Fallos:** El sistema debe ser capaz de recuperarse automáticamente de fallos en los sensores o en la conexión a internet.

3.5.2. Mantenibilidad

- **Modularidad:** La arquitectura del sistema debe ser modular, facilitando la incorporación de nuevas funcionalidades en el futuro.
- **Documentación:** Todo el sistema debe contar con documentación detallada para garantizar su mantenimiento y evolución.

3.5.3. Portabilidad

Multiplataforma: La aplicación móvil debe ser desarrollada en React Native, garantizando su portabilidad entre Android e iOS.

Compatibilidad con Navegadores: La aplicación web debe ser compatible con los principales navegadores sin necesidad de configuraciones adicionales.

4. APÉNDICE

4.1 Test cases

Test Case ID	Test Case Description	Pre condition	Test Steps	Test Data	Expected Results	Actual Results	Pass / Fail
01A	Inicio de sesión	Estar registrado. Estar en la aplicación. No tener un inicio de sesión activo.	1. Abrir la aplicación. 2. Ingresar el usuario. 3. Ingresar la contraseña. 4. Hacer click en el botón de login.	user: nacho password: ********	Login exitoso	Login exitoso	Pass
02A	Inicio de sesión con datos incorrectos	Estar registrado. Estar en la aplicación. No tener un inicio de sesión activo.	1. Abrir la aplicación. 2. Ingresar el usuario. 3. Ingresar la contraseña. 4. Hacer click en el botón de login.	user: nacho password: ********	Login no permitido Mensaje de error	El login no será exitoso	Pass
03A	Inicio de sesión sin datos	Estar registrado. Estar en la aplicación. No tener inicio de sesion activo	1. Presionar el botón de iniciar sesion 2. No ingresar datos 3. Presionar el botón de iniciar sesion	user: password:	Login no permitido Mensaje de error	El login no será exitoso	Pass
04A	Registrarse	Estar en la aplicación. No estar registrado.	1. Presionar el botón de crear cuenta 2. Ingresar correo y contraseña para el registro 3. Presionar el botón de registrarse 4. Ingresar tu nombre de usuario 5. Ingresar un número de teléfono	Correo: prueba@gmail.com Contraseña: ******** Nombre de usuario: prueba Teléfono: 1234567890	Registro de cuenta exitoso	Registro exitoso	Pass
05A	Registrarse sin datos	Estar en la aplicación. No estar registrado.	1. Presionar el botón de crear cuenta. 2. No ingresar ni un correo ni una contraseña. 3. Presionar el botón de registrarse.	correo: contraseña:	No registro realizado	Registro no exitoso	Pass
06A	Registrarse sin usuario	Estar en la aplicación. No estar registrado. Ya haber registrado un correo y contraseña.	1. Presiona el botón de crear cuenta. 2 Ingresa correo y contraseña. 3. Hacer clic en el botón Registrarse 4. Hacer clic en continuar	correo: jarellanomx@gmail.com password: ******** Usuario: Teléfono:	No cambios realizados	No cambios realizados	Pass
07A	Nuevo Proyecto	Estar en la aplicación. Haber realizado el inicio de sesión o registro éxitosamente	1. Hacer clic en Nuevo Proyecto 2. Poner un nombre del proyecto 3. Poner la Ubicación 4. Asignar Fecha de inicio y Fecha final. 5. Poner el cliente 6. Poner la descripción del proyecto. 7. Hacer clic en guardar	Nombre proyecto: Proyecto Prueba Ubicación: Universidad Tecnologica de Tijuana Fecha Inicio: 2025-03-01 Fecha Final: 2025-07-15 Cliente: Ignacio Diaz Descripción: Proyecto de prueba	Proyecto guradado	Proyecto guardado	Pass
08A	Nuevo Proyecto sin datos	Estar en la aplicación. Haber realizado el inicio de sesión o registro éxitosamente	1. Hacer clic en Nuevo Proyecto 2. Dejar los datos vacios 3. Hacer clic en guardar	Nombre proyecto: Ubicación: Fecha Inicio: Fecha Final: Cliente: Descripción:	Mensaje de error: Agregar datos solicitados	Mensaje de error	Pass
09A	Nuevo Proyecto con fecha de inicio anterior a la fecha actual	Estar en la aplicación. Haber realizado el inicio de sesión o registro éxitosamente	1. Hacer clic en Nuevo Proyecto 2. Poner un nombre del proyecto 3. Poner la Ubicación 4. Asignar Fecha de inicio y Fecha final. 5. Poner el cliente 6. Poner la descripción del proyecto. 7. Hacer clic en guardar	Nombre proyecto: Proyecto Prueba Ubicación: Universidad Tecnologica de Tijuana Fecha Inicio: 2024-03-01 Fecha Final: 2025-07-15 Cliente: Ignacio Diaz Descripción: Proyecto de prueba	Mensaje de error: La fecha de incio ya paso	Mensaje de error	Pass
10A	Nuevo Proyecto con fecha final anterior a la fecha de inicio	Estar en la aplicación. Haber realizado el inicio de sesión o registro éxitosamente	1. Hacer clic en Nuevo Proyecto 2. Poner un nombre del proyecto 3. Poner la Ubicación 4. Asignar Fecha de inicio y Fecha final. 5. Poner el cliente 6. Poner la descripción del proyecto. 7. Hacer clic en guardar	Nombre proyecto: Proyecto Prueba Ubicación: Universidad Tecnologica de Tijuana Fecha Inicio: 2025-03-01 Fecha Final: 2025-02-15 Cliente: Ignacio Diaz Descripción: Proyecto de prueba	Mensaje de error: La fecha final no puede ser anterior a la fecha de inicio	Mensaje de error	Pass

11A	Agregar nueva tarea	Estar en la aplicación. Haber realizado el inicio de sesión o registro éxitosamente. Seleccionar un proyecto	1. Hacer clic en detalles del proyecto 2. Hacer clic en Nueva Tarea. 3. Poner un titulo de la tarea. 4. Poner una descripción. 5. Asignar a un miembro. 6. Asignarle una prioridad. 7. Darle una fecha de vencimiento. 8. Darle clic en guardar	Tarea: Tarea 1 2. Descripción: Prueba de una tarea. 3. Asignar a: prueba 4. Prioridad: Media. 5. Fecha de vencimiento: 2025-03-21	Nueva tarea creada	Nueva tarea creada	Pass
12A	Agregar nueva tarea con datos vacíos	Estar en la aplicación. Haber realizado el inicio de sesión o registro éxitosamente. Seleccionar un proyecto	1. Hacer clic en detalles del proyecto 2. Hacer clic en Nueva Tarea. 3. Poner un titulo de la tarea. 4. Poner una descripción. 5. Asignar a un miembro. 6. Asignarle una prioridad. 7. Darle una fecha de vencimiento. 8. Darle clic en guardar	1. Tarea: 2. Descripción: 3. Asignar a: 4. Prioridad: 5. Fecha de vencimiento:	Mensaje de error: Favor de llenar los datos solicitados	Mensaje de error	Pass
13A	Agregar nueva tarea con la fecha superior a la fecha final del proyecto	Estar en la aplicación. Haber realizado el inicio de sesión o registro éxitosamente. Seleccionar un proyecto	1. Hacer clic en detalles del proyecto 2. Hacer clic en Nueva Tarea. 3. Poner un titulo de la tarea. 4. Poner una descripción. 5. Asignar a un miembro. 6. Asignarle una prioridad. 7. Darle una fecha de vencimiento. 8. Darle clic en guardar	1. Tarea: Tarea 1 2. Descripción: Prueba de una tarea. 3. Asignar a: prueba 4. Prioridad: Media. 5. Fecha de vencimiento: 2026-03-21	Mensaje de error: La fecha de la tarea no puede pasar la fecha final del proyecto	Mensaje de error	Pass
14A	Asignar una tarea a un miembro que no esta en el proyecto	Estar en la aplicación. Haber realizado el inicio de sesión o registro éxitosamente. Seleccionar un proyecto	1. Hacer clic en detalles del proyecto 2. Hacer clic en Nueva Tarea. 3. Poner un titulo de la tarea. 4. Poner una descripción. 5. Asignar a un miembro. 6. Asignarle una prioridad. 7. Darle una fecha de vencimiento. 8. Darle clic en guardar	1. Tarea: Tarea 1 2. Descripción: Prueba de una tarea. 3. Asignar a: nadie 4. Prioridad: Media. 5. Fecha de vencimiento: 2026-03-21	Mensaje de error: Asigne la tarea a un miembro	Mensaje de error	Pass
15A	Invitar miembro al proyecto	Estar en la aplicación. Haber realizado el inicio de sesión o registro éxitosamente. Seleccionar un proyecto El miembro a invitar ya debe tener una cuenta	1. Hacer clic en Personal 2. Hacer clic en Invitar 3. Agregar el correo electrónico del miembro a añadir. 4. Asignarle un Rol. 5. Hacer clic en Invitar	Correo: prueba@gmail.com Rol: Empleado	Invitación Enviada	Invitación Enviada	Pass
16A	Invitar a un miembro que ya esta en el proyecto	Estar en la aplicación. Haber realizado el inicio de sesión o registro éxitosamente. Seleccionar un proyecto El miembro a invitar ya debe tener una cuenta	1. Hacer clic en Personal 2. Hacer clic en Invitar 3. Agregar el correo electrónico del miembro a añadir. 4. Asignarle un Rol. 5. Hacer clic en Invitar	Correo: prueba@gmail.com Rol: Empleado	Mensaje de error: El usuario ya esta en el proyecto	Mensaje de error	Pass
17A	Invitar a un miembro con datos vacíos	Estar en la aplicación. Haber realizado el inicio de sesión o registro éxitosamente. Seleccionar un proyecto El miembro a invitar ya debe tener una cuenta	1. Hacer clic en Personal 2. Hacer clic en Invitar 3. No agregar datos 4. Hacer clic en Invitar	Correo: Rol:	Mensaje de error: Favor de llenar los datos solicitados	Mensaje de error	Pass

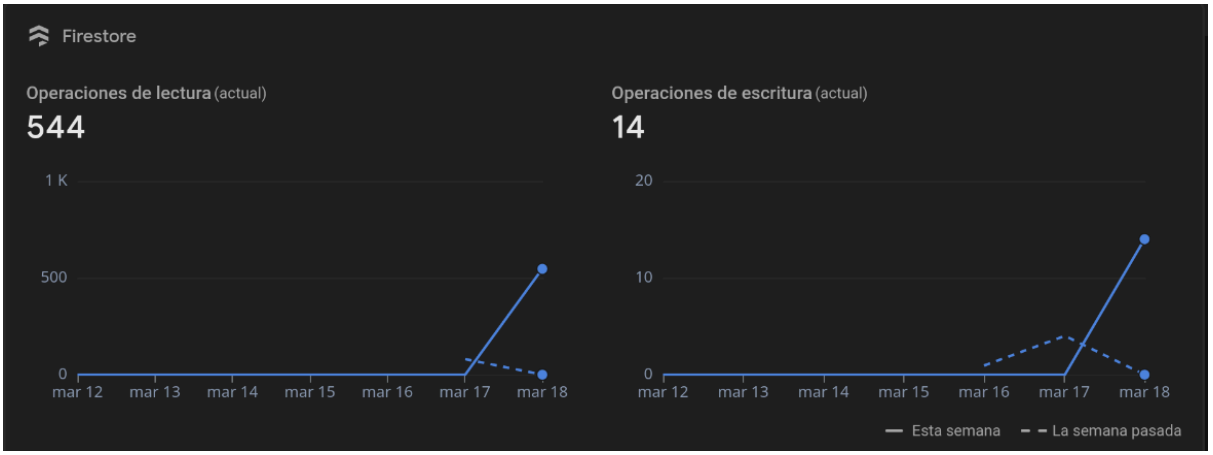
4.2 Tablas propuestas para revisión de código.

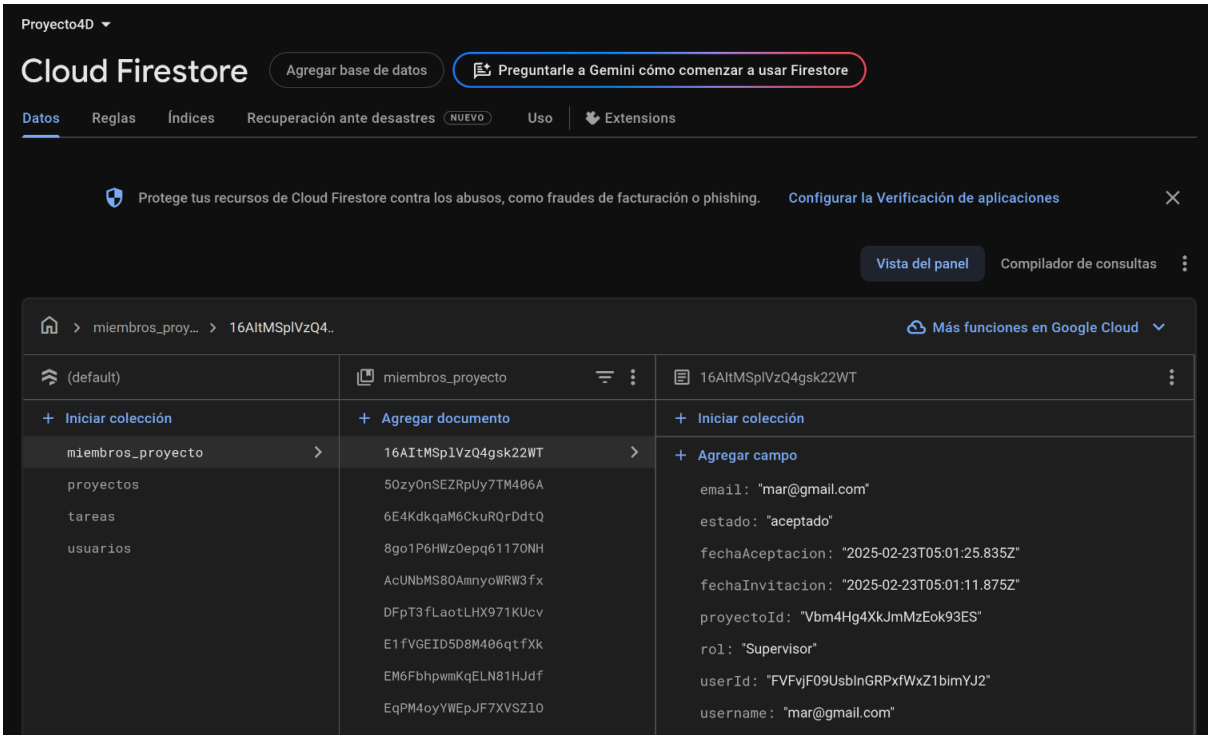
ID	REVISOR	Fecha	Archivo	Comentarios	Estado
1	Luis Diaz	3-1-25	Login.js	Código limpio y funcional	Aceptado
2	Selegna B.	3-2-25	Registro.js	Código validado y funcional	Aceptado
3	Selegna B.	3-2-25	usuarioInfo.js	Información válida	Aceptado
4	Luis Diaz	3-2-25	homeScreen.js	Información acorde a la base de datos	Aceptado
5	Jazmin M.	3-5-25	proyectosScreen.js	Pantalla sin correcciones notorias	Aceptado
6	Astrid A.	3-5-25	homeScreen.js	Creación de proyectos sin fallos graves	Aceptado
7	Luis Diaz.	3-6-2	homeScreen.js	Botón de detalles funcional	Aceptado
8	Astrid A.	3-7-25	infoTareas.js	Buscador de tareas en función	Aceptado
9	Jazmin M.	3-8-25	notiScreen.js	Boton de notificaciones en buen estado	Aceptado
10	Luis Diaz.	3-16-2	proyectosScreen.js	Invitación de miembros en orden	Aceptado

ID	Funcionalidad	Caso de prueba	Resultado esperado	Estado
TC-01	Login	Ingresar credenciales correctas	Acceso permitido	Aprobado
TC-02	Registro	No rellenar los campos de registro	Registro fallido	Aprobado
TC-03	Crear proyecto	Campos vacíos	Creación fallida	Aprobado
TC-04	Crear proyecto	Campos rellenados	Creación exitosa	Aprobado
TC-05	Ver detalles del proyecto	Clickear botón	Redirección de pantalla	Aprobado
TC-06	Finalizar proyecto	Clickear botón	Finalización de proyecto	Aprobado
TC-07	Invitar miembro	Campo de invitación vacío	Rellenar campo	Aprobado
TC-08	Invitar miembro	Campos llenos	Invitación enviada	Aprobado

TC-09	Ver tareas	Clickear botón	Redireccionamiento de pantalla	Aprobado

4.3 Base de datos firebase.





4.4 Partes del código

App Main

```
const app = initializeApp(firebaseConfig)
export const auth = getAuth(app)

const Stack = createStackNavigator()

export default function App() {
  return (
<NavigationContainer>
      <Stack.Navigator initialRouteName="Login" screenOptions={{ headerShown: false }}>
        <Stack.Screen name="Login" component={LoginScreen} />
        <Stack.Screen name="Signup" component={SignupScreen} />
        <Stack.Screen name="UserData" component={UserDataScreen} />
        <Stack.Screen name="Home" component={HomeScreen} />
        <Stack.Screen name="InfoProyectos" component={InfoProyectosScreen} />
        <Stack.Screen name="Proyectos" component={ProyectosArchivadosScreen} />
        <Stack.Screen name="Notificaciones" component={NotificacionesScreen} />
        <Stack.Screen name="ListaTareas" component={ListaTareasScreen} options={{ headerShown: false }}
/>
      </Stack.Navigator>
    </NavigationContainer>

  )
}
```

- Firebase: Para autenticación.
- React Navigation: Para la navegación entre pantallas.

Utiliza un StackNavigator con la pantalla de "Login" como inicio y oculta los encabezados. Define pantallas como "Login", "Signup", "Home", etc.

SignUpScreen

```
export default function SignupScreen({ navigation }) {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [loading, setLoading] = useState(false);

  const [request, response, promptAsync] = Google.useIdTokenAuthRequest({
    clientId: "830529089599-3gc714c6p6qvcq3po5qvcfa0okl76js6.apps.googleusercontent.com"
  });

  const handleSignup = async () => {
    if (!email.trim() || !password.trim()) {
      alert("Por favor completa todos los campos");
      return;
    }

    setLoading(true);
    try {
      await createUserWithEmailAndPassword(auth, email, password);
      navigation.replace("UserData");
    } catch (error) {
      console.error("Signup error:", error);
      alert("Error al registrarse: " + error.message);
    } finally {
      setLoading(false);
    }
  };
```

Este código implementa la pantalla de registro en React Native, permitiendo a los usuarios crear cuentas mediante correo electrónico y contraseña, o con Google. Utiliza Firebase Authentication para gestionar la creación de usuarios y la autenticación con Google. Valida los campos, muestra un indicador de carga y navega a la siguiente pantalla tras el registro exitoso, manejando también los errores.

LoginScreen

```
export default function LoginScreen({ navigation }) {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [loading, setLoading] = useState(false);

  const handleLogin = async () => {
    if (!email.trim() || !password.trim()) {
      alert("Por favor completa todos los campos");
      return;
    }

    setLoading(true);
    try {
      await signInWithEmailAndPassword(auth, email, password);
      navigation.replace("Home");
    } catch (error) {
      console.error("Login error:", error);
      alert("Error al iniciar sesión: " + error.message);
    } finally {
      setLoading(false);
    }
  };
```

Maneja el inicio de sesión:

- Valida email y contraseña.
- Usa Firebase Authentication para iniciar sesión.
- Muestra "cargando" durante el proceso.
- Navega a "Home" si se logra ingresar, o muestra un error.

HomeScreen

```
export default function HomeScreen({ navigation }) {
  const [proyectos, setProyectos] = useState([])
  const [modalVisible, setModalVisible] = useState(false)
  const [nuevoProyecto, setNuevoProyecto] = useState({
    nombre: "",
    ubicacion: "",
    fechaInicio: "",
    fechaFin: "",
    estado: "En Progreso",
    cliente: "",
    descripcion: "",
  })

  useEffect(() => {
    cargarProyectos()
  }, [])

  const cargarProyectos = async () => {
    try {
      const datos = await obtenerProyectos()
      setProyectos(datos)
    } catch (error) {
      alert("Error al cargar proyectos: " + error.message)
    }
  }

  const handleLogout = async () => {
    try {
      await signOut(auth)
      navigation.replace("Login")
    } catch (error) {
      console.error("Logout error:", error)
      alert("Error al cerrar sesión: " + error.message)
    }
  }
```

Maneja la pantalla principal de la app:

- **Muestra proyectos:** Recupera y muestra una lista de proyectos.
- **Añade proyectos:** Permite agregar nuevos proyectos con validación de campos.
- **Actualiza proyectos:** Cambia el estado de los proyectos a "Finalizado".
- **Elimina proyectos:** Borra proyectos.
- **Detalles del proyecto:** Navega a la pantalla de detalles de un proyecto.
- **Cierra sesión:** Permite al usuario cerrar la sesión.

InfoProyectos

```
const cargarEstadisticas = async () => {
  try {
    const stats = await obtenerEstadisticasTareas(proyecto.id);
    setEstadisticas(stats);
  } catch (error) {
    console.error("Error al cargar estadísticas:", error);
    throw error;
  }
};

const cargarMiembros = async () => {
  try {
    const miembrosData = await obtenerMiembrosProyecto(proyecto.id);
    setMiembros(miembrosData);
  } catch (error) {
    console.error("Error al cargar miembros:", error);
    throw error;
  }
};

const cargarTareas = async () => {
  try {
    const tareasData = await obtenerTareasProyecto(proyecto.id);
    setTareas(tareasData);
  } catch (error) {
    console.error("Error al cargar tareas:", error);
    throw error;
  }
};
```

- **Carga estadísticas de tareas:** obtiene y almacena estadísticas de tareas para un proyecto.
- **Carga miembros del proyecto:** obtiene y almacena la lista de miembros de un proyecto.
- **Carga tareas del proyecto:** obtiene y almacena la lista de tareas de un proyecto.

ListaTareas

```
const miembrosFiltrados = miembros.filter(miembro =>
  miembro.username.toLowerCase().includes(busquedaUsuario.toLowerCase())
);

useEffect(() => {
  cargarDatos();
}, []);

const cargarDatos = async () => {
  try {
    setLoading(true);
    const [tareasData, miembrosData] = await Promise.all([
      obtenerTareasProyecto(proyecto.id),
      obtenerMiembrosProyecto(proyecto.id)
    ]);

    let tareasFiltradas = tareasData;
    if (userRole === "Empleado") {
      tareasFiltradas = tareasData.filter(
        tarea => tarea.asignadoA === auth.currentUser?.uid
      );
    } else if (userRole === "Supervisor") {
      const empleadosIds = miembrosData
        .filter(m => m.rol === "Empleado")
        .map(m => m.userId);
      tareasFiltradas = tareasData.filter(
        tarea => tarea.asignadoA === auth.currentUser?.uid ||
                 empleadosIds.includes(tarea.asignadoA)
      );
    }
```

- Filtra miembros por nombre.
- Carga tareas y miembros al inicio.
- Filtra tareas por rol ("Empleado", "Supervisor" y).
- Actualiza el estado y maneja errores.

Notificaciones

```
export default function NotificacionesScreen() {
  const [notificaciones, setNotificaciones] = useState([]);
  const [refreshing, setRefreshing] = useState(false);

  const cargarNotificaciones = useCallback(async () => {
    try {
      const data = await obtenerNotificaciones(auth.currentUser.uid);
      setNotificaciones(data);
    } catch (error) {
      console.error('Error al cargar notificaciones:', error);
    }
  }, []);

  const onRefresh = useCallback(async () => {
    setRefreshing(true);
    await cargarNotificaciones();
    setRefreshing(false);
  }, [cargarNotificaciones]);

  useEffect(() => {
    cargarNotificaciones();
  }, [cargarNotificaciones]);

  const handleAceptarInvitacion = async (invitacionId) => {
    try {
      await aceptarInvitacion(invitacionId);
      await cargarNotificaciones();
    } catch (error) {
      console.error('Error al aceptar invitación:', error);
    }
  };
```

Carga y muestra:

- Recupera notificaciones del usuario.
- Permite recargar la lista.

Gestiona invitaciones:

- Permite aceptar y rechazar invitaciones.
- Recarga notificaciones tras aceptar.

ProyectosArchivados

```
export default function ProyectosArchivadosScreen({ navigation }) {
  const [proyectosPendientes, setProyectosPendientes] = useState([]);
  const [proyectosFinalizados, setProyectosFinalizados] = useState([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    cargarProyectos();
  }, []);

  const cargarProyectos = async () => {
    try {
      const [pendientes, finalizados] = await Promise.all([
        obtenerProyectos("Pendiente"),
        obtenerProyectos("Finalizado")
      ]);

      setProyectosPendientes(pendientes);
      setProyectosFinalizados(finalizados);
      setLoading(false);
    } catch (error) {
      console.error("Error al cargar proyectos:", error);
      setLoading(false);
    }
  };
```

Muestra proyectos archivados:

- **Separa proyectos:**
  - Recupera y muestra proyectos "Pendientes" y "Finalizados" por separado.
- **Carga datos:**
  - Utiliza useEffect para cargar los proyectos al montar el componente.
  - Utiliza Promise.all para cargar ambos tipos de proyectos de manera simultánea.
- **Gestión de carga:**
  - Muestra un indicador de carga mientras se obtienen los datos.
  - Maneja los errores.